Representing Multi-Parent Organizational Structures for Use in High Performance Computing Resource Scheduling Algorithms

Lloyd T. Brown
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd
Part of the Educational Administration and Supervision Commons

BYU ScholarsArchive Citation
https://scholarsarchive.byu.edu/etd/1982

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Representing Multi-parent Organizational Structures for use in High-Performance Computing Resource Scheduling Algorithms

Lloyd T. Brown

A thesis submitted to the faculty of Brigham Young University in partial fulfillment of the requirements for the degree of Master of Science

Michael G. Bailey, Chair
Barry M. Lunt
Quinn O. Snell

School of Technology Brigham Young University April 2010

Copyright © 2010 Lloyd T. Brown All Rights Reserved
ABSTRACT

Representing Multi-parent Organizational Structures for use in High-Performance Computing Resource Scheduling Algorithms

Lloyd T. Brown
School of Technology
Master of Science

Historically, organizational structures of many universities and corporations have followed a strictly tree-based, hierarchical model. These organizations are defined with no more than one parent organization, and typically resource requirements for the organization could be derived from the parent organization. In recent years, however, many institutions have created interdisciplinary research groups which incorporate multiple fields of research across multiple campus organizations. For example, at Brigham Young University, there exists a biophysics research group, a child organization of both the Department of Biology and the Department of Physics, making it unclear how to define its resource requirements in the context of multiple parents from diverse colleges.

As computing resources are allocated to organizations, the requirements of those organizations must be taken into account. However, when organizations have multiple parent organizations, it is unclear which restrictions or allocations are appropriate for the organization, as shown with the biophysics research group described above. Extending the example, if a campus high-performance computing facility restricts resources on an organizational basis, and the Biology and Physics departments are allocated different resource levels, the newly formed biophysics group will need system administrators’ intervention to assure appropriate resource allocation.

This document describes a versatile system for modeling organizational structure, including defining multiple parent organizations, the inheritance of arbitrary properties from parent to children, and, when inherited attributes conflict, includes an extensible mechanism for defining conflict resolution policies. This system allows for arbitrary parameters to be applied at any level of the organizational structure. This inherited information can then be used for resource allocation of the campus high performance computing facility.

Keywords: organizational structure, multiparent, HPC, multiple inheritance
ACKNOWLEDGMENTS

Thanks on this project go to the committee members, who continued to push me along throughout the process. Also to Chris, whose \LaTeX\template (Monson, 2006), and extensive debugging knowledge made typesetting this document significantly easier. And most of all, to LeAndra, who both never gave up on me, and never let me give up on myself.
# TABLE OF CONTENTS

List of Figures ................................................. ix
List of Tables .................................................. xi
List of Code Segments .......................................... xiii

1 Introduction .................................................... 1
   1.1 Statement of Problem .................................. 2
   1.2 Use Cases ................................................. 6
      1.2.1 Use Case 1: Simple Tree Structure ............... 7
      1.2.2 Use Case 2: Multi-Parent Tree Structure ....... 9
   1.3 Research Questions ...................................... 11
   1.4 Scope and Limitations ................................... 11

2 Literature Review ............................................ 13
   2.1 Introduction ............................................. 13
   2.2 Directory Services ...................................... 13
   2.3 HPC Scheduling .......................................... 16
      2.3.1 Priority Calculations ............................... 17
   2.4 Grids ...................................................... 20
      2.4.1 Virtual Organizations ............................... 20
   2.5 Data Structures .......................................... 22
      2.5.1 Graph ................................................ 22
      2.5.2 Tree ................................................. 26
   2.6 Conclusion ................................................. 28

3 Methodology .................................................. 31
   3.1 System Overview ......................................... 31
   3.2 Data Model ................................................ 32
      3.2.1 Data Structure ...................................... 32
## 3.2.2 Data Storage

## 3.2.3 Data Serialization

## 3.3 Tools

## 3.4 Software Licensing

## 3.5 Software Design

- 3.5.1 Periodic Operations
- 3.5.2 Integration with External Systems

## 3.6 Testing

- 3.6.1 Compliance and Correctness Testing
- 3.6.2 Performance Testing
- 3.6.3 Value Demonstration

## 4 Results

### 4.1 Correctness Testing

- 4.1.1 Simple, Valid Case
- 4.1.2 Complex, Valid Case
- 4.1.3 Disconnected Case
- 4.1.4 Cyclic Case

### 4.2 Inheritance Testing

- 4.2.1 Graph Construction
- 4.2.2 Results

### 4.3 Performance Testing

- 4.3.1 Test Specifications
- 4.3.2 Platform Specifications
- 4.3.3 Results

### 4.4 Value Demonstration

### 4.5 Code Samples

- 4.5.1 Daemonization
4.5.2 Checking for File Updates ........................................ 74
4.5.3 Reading Configuration File and Command-line Arguments .... 75
4.5.4 Dynamic Loading of Functions .................................. 76
4.5.5 Graph Traversal .................................................... 77
4.5.6 Data Serialization .................................................. 79
4.5.7 Conflict Resolution Functions .................................... 80
4.5.8 Data Export ......................................................... 81

5 Conclusions .................................................................. 83

5.1 Research Summary ................................................... 83
5.2 Value of Research ..................................................... 85
5.3 Further Research ....................................................... 86
  5.3.1 Graph Algorithm Limitations ................................... 86
  5.3.2 Users’ Relationship to Organizations ......................... 88
  5.3.3 User Interface ..................................................... 88
5.4 Conclusion ............................................................... 89

References ..................................................................... 91

A Software Licensing ....................................................... 95

A.1 Software License Association ....................................... 95
A.2 License Texts ........................................................... 95
  A.2.1 GNU GPL v2.0 .................................................... 95
  A.2.2 GNU LGPL v2.1 .................................................. 100
  A.2.3 Modified BSD (“3 clause”) License ......................... 106
  A.2.4 Python Software Foundation License, v2 ................ 106
  A.2.5 Creative Commons Attribution 3.0 ....................... 110

B Code Segments ............................................................. 115

B.1 XML Code .............................................................. 115
  B.1.1 XML Document Type Definition ......................... 115
## LIST OF FIGURES

1.1 Flat, Mutually Exclusive, Organizational Structure ........................................ 3
1.2 Organizational Structure with Multiparent Organization ................................. 4
1.3 Example of Hierarchical, Tree-based Organizational Structure ......................... 5
1.4 Simple Tree-based Inheritance Using an Override Policy ............................... 8
1.5 Multi-Parent Tree-based Inheritance using a Keep-Maximum Policy .................. 10

2.1 Examples of Hierarchical VO s ........................................................................ 23
2.2 Visual Representations of Basic Graphs .......................................................... 24
2.3 Visual Representations of Basic Trees .............................................................. 27

3.1 Data Serialization Example Graph ................................................................. 35
3.2 Overall Recursive Flow for Graph Traversal .................................................... 38
3.3 Simple, Valid Test Case ................................................................................... 41
3.4 Valid Organizational Structure with Non-single Root ..................................... 42
3.5 Invalid, Disconnected Organizational Structure .............................................. 42
3.6 Invalid, Cyclic Organizational Structures ...................................................... 43
3.7 Inheritance Test Case Graph Assuming a Keep-Average policy ..................... 45
3.8 Organizational Inheritance Diagram for B Simulations .................................. 50

4.1 Example of Regular Tree with Depth=2 and Branching Factor=3 .................... 61
4.2 Performance Tests: Average Traversal Time .................................................... 64
4.3 Scenario 1A Simulated Processors Allocated per User .................................. 67
4.4 Scenario 1B Simulated Processors Allocated Per User .................................. 67
4.5 Scenario 1A Simulated Processor-seconds Queued Per User ....................... 68
4.6 Scenario 1B Simulated Processor-seconds Queued Per User ....................... 68
4.7 Scenario 2A Simulated Processors Allocated Per User .................................. 69
## LIST OF TABLES

3.1 Statically Assigned and Inherited Values for Graph in Figure 3.7 . . . . . . . 46

4.1 Inheritance Correctness Testing Results . . . . . . . . . . . . . . . . . . . . 60

4.2 Performance Tests: Average Traversal Time . . . . . . . . . . . . . . . . . . 63

A.1 Software/Documentation Utilized and Associated Licenses . . . . . . . . . . 95
## LIST OF CODE SEGMENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Code/Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>GraphML Code for Graph in Figure 3.1</td>
<td>34</td>
</tr>
<tr>
<td>4.1</td>
<td>Code to Build Valid, Moderately Complex Test Case</td>
<td>52</td>
</tr>
<tr>
<td>4.2</td>
<td>Result of Structure Validation for Valid, Moderately Complex Test Case</td>
<td>53</td>
</tr>
<tr>
<td>4.3</td>
<td>Code to Build Valid Organizational Structure with Non-single Root</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>Result of Structure Validation for Valid Structure with Non-single Root</td>
<td>55</td>
</tr>
<tr>
<td>4.5</td>
<td>Code to Build Invalid, Disconnected Structure</td>
<td>55</td>
</tr>
<tr>
<td>4.6</td>
<td>Result of Structure Validation for Invalid, Disconnected Structure</td>
<td>56</td>
</tr>
<tr>
<td>4.7</td>
<td>Code to Build Invalid, Simple Cyclic Organizational Structure</td>
<td>57</td>
</tr>
<tr>
<td>4.8</td>
<td>Code to Build Invalid, Complex Cyclic Organizational Structure</td>
<td>58</td>
</tr>
<tr>
<td>4.9</td>
<td>Result of Structure Validation Tests for Cyclic Scenarios</td>
<td>59</td>
</tr>
<tr>
<td>4.10</td>
<td>Code to Daemonize a Program</td>
<td>73</td>
</tr>
<tr>
<td>4.11</td>
<td>Calling the Daemonize Tool</td>
<td>74</td>
</tr>
<tr>
<td>4.12</td>
<td>Checking for File Updates</td>
<td>74</td>
</tr>
<tr>
<td>4.13</td>
<td>Reading and Parsing Command-line Arguments</td>
<td>75</td>
</tr>
<tr>
<td>4.14</td>
<td>Parsing Configuration Files</td>
<td>76</td>
</tr>
<tr>
<td>4.15</td>
<td>Dynamic Loading of Functions</td>
<td>76</td>
</tr>
<tr>
<td>4.16</td>
<td>Graph Traversal Algorithm</td>
<td>77</td>
</tr>
<tr>
<td>4.17</td>
<td>Data Serialization</td>
<td>79</td>
</tr>
<tr>
<td>4.18</td>
<td>Conflict Resolution Function Examples</td>
<td>81</td>
</tr>
<tr>
<td>4.19</td>
<td>Data Export</td>
<td>81</td>
</tr>
<tr>
<td>B.1</td>
<td>XML DTD for Data Structure</td>
<td>115</td>
</tr>
<tr>
<td>B.2</td>
<td>XML Code for Inheritance Testing Graph</td>
<td>116</td>
</tr>
<tr>
<td>B.3</td>
<td>Conflict Resolution Example Code</td>
<td>117</td>
</tr>
<tr>
<td>B.4</td>
<td>Correctness Tests</td>
<td>118</td>
</tr>
<tr>
<td>B.5</td>
<td>Code to Daemonize a Program</td>
<td>122</td>
</tr>
</tbody>
</table>
B.6  DTD Validation Code ....................................................... 123
B.7  External Export Code .................................................... 127
B.8  GraphML Read/Write Code .............................................. 128
B.9  Graph Node Class Code .................................................. 132
B.10 Exceptions Specific to Multiparent Application ....................... 133
B.11 Main Multiparent Application Code ..................................... 134
B.12 Performance Test Coding ................................................ 141
B.13 Serialization/Deserialization Code ..................................... 144
B.14 Inheritance Correctness Validation Output (see Section 4.2) .......... 145
C.1  Default Configuration File .............................................. 150
C.2  Specific Configuration File ............................................. 151
E.1  GraphML Code for Graph in Figure E.1 ................................. 172
1 INTRODUCTION

In a high-performance computing environment, a large number of computing resources, be they processors, memory, secondary storage, network bandwidth, or simply time, must be allocated among a number of competing resource requests, usually associated with some sort of computing job or task. However, despite the relatively large number of computing resources available, one will still find that even more resources are desired by the users, and would be used if available. Indeed, there is often no practical limit to the amount of computing resources that users of a high-performance system would utilize. However, usually due to budgetary constraints, neither the purchase of new resources on that scale, nor their maintenance costs, is usually feasible. Therefore, it becomes necessary to find some mechanism for allocating resources to organizations and users requesting them, according to some pre-defined concept of fairness.

In a number of instances, the problem of assigning resources has been solved using purely economic principles. As was stated previously, despite the large number of resources available, there are still more computing resources desired than are available. Therefore, a scarcity, as it is defined in economic theory, exists, causing the creation of a potential market. In instances where the use of those resources must be purchased from a service provider, the purchase price may simply be adjusted until the resources available, or the supply, matches the resources desired, or the demand. Admittedly, this means that some individuals or organizations that would have used the computing resources at the lower prices will choose to find alternative means of performing their processing. However, balancing out the supply and demand using these economic principles neatly sidesteps the need for
determining a fairness mechanism for that resource allocation. Quite simply, people have access to resources they are willing and able to pay for.

In several contexts, however, this idea of economic balance to allocate resources is not desirable. For example, at some university high performance computing centers or labs, the computing resources have been provided by outside financial donations, with the stipulation that the resources are available for use by anyone in the university with a legitimate need, without any monetary cost. These stipulations preclude any possibility of using the economic principles described above to eliminate the need for some metric of fairness to be used in allocating resources. Indeed, without charging for the use of the resources, there must, by necessity, exist some set of policies that specify how these resources are to be allocated. Predominantly, this is done on an organizational basis, assigning resources to organizations in particular ratios or fashions, according to the policies of the installation and sponsoring organization.

1.1 Statement of Problem

The task of assigning or scheduling resources based upon some idea of fairness and efficiency is a difficult, but largely solved task, through the use of resource scheduling software such as Moab Workload Manager™ by Adaptive Computing, Inc. (Adaptive Computing, Inc., 2009), or Sun Grid Engine (Sun Microsystems, 2009). Unfortunately, these systems generally do not incorporate any significant internal concept of organizational structure, instead leaving that function to external software or resources, such as a local flat file, or a directory service. This leaves the organizational structure, as seen by the scheduler software, as a series of separate organizations, each completely independent from the others, with their own mutually exclusive set of users, as illustrated in Figure 1.1. As should be clear, while this might be a helpful model in some instances, this does not accurately reflect the organizational structure, including parent-child relationships between those organizations.
The use of directory services, such as those provided by ActiveDirectory™ by Microsoft (Microsoft, inc., 2007), eDirectory™ by Novell (Novell, inc., 2007), or the open-source OpenLDAP project (OpenLDAP, 2009), enhances the concept of organizational structure by allowing the organizations in question to reside in a hierarchical, tree-like structure, creating parent-child relationships with a one-to-many cardinality. For an example of this, see Figure 1.3.

This concept of organizations in a pure tree hierarchy adds additional functionality that the flat model fails to. For example, when a tree structure is present, this defines a distinct parent-child relationship, and it is, in theory, possible to cascade or inherit both schemas and attributes from parent to child, the former allowing for an entity’s structural changes to be inherited, and the latter for arbitrary attribute values. In particular, those attribute values can be used in scheduling algorithms for high-performance systems, allowing usage targets and limitations to be placed on organizations, while automatically having those restrictions cascade down to child organizations.
Figure 1.2: Organizational Structure with Multiparent Organization
Figure 1.3: Example of Hierarchical, Tree-based Organizational Structure
Unfortunately, while this model is certainly an improvement over the flat structure shown earlier, modern organizations are much more complex than can be easily modeled using a simple tree. For example, consider an interdisciplinary research group studying biophysics. This relatively unique organization crosses boundaries between organizations, in this case the Department of Biology and the Department of Physics as seen in Figure 1.2.

This multi-parent structure more accurately models the relationship that this particular organization has with both parent organizations, but presents a problem. None of the many current directory services examined in this research were able to model this structure directly, and, by extension, none includes a mechanism for resolving attribute value inheritance conflicts. To illustrate this conflict resolution problem, consider a situation in which the structure shown in Figure 1.2 was used in a high performance scheduling system, and a limitation of a maximum of $X$ total processors per job was applied to the Physics department organization, and a maximum of $Y$ total processors per job was applied to the Biology department; assuming $X \neq Y$, without an inheritance conflict resolution mechanism, it would be unclear what limitation to cascade down to the Biophysics research group.

This thesis proposes the design of a new management framework for modeling common organizational structures, allowing organizations to be associated with an unlimited number of parent organizations. In addition, by allowing this multi-parent inheritance, this thesis also proposes that inheritance conflict resolution be handled through the use of externally defined policies, allowing installation-specific mechanisms for resolving those inheritance conflicts.

1.2 Use Cases

In order to clarify how the proposed system will apply to specific situations, the following example scenarios are provided to illustrate some of the capabilities of the system.
1.2.1 Use Case 1: Simple Tree Structure

As shown in Figure 1.4, the organizational structure follows a simple tree model, in which each organization has no more than one parent organization. In this situation, a system administrator applies a policy of $PPN = 32$, meaning a per-job policy of 32 processors per node maximum, to the ultimate parent organization, the University. In the case of the College of Life Sciences, and its child organization, the Department of Biology, no overriding policies are set, and therefore these two organizations simply inherit the $PPN = 32$ policy, with no changes; the users $UserI$, $UserJ$, and $UserK$ would experience this policy. The administrator likes the College of Engineering and Technology more, however, and overrides the policy at the college level, setting a policy of $PPN = 64$. With no further override applied, this policy is inherited by the Department of Mechanical Engineering, and to the associated users, $UserA$ and $UserB$. Finally, the administrator overrides the level to $PPN = 48$ for the College of Physical and Mathematical Sciences, giving them some preference, albeit less than that given to the College of Engineering. This value is simply inherited to the Department of Computer Science, and its users, while the administrator overrides the value for the Department of Physics, reducing it back down to $PPN = 32$. 
Figure 1.4: Simple Tree-based Inheritance Using an Override Policy
1.2.2 Use Case 2: Multi-Parent Tree Structure

In order to address the limitations of the scenario discussed in Section 1.2.1, it becomes necessary to describe a more optimal scenario, and thereby illustrate the optimal operation of the proposed solution. Therefore, beginning in Figure 1.5, we see an organizational structure involving multiple layers of inheritance. In this case, the system administrator applies a default policy of $MAXPROC = 32$ to the ultimate parent organization, specifying that any particular user’s jobs may use only up to 32 processors total. In the case of the College of Life Sciences, and its direct child organization, the Department of Biology, this value is simply inherited directly, and would therefore apply to users UserI, UserJ, and UserK. The administrator, however, explicitly overrides the policy for the College of Physical and Mathematical Sciences, specifying $MAXPROC = 128$. This policy is then inherited to the Department of Physics, and therefore to the users UserC, UserD, and UserE. However, the Biophysics Research Group is defined as a child organization of both the Department of Physics, and the Department of Biology. Since this research group could theoretically inherit this attribute from either parent, and the attribute values are different, the inheritance conflict resolution policy comes into play. In this case, the policy specifies that the maximum value is inherited, meaning that the Biophysics group inherits the $MAXPROC = 128$, rather than the lower $MAXPROC = 32$.

The **Keep-Maximum** policy utilized for this use case, and shown in Figure 1.5, is intended as an example of potential inheritance conflict resolution policies. Additional policies might include a **Keep-Average**, or a **Keep-Minimum** policy. Since the inheritance conflict resolution policies are intended to be implemented in an extensible way, the list of potential policy implementations is fundamentally unlimited.
Figure 1.5: Multi-Parent Tree-based Inheritance using a Keep-Maximum Policy
1.3 Research Questions

Through this research, the following questions will be answered:

- Is it possible to model organizations in a way that will enable each organization to have any number of parent organizations?
- Is it possible to define certain attributes that can cascade or inherit down those parent-child relationships?
- What common inheritance conflict resolution policies should exist, and how should they be modeled?
- How can the data be modeled in such a way as to make it easily accessible for use by external services, such as high performance scheduling software?
- How much load will the software put on a server of average capability, and if excessive, how can the software be modified to allow for multiple servers to cooperatively provide the service, splitting the load among them?

1.4 Scope and Limitations

This thesis and associated software development provide several features. The multi-parent organizational structure will be established, with the inheritance of specified attribute values, and conflict resolution policies using a rules engine. In addition, this project will benchmark the system and its processing load, to determine its scalability. However, it will not provide multiple schemas for defining different types of organizations, or include user information. Also, this project will simply provide a mechanism for providing arbitrary attribute data associated with organizations; although that data will almost certainly be utilized in scheduling algorithms, this project itself will not provide those scheduling algorithms.
2 LITERATURE REVIEW

2.1 Introduction

In order to fully understand the nature of the problem described in this document, and its proposed solution, it is necessary to review related subjects. These include discussions of traditional directory services, explaining which attributes are desirable, and the reason that the facilities provided by these systems are inadequate. In addition, the general organization of High-Performance Computing (or HPC) scheduling software will be discussed, including a discussion of how organizational information might be utilized in the priority calculation functionality; the relatively meager organizational structure facilities of these scheduling systems will also be examined. The scheduling and directory service concepts will also be extended into the realm of Grids, where independently-managed computational resources are cooperatively scheduled, and credential information is shared to further that goal. Finally, some basic data structure theory will be provided, including discussions of both graphs and trees, explaining which attributes of each are the most and least suitable for the final solution proposed by this document.

2.2 Directory Services

Directory services provide a uniform method of storing arbitrary information, and retrieving that information. This information need not be limited to any particular category. Indeed, often directory services are used to store user information, including names, organizations the user belongs to, login names, and passwords; as well as information about a corporate organizational structure, and assets owned by that corporation.
While directory services may provide a more generic data storage mechanism, the most useful feature in the context of this document is the ability to store and utilize user information, including organizational membership information.

The vast majority of current directory services base their data model and interactions upon the x.500 series of specifications, albeit with varying levels of completeness. Therefore, a discussion of the standards and their relative usefulness should prove useful.

The x.500 standard (ITU-T, 2005a; Yeong, Howes, and Kille, Yeong et al.) defines a globally unique name space. Each entity within that space therefore is identified by a globally unique name. These names conform to the pattern \( type_1 = value_1, type_2 = value_2, \ldots, type_n = value_n \), where each \( type_n \) refers to a particular class of entity, and each named \( value_n \) represents a path from the root of the directory tree to the entity in question. (Chadwick, 1994, 3.6) Each fully-qualified, globally unique name is referred to as a distinguished name or DN. For example, a particular entity might be defined with the DN of \( uid = \text{username}, ou = \text{department}, o = \text{company} \); note that the order of the entries from least-specific to most-specific, or visa-versa is implementation specific, and may vary depending on context. In this case, the types of \( uid, ou, \) and \( o \) are commonly used to represent the user ID, the organizational unit, and the organization, although any type can be defined.

In this model, a directory consists of a rooted, unordered, undirected tree, although, as is noted in Section 2.5.2, the implementation of the directory structure must often utilize directional elements like memory pointers or references. Nevertheless, with the name space defined above, it becomes a very simple task to determine parent entities of a particular fully-qualified DN. Quite simply, one must simply remove the most specific portion of the DN, and the result is the DN of the parent entity. For example, the parent entity of \( uid = \text{username}, ou = \text{department}, o = \text{company} \) is \( ou = \text{department}, o = \text{company} \). Any number of levels may be used in this tree structure. (ITU-T, 2005a, 9.8)

Within the directory service, each entity is built based on a class definition or schema, a similar concept to that used in object-oriented programming. This schema includes defini-
tions for what attributes can be included in the entry. For example, a person entry will likely include fields for given name, surname, office location, home address, home telephone, work telephone, etc., while an organization entry, for example for a department within a company, may have a telephone, a contact person, etc. Modeling a surname, for example, makes very little sense for an organization.

As in object-oriented programming, the directory structure can define new classes of entities, and inherit schemas from those organizations. For example, one might define a class to represent persons, and create a subclass to represent a specific type of person. Perhaps a particular department wants to track information for their employees that is not included in the main schema. The subclass can even inherit from multiple parent classes, similar to the inheritance model of C++ classes. (Chadwick, 1994, 3.7)

At first glance, the inclusion of this versatile inheritance model would seem to fit the definition of the structure this document is attempting to define. However, it is extremely important to understand that this inheritance model only inherits schemas between classes, not values between related objects of a particular class. Indeed, while some of the concepts are related, the structure described in Chapter 1 and illustrated in Figure 1.2 defines a set of related entities, or objects, each having the same schema. The inheritance model proposed in this document does not represent the inheritance of the possibility of a particular attribute, but rather the inheritance of the value associated with a particular attribute.

In addition, the directory service definition requires that each entity have a single distinguished name. (ITU-T, 2005a, 6) Aliases are allowed so that a particular entity can appear to reside in multiple locations in the tree. However, similar to attempting to organize a file system directory structure using symbolic links or shortcuts, this does not easily enable an entity to have multiple parents. (ITU-T, 2005b, 9.8) A particular entity may reside in a single location in the tree, and be aliased to appear to reside in other locations in the tree, but this structure quickly becomes very unwieldy, and difficult to maintain. Unfortunately, aliases are generally implemented as a reference that points toward the original entity, and
the original entity is often not aware of the list of aliases that point to it. The aliasing operation allows a particular entity or alias to know where that particular single parent entity is, by the method described above, but this does not provide a method of traversing from a single entity to more than one parent. In this way, they closely resemble symbolic links in a POSIX file system.

Since the solution this document proposes involves the modeling of organizational structures, it therefore becomes important to understand current solutions for modeling that structure. Directory services make up a large portion of this solution set, and as a result their capabilities must be considered. As described, the directory service model, while clearly providing a hierarchical, tree-like structure, does not provide sufficient versatility for the multi-parent inheritance model described. Additionally, the inheritance provided by the \textit{x.500} specification, upon which directory services are based, only provides for schema inheritance, and not inheritance of the values themselves. Finally the directory services do not include any inheritance conflict resolution mechanism, although this is understandable in light of the lack of multi-parent inheritance which would necessitate this feature.

2.3 HPC Scheduling

In high-performance computing, or HPC, scheduling involves the task of allocating resources to computational tasks. These resources may be anything that is schedulable in this context. Common resources include number of processors, speed of processors, amount of RAM, Network Speed, Network interfaces (eg. Infiniband vs. Ethernet), storage capacity, and even floating or node-locked software licenses. Most major scheduling software also includes the ability to incorporate arbitrary attributes into the list of resources. In addition, the scheduling software can utilize a large amount of information in making the decisions where to allocate those resources.
2.3.1 Priority Calculations

The most basic form of scheduling in any computing system, including high-performance systems, would be a simple FIFO queue. Using this model, the scheduling software simply evaluates periodically to find unallocated resources. If it finds available resources, it considers the first process or job waiting in the queue. If that job will consume resources up to, but not exceeding, the set of available resources, that job is started. (Stallings, 2005, 402) Whether or not the job is started, at this point the scheduling iteration is complete. Some enhancements to this model include considering additional jobs after a job is started, and allowing more jobs to start immediately, rather than limiting the scheduler to a maximum of one job starting per iteration. However, while refinements of this type do improve the efficiency of the scheduler, they do not fundamentally change the “First-in, First-out” nature of this scheduling approach.

The efficiency, in terms of resources used, of a simple FIFO scheduling queue is difficult to beat. However, most high performance computing facilities prefer to implement policies that improve the *fairness* of the resource allocation, even if it means the loss of some of the efficiency mentioned. Most schedulers use a priority mechanism to order the jobs in the queue based upon some criteria. Once this ordering is established, the scheduler may simply continue to consider jobs in the order they reside in the queue. One could even trivially implement the FIFO queue inside the prioritization mechanism by eliminating all the priority calculation factors other than those that increase priority directly in proportion to time spent in the queue. Outside that possibility, however, most high performance computing facilities choose to include other factors in that priority calculation, including, but not limited to, the factors described below. It is worth noting, however, that these priority adjustments only affect the order that jobs are considered to be run. If the list of eligible jobs is empty, or cycles too quickly, these priority adjustments may become ineffectual.

As an example, in the Moab™scheduler, the priority for a job can be calculated using an equation like that shown in Equation 2.1, where $f_n$ is the feature you’re calculating
priority based on, and $k_n$ is a constant used to apply a different weight to that factor: (Adaptive Computing, Inc., 2009, 5.1)

$$priority = \sum_{i=1}^{n} k_n f_n$$  \hspace{1cm} (2.1)

While scheduling software can utilize a great many factors in calculating the priority, and therefore execution order, of a series of processing jobs, a certain subset of these are the most common, and are outlined here:

**Historical Usage**

One of the most common factors considered in calculating priority of a job within a queue revolves about historical usage patterns. For example, in the Moab\textsuperscript{TM} scheduler, this facility is termed *Fairshare*. (Adaptive Computing, Inc., 2009, 6.3) Using this model, priority adjustments can be based upon how much of a particular resource a user, a group, etc., has used historically. This is most often used to increase priority for casual users, and decrease priority for power users. This way, if a user has historically been a large user of the system, another user will get a higher priority on his or her jobs than the user will on his. This continues until the utilization patterns that caused the priority adjustments, age beyond the time window of statistics considered for this calculation. Overall, this facility is used to provide a balancing effect, allowing as many users as possible to utilize the system. Technically one would simply need to multiply the priority calculation weight, $k_n$ in Equation 2.1, which corresponds directly to the user’s historical usage, by some number less than 0, to cause the scheduler to *reward* rather than punish those users with significant historical usage. However, this policy is rarely seen, as many facilities have an expressly stated goal of equality that this approach would undermine.
Job Resource Request

Often, the list of resources that a particular job requests is also utilized in priority calculations. For example, if an HPC facility wants to encourage fewer jobs with lower processor requirements, and more jobs with higher processor requirements, all that is required is to adjust the priority calculation equation to include the processor count, multiplied by some factor greater than 1. Similarly, a facility could conceivably adjust priority calculation based on number of nodes requested, amount of memory requested, amount of time, etc. In essence, if the scheduler can consider something a resource, and can therefore schedule based on it, this can be used to calculate job priorities. (Adaptive Computing, Inc., 2009, 5.1.2.3)

Service Levels

An additional method often used to adjust priority of a job in HPC scheduling, is based on the idea of the level or quality of service provided by the scheduler. For example, often a policy is implemented that increases priority based on the total queue time, or the amount of time a job has been waiting in the queue. By multiplying the amount of time spent waiting by some positive value, the scheduler may guarantee that, all other things being equal, jobs that have waited the longest will be the first scheduled. (Adaptive Computing, Inc., 2009, 5.1.2.4.1)

Credentials

Finally, scheduling may be accomplished based upon some credential of the job, or some attribute of the job’s owning user or group. For example, an administrator may explicitly state that an individual gets a priority boost on all his or her jobs. Alternatively, a specific research group may have their priority reduced by some specific amount.
2.4 Grids

Computational grids consist of an aggregation of disparate resources, usually independently owned and managed, for the purpose of resource and information sharing. (Foster et al., 2001, 2) Often a grid is described as a “cluster of clusters”. More generically in a grid infrastructure, independent entities provide available computational resources to a pool, and tasks may request access to those resources. These resources may be directly computational systems, including computational clusters, but, as described in Section 2.3, many other items may be schedulable as resources, including network bandwidth, software licenses, or storage system capacity.

Grids form a peer-to-peer relationship between participating institutions. Each entity attached to the grid may be either a resource provider, a resource consumer, or, as is most often the case, both. Protocols, including the GRIP, or Grid Resource Information Protocol (Foster et al., 2001, 4.3) from the Globus Toolkit, are utilized to discover resources, and communicate that information across the intra-grid communication media. Additional protocols, including the Grid Resource Access and Management, or GRAM (Foster, 2005), also included with the Globus Toolkit, provide mechanisms for requesting allocation of remote grid-member resources for a particular task, and launching the remote processes appropriately.

2.4.1 Virtual Organizations

The majority of grid based architectures orient themselves on the idea of a virtual organization, or VO. Within this context, a Virtual Organization, “is a group of participants who seek to share resources for some common purpose.” (Dumitrescu and Foster, 2004, 1). This organization represents the aggregation of computational resources (resource providers or producers) and users of those resources (or resource consumers) across disparate organizations. Indeed, the concept of a VO is core to the definition of a grid, as it is the fundamental unit of inter-organization collaboration. Each entity, whether producer, consumer, or both,
belongs to one or more VOs, in a model very similar to group membership credentials, as is often used in POSIX-like file systems or process tables. Each resource producer participating in a VO provides a set of resources to the VO for use, according to local usage policies. These policies might include some desired or strictly enforced ratio of local use vs. remote use, for example. In turn, consumers participating in the VO may consume those resources with scheduled computing tasks. Depending on the nature of the sharing relationships and the nature of the grid toolkits providing those relationships, each task may be assigned to a specific VO, and may only consume the resources from that VO, or may utilize all the resources available to the consumer, as provided by the consumer’s VO relationships.

To further complicate matters, the VOs involved in a particular grid may, in and of themselves, have some form of inter-VO structure, which may be modeled in any number of ways, similar to those methods and models described in Section 2.5. Of particular interest is the proposed hierarchical VOs proposed as a method of organizing projects and sub-projects. (Kim and Buyya, 2006) In this model, as shown in Figure 2.1, the individual VOs have designated parent VOs, and are arranged in a strictly tree-oriented, hierarchical fashion, similar to that shown in Figure 1.3. Indeed, the only major difference between the structures described in Figures 2.1 and 1.3 are simply the nature of the organizations in question; in one, the organizations are specifically virtual, while the other simply models a more generic organization. In either case, an organization may have a maximum of one parent. In addition, the organizational model described has a relatively limited inheritance model. Each supplier of computational resources supplies a particular set of resources to a particular VO, and that VO’s children, by definition, inherit access to that entire resource set. For example, as shown in Figure 2.1, resource provider \( R1 \) provides 50% of its resources, according to some metric, to Virtual Organization \( VO-A \). As a result, user \( U1 \) has access to that resource, since \( U1 \) is a direct member of \( VO-A \). However, according to the proposed inheritance model, users \( U2, U3, U4, \) and \( U5 \) also have full access to that resource, since they are direct members of a descendant VO of \( VO-A \). In essence, within this model, the VOs themselves have no
mechanism for adjusting or reducing the set of resources available to their descendants. In Figure 2.1, VO-A cannot say, for example, that 30% of the resources it inherited from R1, or 15% of R1’s total, may be accessed by VO-A1 and its descendants, and 70% of the inherited resources, or 35% of R1’s total, may be accessed by VO-A2.

2.5 Data Structures

The idea of structuring data in some fashion is a fundamental concept in software design. Often multiple courses are taught on the subject when one is pursuing a degree in Computer Science or Software Engineering, making it impossible to discuss the subject in its entirety, within this context. Additionally, much of the discussion would not be relevant here, for example, data structures like linked-lists, hash tables, and maps, which are useful in general, but are not relevant here. Instead, this document will be discuss only the data structures commonly used to organize information about organizations or user credential information, or provide an integral part of the solution discussed in this document.

2.5.1 Graph

In mathematical theory, a graph is “a pair \((V, E)\), where \(V\) is a finite set and \(E\) is a binary relation on \(V\)”. (Cormen et al., 1990, 86) Each element of \(V\), called a vertex, usually is taken to represent an entity, while the elements of \(E\), traditionally referred to as edges, model the relationship between them. Graphs can be easily categorized into two major styles, depending on whether the edges in \(E\) contain any form of directionality. These types are called undirected and directed graphs.

Structure and Representation

An undirected graph contains edges in \(E\) that contain no direction information. As a result, the relationships between the entities is commutative, meaning that, for example in Figure 2.5.1, \(A\) is related to \(B\) in the exact same way \(B\) is related to \(A\). Similarly, the relationships
Figure 2.1: Examples of Hierarchical VOs (Kim and Buyya, 2006)
between the entities modeled by the graph have no directionality either. For example, a peer-to-peer computer network, where each member of the network is an equal peer of the others, follows the model of an undirected graph. An example of this network type, a grid, is discussed in Section 2.4.

Directed graphs include some form of direction information associated with each edge in $E$. Effectively, this means that one can assign some sort of role with respect to the entities in the relationship. For example, one might model the flow of information in this fashion. In the example diagram in Figure 2.5.1, taken as such an information flow diagram, node $G$ receives information from nodes $I$ and $E$, and sends information to node $H$.

The structure of a graph, especially a directed graph, is sufficiently general that it becomes useful in designing the data structure defined in this document. The directionality
information provided by the edges in the directional graph can be utilized to define the
roles of the entities in the relationship - parents and children, for example. However, while
the structure being proposed in this document is technically a directional graph, it contains
additional restrictions that are not necessarily excluded from the more general directed
graph model. Both directed and undirected graphs may contain cycles. In either graph
model, directed or undirected, a cycle occurs when paths, or series of nodes connected by
dges, exist through the graph that allows one to return to a node that one already traversed.
(Cormen et al., 1990, 88) For example, in Figures 2.5.1 and 2.5.1, we see this phenomenon.
In Figure 2.5.1, we see cycles made up of the following paths:

- \((A, B, F, C, A)\)
- \((A, D, F, C, A)\)
- \((D, F, E, D)\)
- etc.

In the directed graph example, seen in Figure 2.5.1, we similarly see cycles made up of
the following paths:

- \((A, C, F, D, A)\)
- \((D, E, F, D)\)
- etc.

Notice that in Figure 2.5.1 that the path \((A, C, F, B, A)\) does not form a cycle, since
two of those edges, \((B, F)\) and \((A, B)\) can only be traversed in the opposite direction. This
feature of having multiple paths between points, for example from \(A\) to \(F\) is a desirable at-
ttribute. Indeed, having multiple node genealogies is in essence the innovation being proposed
in this document.

In addition, the definition of a graph allows for multiple disparate sections, with
no edges connecting the sections. In this situation, a graph is said to be disconnected.
Conversely, if for every two vertices in the graph, there exists some path between them, the graph is said to be **connected**. While it is theoretically possible to manage an organizational structure that contains disconnected graphs, for simplicity’s sake, this document will operate under the assumption that it is only necessary to maintain a single structure, implying a connected graph.

Therefore, as one examines the example structure shown in Figure 1.2, one sees quickly that, while the structure proposed in this document is certainly a graph, it becomes necessary to further define it as a directed, acyclic (without cycles), connected graph. In this way, while one can use the graph data model to store the structure proposed, only a certain class of graphs will be considered valid structures in this context.

### 2.5.2 Tree

Trees may be defined as a special case of a graph. In general, a tree is defined as a “connected, acyclic, undirected graph.” (Cormen et al., 1990, 91) In practice, however, the data structures used to store trees often use something that more closely resembles a *directed*, connected, acyclic graph. Quite simply, the data models used to store these structures in memory are often designed around the use of memory references or pointers. In effect, these pointers are placed in one location, and point to another, like the edges of a directed graph. In most situations, this is not a significant detriment, since trees are often traversed in only one direction. In the cases where the edges need to be traversed bi-directionally, pairs of pointers are often used, one pointing in each direction.

When trees contain a particular attribute, for example a static branching factor, they are often classified based on that attribute. For example, one of the more useful tree structures is the *binary* tree, as seen in Figure 2.5.2. In this case, the branching factor is 2, meaning that each node has exactly two children. Obviously, in some cases, the node’s children are not fully populated, but nevertheless, each may be defined as the *left* or the *right* child of the current node.
While not strictly a part of the definition, many useful trees also are ordered and rooted. To be ordered, the children vertices of a particular node must have some order among themselves. For example, in Figure 2.5.2, if in some fashion $B$ comes before $C$, and $C$ comes before $D$, then this must be an ordered tree. If the order among points $B$, $C$, and $D$ is nonexistent, we say the tree is unordered. To be rooted, a tree must have one node distinguished from the others as the ultimate parent. In Figures 2.5.2 and 2.5.2, for example, the $A$ node is considered to be the tree’s root. Note that these attributes aren’t used for the structure requirements outlined in this document, but are nevertheless useful for general discussion.

When comparing the proposed data structure, as seen in Figure 1.2, with what we know about tree structures, we see a structure that comes very close to a tree. The only elements of the proposed data structure that seem to either violate the tree design, or are simply missing from that design, are the directionality of the edges, and the presence of multiple paths leading into a particular node. Since a tree is technically undirected, and
acyclic, in order to design the proposed data structure, one must describe the concept of a directed tree, or a tree in which the relationships between nodes have directionality. There is, in essence, the same relationship between undirected and directed graphs, as described in Section 2.5.1, as there is between the strict, undirected definition of a tree, and our proposed modification to make it directed. Also the addition of directionality information, especially in a rooted tree, allows the addition of multiple paths to a particular node, without violating the acyclic nature of the tree.

A tree may also be recursively defined as either an empty tree, or a root node followed by a set of child trees. Each of these child trees may then be either an empty tree, or a node followed by a set of child trees, etc. This definition implies that at each level the subtrees are completely separate, and no cross-linking is allowed. According to this definition, the multi-path structure shown in Figure 1.2 is not a tree, and therefore can only be modeled using the more generic graph data model. It is anticipated that, due to the ease of implementing this definition of a tree, when compared to the earlier adaptation which allowed multiple paths to a particular vertex, that most tree data structure libraries to date will not support the structure in question, and a graph data structure library is necessary.

2.6 Conclusion

As is often the case, the design of a framework or application for use in high-performance computing, is certainly not a trivial task. A great deal of planning and background understanding is necessary in order to properly design the framework in question to perform optimally. Therefore, it is critical that one have a significant understanding of the existing credential management systems, including the x.500 series of specifications, and the associated directory systems. It is important to understand both the capabilities and shortcomings of the exiting scheduling systems, including both their scheduling priority algorithms, and the included credential mechanisms. A strong understanding of the relevant data structures, including trees and graphs, provides a significant boon during the design process for such
a framework. Each of these provides a new aspect of understanding, that allows the implementer to design and build that framework in an optimal way, utilizing an appropriate environment, etc. Therefore, each of these is essential for the full understanding of the design, as proposed in Chapter 3.

In addition, the current solutions to modeling multi-parent organizational structures, including both x.500-based directory services, and current software scheduling systems, fail to meet the needs of the proposed organizational model. Many of these fail to provide any mechanism of defining relationships between organizations at all. Others allow relationships to be described, but either only allow peer-to-peer relationships, as in the grids discussed in Section 2.4, or only allow single-parent relationships, as in the case of the directory services discussed in Section 2.2. In the cases where inheritance is allowed, only schema inheritance is provided for, rather than the attribute inheritance which this research proposes. Therefore, existing solutions are inadequate for the proposed data model and associated inheritance algorithm.
3 METHODOLOGY

As with all software development processes, the system described in the preceding chapters produces certain requirements for the developer and development environment. Numerous options exist for the development language, overall data model, operating system environment, etc. The nature of the system described in the preceding chapters predetermines some of these details, while others are left to the discretion of the researcher. Even in these discretionary areas, however, certain options provide additional benefits that must be considered. In essence, making the most appropriate decisions provides an optimized development environment, greatly reducing the development time, and reducing the probability of development errors.

3.1 System Overview

In order to manage periodic tasks, update the in-memory structure, etc., the software will be designed as a daemon-style software component which will stay resident in the system, allowing it to respond to changes in the specified structure and perform other maintenance in the background. The information used to model the organizational structure in question will be stored in a file on the server’s local file system, using the format specified in Section 3.2.3 and Appendix E. It is also anticipated that administrators of the system will periodically change the structure and configuration options, and as a result, the software will periodically re-read the specified input and configuration files, and make appropriate changes, if any, to the in-memory structure; this task in particular can be done in a continuous fashion, benefiting from the daemon-style software design. In this way, the administrator implementing
the tool described can easily make changes, and have confidence that both the structure, and configuration information, will be correctly re-evaluated in short order. The software will also include a mechanism for exporting the overall structure, including both inherited and assigned values, for use in serialization as discussed in Section 3.2.3, or for integration into external services, as discussed in Section 3.5.2.

3.2 Data Model

3.2.1 Data Structure

Modeling the relationship between organizations will be accomplished by modeling the organizations as objects in the NetworkX\textsuperscript{1} object-oriented graph library. This library allows for differentiation between parent and child organizations through the medium of directional graph edges. Since the library provides mechanisms for traversing the parent-to-child link with equal facility in either direction, the data will be modeled using the traditional parent-pointing-to-child orientation.

For the purposes of this tool, the organizational structure must:\textsuperscript{2}

- be connected
- be directional
- be acyclic

3.2.2 Data Storage

In order to minimize the data losses in the event of a software or hardware failure, the software described will periodically store the structure’s state information to a storage medium, presumably a local hard drive. This periodic checkpoint will provide an administrator a

\textsuperscript{1}See http://networkx.lanl.gov
\textsuperscript{2}For more information on graph terminology, and the reasoning behind these requirements, see Section 2.5.2
consistent state to use in re-starting the software in the event of a failure, in essence check-pointing the critical contents of memory. The amount of time between checkpoints will be configurable via a configuration file. That configuration parameter will provide a coarse time value, in the sense that the checkpoint task will be initiated based on the configured interval. However, because other processing tasks will exist, this may delay the initialization of the checkpoint task.

3.2.3 Data Serialization

While the software described will maintain a memory state that represents the set of organizational relationships, it is necessary to pre-load a configuration set from disk storage as the software is launched, and to store that information to disk periodically, as described above in Section 3.2.2. Additionally, the software will periodically re-read the specified organizational model and configuration files, updating the in-memory structures as necessary, to react to changes by the system administrator. The software will be designed to utilize the GraphML standard format for defining graphs. Serialization and import routines and tools will be provided for this format, and the path to the structure file will be configurable via a configuration file. All configuration file syntax is documented in Appendix C. The GraphML syntax is documented in Appendix E.

In order to illustrate the translation between the in-memory graph and the data serialization format, see Figure 3.1 and Code Segment 3.1. For this XML-based GraphML serialization, the data will be validated using a provided XML Document Type Definition, as shown in Appendix B in Section B.1.1. Upon import, the graph information will also be analyzed to verify that it meets the requirements shown in 3.2.1
Code Segment 3.1: GraphML Code for Graph in Figure 3.1

```xml
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="organization_name" edgedefault="directed">
    <node id="A">
      <data key="attr">val2</data>
      <data key="attr2">val10</data>
    </node>
    <node id="B">
      <data key="attr">val3</data>
    </node>
    <node id="C" />
    <node id="D">
      <data key="attr">val1</data>
    </node>
    <node id="E">
      <data key="attr">val4</data>
    </node>
    <node id="F" />
    <node id="G" />
    <node id="H" />
    <node id="I" />
    <edge source="A" target="B" />
    <edge source="A" target="C" />
    <edge source="B" target="F" />
    <edge source="C" target="F" />
    <edge source="D" target="A" />
    <edge source="D" target="E" />
    <edge source="E" target="F" />
    <edge source="E" target="G" />
    <edge source="F" target="D" />
    <edge source="G" target="H" />
    <edge source="I" target="G" />
  </graph>
</graphml>
```
3.3 Tools

As it is anticipated that this research will not re-implement standard graph algorithms and operations, it is necessary that the programming environment provide corresponding libraries, as is often the case with languages utilizing an object-oriented environment. Additionally, in order to allow installation-specific conflict resolution policies (see Section 3.5.1) and data export routines (see Section 3.5.2), the ability to both load and execute external programming code is absolutely crucial. Although there is some concern for the overall com-
putational speed, while executing in most implementations, the organizational structure will change only rarely. As a result, as long as a reasonable convergence or propagation time is assured (as discussed in Sections 3.6.2 and 4.3), then the software’s performance matters very little, and the question of whether the language is compiled or interpreted is largely irrelevant. Additionally the programming language and environment should be easy to learn and understand, allowing for other extensions and modifications to the software by future developers.

Based on these criteria, the Python programming language will be utilized. This interpreted language, despite some very unique syntax rules, is widely used in industry, and has a full-featured set of libraries. Because of the ubiquity, an individual installation administrator should have very little difficulty finding documentation in building the conflict resolution policies discussed in Section 3.5.1, presuming that the provided examples are insufficient for either filling the needs, or at least serving as examples for a new resolution function. Additionally, should the implementer choose to extend the software functionality, the relative ease of finding documentation for such a common language will also be to the implementer’s advantage.

3.4 Software Licensing

In order to ensure the maximum applicability of the software tools associated with this document, all software utilized will follow an open-source model, and will be available free of charge by the appropriate licenses. The specific licenses that are applied to each software tool and the text of those licenses are found in Appendix A. Unless otherwise noted, software developed in association with this document is licensed according to the GNU General Production License version 2.0, as found in Appendix A, in Section A.2.1.
3.5 Software Design

3.5.1 Periodic Operations

As the software launches, it must retrieve and parse the information contained in the specified configuration file (syntax specified in Appendix C), and the configured graph model file which represents the organizational structure. This data model file will also be read periodically, with the interval to wait between reads a configurable value specified in that configuration file, and the appropriate changes applied. When structure updates occur, the software will perform both a structure validation, verifying that the structure still complies with the requirements found in Section 3.2.1, and a complete graph inheritance traversal, to provide a consistent state for the in-memory data which will reflect the changes in the file. This way, the data will be available immediately for use by external tools, through the use of the specified export functionality.

Processing and Traversal

As referenced in Section 3.2.1, the graph will be traversed from child to parent, rather than the other way around. The algorithm will be recursive, and will maintain a flag on a per-node basis to note whether or not this node has already been traversed for this iteration. As a result, each node within the graph will have that flag cleared at the beginning of the traversal.\footnote{This clearing operation will be accomplished iteratively for simplicity. While there are certainly more efficient methods of doing this, it is anticipated that the number of nodes will be relatively low, and this won’t matter sufficiently to implement a more exotic algorithm.} Each node’s traversal will follow the overall recursive flow found in Figure 3.2.

Data Checkpointing and Storage

In order to mitigate any losses due to software crashes, the software will periodically save a serialized copy of its memory state. Like the periodic re-reading of configuration and model files, the interval for this periodic save will be configurable via the configuration file.
Figure 3.2: Overall Recursive Flow for Graph Traversal
described in Appendix C. The output format will be the same as specified for the input file, as specified in the configuration file. An additional configuration parameter will be utilized to control whether or not the calculated inheritable values will be serialized as well as the pre-specified values. The checkpoint file will be usable as a new input graph structure file, or “model file”, although if the inherited values are serialized, the inheritance behavior will be affected, as they will appear to be statically assigned. Without serializing the calculated values, the output of the checkpoint file should be the same as the corresponding input file, barring some negligible formatting and whitespace differences.

Integration of Conflict-resolution Rules

The system will utilize an extension mechanism to allow individual site implementers to create their own rules for resolving conflicts between the values for a specific attribute inherited from multiple parent organizations. This will be accomplished by utilizing a function or method in the programming environment as a loadable module. The function will need to conform to a specific prototype, documented in Section 4.5.7 on page 80. This function will be called each time an organization inherits conflicting values for a particular inherited attribute. The function will be provided copies of all organization names and corresponding values for the attribute in question for the organization in question and its parent organizations. This function will then need to evaluate the situation based upon its internal rules, and return a value for the attribute that will be assigned to the target organization. Because it is extremely difficult to anticipate all possible situations which require inheritance conflict resolution, allowing the individual administrator to implement separate rules will provide necessary flexibility.

In order to shorten the implementation time for the administrator, and to evaluate the feasibility and scalability of common situations, a number of examples of the conflict resolution policy will be provided, including the following:
3.5.2 Integration with External Systems

A software system as described here does very little good unless it provides a mechanism for integration with external tools. While it is theoretically possible to integrate these external systems with an external tool by parsing the serialized checkpoint file\(^4\), this presumes that the external tool understands the corresponding serialization format, and that the software is configured to serialize the inherited and calculated values. This would require a number of additional steps to get the data into an external tool’s native format, and is certainly not the best use of computational resources. Therefore, the software will also provide a mechanism for exporting the graph’s nodes, and both statically-assigned and inherited values, into other file formats. Similar to the mechanism for integrating external conflict resolution rules, this will take the form of a method or function call, which will be called once per graph node every time the graph is updated, providing it with the graph node’s name and associated attributes. In this way, the implementer can provide the data to external systems in whatever format is appropriate. Examples of code used to export the data into common output formats will also be provided.

3.6 Testing

3.6.1 Compliance and Correctness Testing

Graph Correctness

As discussed in Section 3.2.1, not all graphs will be appropriate for use in this project. Therefore, the software in question will provide a method of validating the structure accord-

---

\(^4\)This serialization file format is described in Section 3.2.3.
ing to both the generic graph definition, but with the additional criteria of being connected, directional, and acyclic. Therefore, in the context of validation testing, several test cases representing graphs both with and without the specified criteria will be tested to determine the software’s ability to handle these issues. For correctness, the software will be required to reject each of the examples with a criteria problem, identifying the criteria that fails the validation tests. The software will correctly accept the valid graphs that do not exhibit any of these additional criteria.

The following test cases will be utilized to validate the graph structure checking methods of the software described.

**Simple, Valid Case** The organizational structure shown in Figure 3.3 represents a medium-complexity organizational structure, including multiple inheritance, which does not violate any of the constraints discussed in Section 3.2.1. This represents a valid structure, and no errors should be generated when this structure is validated.

**Valid, Nonrooted Case** The organizational structure shown in Figure 3.4 represents a more complex, valid organizational structure, which does not start with a single root node.
As this still represents a valid organizational structure, the software should generate no errors for it.

**Disconnected Case**  The organizational structure shown in Figure 3.5 represents an invalid organizational structure, since it involves two completely disconnected graphs. When this is validated, the software should provide an appropriate error message.

**Cyclic Cases**  The organizational structure shown in Figure 3.6.1 represents an invalid organizational structure, since it involves a simple cycle between nodes $B$ and $C$. Similarly, the structure found in Figure 3.6.1 represents an invalid cycle involving nodes $C$, $D$, and $G$. 
In either of these cases, the software should discover the cycle, and return an appropriate error message.

**Inheritance Correctness**

In order to demonstrate the correctness of the inheritance model, a scenario which includes the following attributes will be demonstrated and documented with the corresponding results.

The following elements are critical for testing the inheritance correctness:

- Simple single-parent inheritance from parent to child
- Inheritance from two or more parents each with different values for the same attribute, with a *keep-average* conflict resolution policy
- Single-parent inheritance across multiple generations
- Multi-parent inheritance across multiple generations, with a *keep-average* conflict resolution policy
In order to test these requirements, the structure shown in Figure 3.7 was designed, and represents a valid structure that demonstrates inheritance principles, and assumes a keep-average policy. The value for a particular node is represented by the statically assigned value, if assigned to that particular node; the inherited value, if the node has only one parent; or the average of inherited values, if the node has more than one parent. The expected values for each node, as well as the values’ status as either inherited or statically-assigned, is shown in Table 3.1. Of particular interest are nodes B and H, where the value inherited is overwritten by the statically-assigned value, and nodes F and L, where the node has multiple parents, and is assigned the average of the parent values as its value.
Figure 3.7: Inheritance Test Case Graph Assuming a Keep-Average policy
Table 3.1: Statically Assigned and Inherited Values for Graph in Figure 3.7

<table>
<thead>
<tr>
<th>Node</th>
<th>Value</th>
<th>Static or Inherited</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>Static</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>Static</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>Inherited</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>Inherited</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>Inherited</td>
</tr>
<tr>
<td>F</td>
<td>1.5</td>
<td>Inherited</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>Inherited</td>
</tr>
<tr>
<td>H</td>
<td>3</td>
<td>Static</td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>Inherited</td>
</tr>
<tr>
<td>J</td>
<td>2</td>
<td>Inherited</td>
</tr>
<tr>
<td>K</td>
<td>2</td>
<td>Inherited</td>
</tr>
<tr>
<td>L</td>
<td>1.83</td>
<td>Inherited</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
<td>Inherited</td>
</tr>
<tr>
<td>N</td>
<td>3</td>
<td>Inherited</td>
</tr>
</tbody>
</table>

3.6.2 Performance Testing

Testing Necessity

While the software associated with this project will be targeted for use in high performance computing, this software in and of itself does not have a significant need to be a high-performance application. Therefore, only one aspect of the performance is of significant concern. Specifically, it will be desirable that the graph traversal will be accomplished very quickly, in some very small fraction of a second, so that the delay in re-converging the graph will not be noticeable. Therefore, the test cases found below will be used to demonstrate the convergence of the graph utility in a time reasonably below the human timescale.

Test Description

In order to demonstrate convergence time of the graph processing algorithm, uniform trees of varying depths and branching factors will be generated, and the inheritance values will be propagated throughout. Since the inheritance conflict resolution function will be imple-
mented in each installation, and the processing time will vary, that timing component will be eliminated in this test by testing regular trees without any multiple-inheritance. This will therefore become a test of the software’s ability to do graph traversal utilizing the algorithm discussed in Section 3.5.1. A chart will be provided that shows the average processing time related to the number of nodes of the tree in question.

### 3.6.3 Value Demonstration

As an illustration of the value of the research, the completed software will be used to demonstrate the effect on scheduling algorithms. In particular, the Moab Workload Manager™ by Adaptive Computing, Inc. will be used to simulate the scheduling of a small cluster of computational nodes. Three scenarios will be simulated, and will be analyzed to show several metrics describing the usefulness of the system, showing the results both with and without the weighted adjustments that the system would provide. The metrics provided will include the following:

- Total running time of the simulation
- Total amount of processor-seconds remaining for queued jobs per user, graphed against simulation time
- Total amount of running jobs per user, graphed against simulation time

Each scenario was simulated with a 30 second scheduling interval, with appropriate statistics gathered each iteration. The simulations will be run until the last queued job has completed running. The total running time will be considered to be from the time the first job begins execution, to the time that the last job has completed.

### Resources Simulated

The simulation requires the creation of a “resource trace,” (Adaptive Computing, Inc., 2009, 16.3.2) which describes the resources under the scheduling authority of the software in ques-
tion. As all the scenarios below will utilize the inherited values, or lack thereof, as a simple weight used in the calculation of the job’s priority, the only resources that it becomes critical to simulate are number of CPUs, and time. Time is inherent in the simulation mode of the software. For the first two examples, which are contrived to clearly show the effects of the resource allocation, a resource trace will be provided to simulate 150 computational nodes, each with 4 CPUs, for a grand total of 600 CPUs that can be scheduled. For the third scenario, adapted from real-world data, it becomes necessary to use nodes with 8 CPUs each, and limit the nodes to 25, for a grand total of 200 processors. This is necessary in order to make certain that the total processor count is below the number requested in the real-world simulation, since if this were not the case, the priority calculation where this calculation is integrated would be moot, as all the jobs would be started as soon as possible.

**Common Simulation Information**

- **User/Organizational Information**
  - 3 users will be simulated (called *user1*, *user2*, and *user3*)
  - Each user will be assigned to a different organization (called *org1*, *org2*, and *org3*)
  - *user1* will be assigned to organization *org1*; *user2* will be assigned to *org2*; *user3* will be assigned to *org3*.

- **Simulation/Scenario Information**
  - Each simulation will be run twice, designated as the A and B simulations. For example, there will be simulations 1A, and 1B, which will use the same workload and resource trace information. The only difference between corresponding simulations will be a differing priority calculation weight, as follows:
  - The A simulations will utilize the following weights:
    * *org1* - 10
    * *org2* - 10

48
The B simulations will utilize the following weights:

- org1 - 10
- org2 - 15
- org3 - 20

- These weights are meant to simulate a situation in which, for lack of multiple inheritance, the A simulations only allow org2 to inherit from org1, while the B simulations allow org2 to inherit from both org1 and org3, applying a keep-average conflict resolution policy, as shown in Figure 3.8.

**Simulation Scenario 1**

- Each user will submit 1000 1-processor jobs requesting 1 hour of runtime each, for a total of 3000 processor-hours requested
- Each job will be executed based on the scheduling algorithm, and will be run for the full 1 hour requested
- Jobs will all be queued at the beginning of the simulation

**Simulation Scenario 2**

The jobs submitted in this scenario will be submitted as shown below, and will run the full time requested. Each user will submit jobs in the following profile:

- **User1:**
  - 1000 1-hour, 1-processor jobs, all submitted at the beginning

- **User2:**
  - 500 1-hour, 1-processor jobs, submitted at the beginning
  - 500 1-hour, 1-processor jobs, submitted 2 hours from the beginning
• User3:
  
  - 250 1-hour, 2-processor jobs at the beginning
  - 250 1-hour, 2-processor jobs, submitted about 3 hours from the beginning

Simulation Scenario 3

This scenario is modified from a real-world resource trace taken from one day of the scheduling system at the Fulton Supercomputing Lab at Brigham Young University. The user information was anonymized, and jobs were randomly assigned among the three simulated users. The specific details about this simulation is found in Appendix D.
4 RESULTS

A sample implementation of a multiple inheritance provisioning system has been developed to demonstrate the validity of this thesis. As discussed in Section 3.6, a series of compliance tests must be passed in order to complete the validation. Each of these tests is presented here, with corresponding result information.

4.1 Correctness Testing

Although multiple approaches are possible for constructing arbitrary graphs for each test case, the graphs utilized in this section were constructed programmatically, utilizing the NetworkX library. For this to be accomplished, the library requires that the appropriate program objects representing the nodes be constructed, as shown, for example, in Code Segment 4.1 on lines 3-10, and those nodes be added to the graph, as shown in Code Segment 4.1 on line 11. The edges between nodes are designated as ordered pairs of nodes, which are added, as seen in Code Segment 4.1 on lines 12-21. Since this is a directed graph, and each edge has a designated source node and target node, the pairs of nodes are listed with the source node first, followed by the target node. Therefore, in Code Segment 4.1, the first edge added (on line 13) goes from $a$ to $b$, the next (on line 14) from $a$ to $c$, etc. For convenience, each graph scenario was added to an overall structure, with a description, and expected result (“True” if valid, “False” otherwise), as seen, for example, in Code Segment 4.1 on lines 22-26. In this way, the tests could all be performed in rapid succession, comparing the expected result with the actual result; if the two matched, the test was considered a success; the code used to perform the tests is outlined in Code Segment B.4 on page 118 on
lines 98-112, and, by function reference, Code Segment B.11 on lines 59-75. For clarity, each test shown here will include the code used to construct the graph, and the corresponding result output is included in each section. Since each of these tests represent the ability to identify the validity of the graph structure, and do not actually perform the inheritance, the arbitrary attributes added to each node are not significant.

4.1.1 Simple, Valid Case

Graph Construction

This scenario, shown in Figure 3.3, represents a valid structure involving eight nodes, two of which inherit from multiple parent nodes. Since this represents a valid structure, under the criteria established in Section 3.2.1, this test is expected to return a “True” value, representing the graph’s recognized validity. The code used to build this graph is found in Code Segment 4.1.

```python
# "Valid, moderately complex test case"
graph = nx.DiGraph()
a = gr.graphnode("A", {'attrib': '1'})
b = gr.graphnode("B", {'attrib': '1'})
c = gr.graphnode("C", {'attrib': '1'})
d = gr.graphnode("D", {'attrib': '1'})
e = gr.graphnode("E", {'attrib': '1'})
f = gr.graphnode("F", {'attrib': '1'})
g = gr.graphnode("G", {'attrib': '1'})
h = gr.graphnode("H", {'attrib': '1'})
graph.add_nodes_from((a, b, c, d, e, f, g, h))
graph.add_edges_from((
    (a, b),
    (a, c),
    (a, d),
    (b, c),
    (d, e),
    (d, f),
    (e, g),
```

52
(e, h), (f, g))
graphs.append(
    {'graph': graph,
     'expected_result': True,
     'description': "Valid, moderately complex test case"
    })

Results

Because this scenario represents a valid structure, no error message was anticipated, and none was returned, as shown in Code Segment 4.2. Therefore, this test was a success.

Code Segment 4.2: Result of Structure Validation for Valid, Moderately Complex Test Case
1 Checking graph: Valid, moderately complex test case
2 Test Success!

4.1.2 Complex, Valid Case

Graph Construction

This scenario, shown in Figure 3.4, represents a slightly more complex, but still valid structure than that shown in Figure 3.3 and Section 4.1.1. This scenario represents a graph of 10 nodes, one of which inherits from multiple parents. In particular, this test represents a non-rooted structure, most significantly demonstrating that the structure need not conform to the definition of a tree (discussed in Section 2.5.2) in order to be usable in this context.

The code used to build this graph is found in Code Segment 4.3.
Code Segment 4.3: Code to Build Valid Organizational Structure with Non-single Root

```python
# "Valid organizational structure with non-single root"

import networkx as nx

graph = nx.DiGraph()
a = gr.graphnode("A", {'attrib': '1'})
b = gr.graphnode("B", {'attrib': '1'})
c = gr.graphnode("C", {'attrib': '1'})
d = gr.graphnode("D", {'attrib': '1'})
e = gr.graphnode("E", {'attrib': '1'})
f = gr.graphnode("F", {'attrib': '1'})
g = gr.graphnode("G", {'attrib': '1'})
h = gr.graphnode("H", {'attrib': '1'})
i = gr.graphnode("I", {'attrib': '1'})
j = gr.graphnode("J", {'attrib': '1'})

graph.add_nodes_from((a, b, c, d, e, f, g, h, i, j))
graph.add_edges_from((a, c), (b, c), (c, d), (c, e), (c, f), (c, g), (e, h), (f, i), (f, j))

graphs.append(
    {'graph': graph,
     'expected_result': True,
     'description': "Valid organizational structure with non-single root"}
)
```
Results

In a situation similar to that seen in Section 4.1.1, this graph is also considered valid, and therefore no error message was anticipated, as was the case as shown in Code Segment 4.4. Therefore, this test was also a success.

Code Segment 4.4: Result of Structure Validation for Valid Structure with Non-single Root

```
Checking graph: Valid organizational structure with non-single root
Test Success!
```

4.1.3 Disconnected Case

Graph Construction

This scenario, as seen in Figure 3.5, represents a scenario in which the graph is disconnected, providing no paths between groups of nodes. In this case, for example, one can traverse among nodes a, b, and c, as well as between nodes d, e, f, g, and h, but it is impossible to traverse between those groups of nodes. This case should, therefore, be determined to be disconnected by the software provided, demonstrating the implementations ability to identify directed graphs. The code used to build this graph is found in Code Segment 4.5.

Code Segment 4.5: Code to Build Invalid, Disconnected Structure

```
# "Invalid, disconnected organizational structure"
graph = nx.DiGraph()
a = gr.graphnode("A", {'attrib': '1'})
b = gr.graphnode("B", {'attrib': '1'})
c = gr.graphnode("C", {'attrib': '1'})
d = gr.graphnode("D", {'attrib': '1'})
e = gr.graphnode("E", {'attrib': '1'})
f = gr.graphnode("F", {'attrib': '1'})
g = gr.graphnode("G", {'attrib': '1'})
```
Results

This scenario represents a disconnected graph, which is not considered valid in this context. Therefore, it was anticipated that the graph validation software would generate an error condition with appropriate error messages. As seen in Code Segment 4.6, this was the case, and therefore this test was also a success.
4.1.4 Cyclic Case

Graph Construction

The scenarios described in Figures 3.6.1 and 3.6.1 represent scenarios in which cycles, as defined in Section 2.5.1, exist between nodes. Each of these scenarios should then be identified as having cycles, and be rejected by the software. The programming code used to build these graphs are found in Code Segments 4.7 and 4.8.

Note that this test in particular used the a facility in the NetworkX graph library to determine whether or not the graph in question was both directed and acyclic. It accomplished this by attempting to generate a topologically-sorted sequence of the graph, using a depth-first traversal. In this test, each node is traversed, and marked as traversed, in a depth-first manner. If any already-traversed node is encountered during the traversal, then a cycle exists in the graph, and the topological order does not exist. Alternatively, if the topologically-sorted sequence does exist, the graph must, by definition, be both directed and acyclic. Therefore, the library code checks the result of the topological sort function, and uses this to determine whether or not a particular graph is acyclic.

Code Segment 4.7: Code to Build Invalid, Simple Cyclic Organizational Structure

```python
# "Simple Invalid, Cyclic organizational structure"
graph = nx.DiGraph()
a = gr.graphnode("A", {'attrib': '1'})
b = gr.graphnode("B", {'attrib': '1'})
c = gr.graphnode("C", {'attrib': '1'})
graph.add_nodes_from((a, b, c))
graph.add_edges_from((
    (a, b),
    (a, c),
    (b, c),
    (c, b)
))
graphs.append(
    {'graph': graph,
     'expected_result': False,
     'description':
    })
```
Results

Like the scenario in Section 4.1.3, the two scenarios here are also considered invalid, since they contain cycles. Therefore, we anticipated error conditions and appropriate messages to be generated during the validation tests. As seen in Code Segment 4.9, these two tests were also successful.
4.2 Inheritance Testing

4.2.1 Graph Construction

The inheritance test example, as shown in Figure 3.7, was modeled using the GraphML XML language, as described in Appendix E, and then interpreted using the appropriate input/output tool. The graph was analyzed, both before and after traversal, outputting each node’s statically-defined and inherited attributes. Because of its length, the XML used to model the graph is shown in Code Segment B.2 on page 116. The programming code used to perform the validation occurs in Code Segment B.4 on lines 114-128.

4.2.2 Results

Because of the length of the output from this test, the full output has been placed in Code Segment B.14 on page 145, and summarized in Table 4.1. Based on this, this test was also a success.

Note that nodes B and H show both statically-assigned, and inherited values. As shown in Code Segment Code Segment B.11 on lines 49-52, any statically-assigned attributes overwrite inherited values for that same attribute name. However, this only occurs as the node is providing its attributes for children to inherit. As a result, although those two nodes show both static and inherited values, this does not adversely affect the inheritance results.
Table 4.1: Inheritance Correctness Testing Results

<table>
<thead>
<tr>
<th>Node</th>
<th>Expected Results</th>
<th>Actual Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Static</td>
<td>Inherited</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>J</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>K</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>L</td>
<td>1.83</td>
<td>1.83</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>N</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

4.3 Performance Testing

4.3.1 Test Specifications

The graphs that were traversed for this performance test were regular trees of specific depths and branching factor. For an example of this, see Figure 4.1. Note that this test only measured the traversal algorithm, and does not include any performance information on any conflict resolution mechanism, as that will be specific to an implementation. The use of a regular tree does not demonstrate any performance measurement of a multiparent structure, but in order to isolate the performance aspects of the traversal algorithm from the specific conflict resolution mechanism, this was necessary.

It is anticipated that the real-world applications of this algorithm will involve the traversal of graphs that are maintained manually, by a system administrator. Therefore, to provide a reasonable upper boundary for the tests, only graphs with less than 4000 nodes were evaluated; it is anticipated that an organizational structure with this many or
more organizations would prefer a more automated approach for designing and building
the organizational structure than that advocated by this research. Since the traversal tests
utilize regular trees of specific depth \( d \) and branching factor \( b_f \), we can easily calculate
the node count using the following equation:

\[
\sum_{n=0}^{d} b_f^n = \frac{b_f^{d+1} - 1}{b_f - 1}
\] (4.1)

Although it was initially intended to calculate all possible combinations of branching
factor and depth, both ranging from 2 to 10, many of these cases were rejected for having too
many nodes. For example, the above equation used for a depth of 10, and branching factor
of 10 yields a graph with 11,111,111,111 nodes. The combinations of depth and branching
factor found in table 4.2 were below the arbitrary 4000 node limit, however, and were tested.
For the performance test, each tree was generated, traversed 1000 times, with the average
traversal time reported.
4.3.2 Platform Specifications

The performance tests were run on two platforms, with the following specifications:

- **Apple MacBook Pro**
  - Processor: Intel Core 2 Duo 2.4 GHz
  - Operating System: Mac OS X v10.5.7 (Leopard)
  - RAM: 2 GB DDR2 667 MHz

- **Dell Latitude 2100 (Netbook)**
  - Processor: Intel Atom N270 1.6 GHz
  - Operating System: Ubuntu Linux 9.04 (Jaunty Jackalope)
  - RAM: 2 GB DDR2 533 MHz

4.3.3 Results

The average time over each set of 1000 traversals is recorded in Table 4.2 and graphed in Figure 4.2. The worst case scenario shown here shows an average of less than 0.1 seconds per traversal. Notice also that, as shown in the graph in Figure 4.2, the traversal algorithm scales linearly with the total number of nodes in the graph.

4.4 Value Demonstration

As described in Section 3.6.3, a set of scenarios were simulated, using the Moab™ scheduling software, to provide an example of the scenario in which the inheritance model described in this document might be of use. In this case, job priority was calculated entirely using the value assigned to the organization to which that user belongs. For further details on the specific values associated with each organization, in each scenario, one may refer to Section 3.6.3.
Table 4.2: Performance Tests: Average Traversal Time

<table>
<thead>
<tr>
<th>Depth</th>
<th>Branching Factor</th>
<th>Node Count</th>
<th>Average Traversal Time (sec)</th>
<th>MacBook</th>
<th>Netbook</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>7</td>
<td>0.0000534</td>
<td>0.0001376</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>13</td>
<td>0.0001026</td>
<td>0.0002623</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>15</td>
<td>0.0001157</td>
<td>0.0002943</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>21</td>
<td>0.0001656</td>
<td>0.0004285</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>31</td>
<td>0.0002420</td>
<td>0.0006319</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>31</td>
<td>0.0002371</td>
<td>0.0006143</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>40</td>
<td>0.0003135</td>
<td>0.0007933</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>43</td>
<td>0.0003352</td>
<td>0.0008822</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>57</td>
<td>0.0004654</td>
<td>0.0011732</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>63</td>
<td>0.0004814</td>
<td>0.0012469</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>73</td>
<td>0.0005969</td>
<td>0.0014873</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>85</td>
<td>0.0006578</td>
<td>0.0017262</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>91</td>
<td>0.0007160</td>
<td>0.0018734</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>111</td>
<td>0.0010090</td>
<td>0.0022755</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>121</td>
<td>0.0009491</td>
<td>0.0021449</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>127</td>
<td>0.0009718</td>
<td>0.0024995</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>156</td>
<td>0.0012135</td>
<td>0.0031754</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>255</td>
<td>0.0019839</td>
<td>0.0053039</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>259</td>
<td>0.0020082</td>
<td>0.0055808</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>341</td>
<td>0.0025409</td>
<td>0.0073033</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>364</td>
<td>0.0028421</td>
<td>0.0077447</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>400</td>
<td>0.0033137</td>
<td>0.0086619</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>511</td>
<td>0.0039618</td>
<td>0.0110720</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>585</td>
<td>0.0047585</td>
<td>0.0129554</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>781</td>
<td>0.0060528</td>
<td>0.0175684</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>820</td>
<td>0.0065442</td>
<td>0.0184196</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>1023</td>
<td>0.0079601</td>
<td>0.0226928</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1093</td>
<td>0.0085606</td>
<td>0.0249106</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>1111</td>
<td>0.0094752</td>
<td>0.0253423</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1365</td>
<td>0.0106066</td>
<td>0.0313206</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>1555</td>
<td>0.0122815</td>
<td>0.0359572</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>2047</td>
<td>0.0162212</td>
<td>0.0471211</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>2801</td>
<td>0.0226412</td>
<td>0.0660253</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>3280</td>
<td>0.0260867</td>
<td>0.0766359</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3906</td>
<td>0.0309576</td>
<td>0.0925488</td>
<td></td>
</tr>
</tbody>
</table>
As also discussed in Section 3.6.3, three metrics are utilized here to illustrate the overall effect of those priority calculations. In both scenarios, since both the overall workload, and the set of resources available to service that workload, were the same, the total running time, measured as the difference between the first job being queued and the last job completing, was also the same, totaling 18,000 seconds, or 5 hours. This can be seen very clearly in Figures 4.3, 4.4, 4.5, and 4.6, where all the workloads’ final termination points, land at the same point on the horizontal axis. However, as those figures also illustrate, significant differences occur in the order and rates in which the various users’, and therefore organizations’ workloads, are serviced.

For the first scenario described in Section 3.6.3, figures 4.3 and 4.4 represent the total number of processors allocated to each user for running jobs, plotted against time. Similarly, Figures 4.5 and 4.6 represent the total remaining processor-seconds for queued jobs, whether

Figure 4.2: Performance Tests: Average Traversal Time
they are running or not. The corresponding results from the second scenario described in Section 3.6.3 are represented by Figures 4.7, 4.8, 4.9, and 4.10, and the third scenario’s results are represented by Figures 4.11, 4.12, 4.13, and 4.14. Notice the following:

- In all three scenarios, job priority is calculated entirely based upon the assigned organization’s priority. In most real-world situations, other components would be included in the priority calculation, including queued time, historical usage, etc., as discussed in Section 2.3.1.

- In the first two scenarios, each user has sufficient job requests to completely fill up the resources available.

- In all scenarios, user3 has a higher-priority organization than either of the other two. However, due to the differences in resources requested among the different scenarios, user3 only fills up the system completely in Scenario 1.

- In the all three scenarios, during the $A^1$ simulation, both user1 and user2 have equal priority. However, only in the first scenario is the workload requested by user1 and user2 exactly identical, and therefore only in Figure 4.3, both lines exactly overlap for the full duration.\(^2\)

- In the $B$ simulations of the first two scenarios, user2 has a priority between that of user1 and user3. Therefore, when user3’s jobs no longer fill up the system, the extra processors are allocated first to any remaining jobs owned by user2, before they are allocated to jobs owned by user1.

- In the third scenario, the total amount of processors and time requested by each user is not even, and therefore the lines representing user1 and user2 do not overlap. Nevertheless, their lines in Figures 4.11 and 4.13 are much closer to each other, demonstrating their equal priority, than the corresponding lines in Figures 4.12 and 4.14.

\(^1\)Recall from Section 3.6.3, that the $A$ scenarios represent the scenarios without multiple inheritance, while the $B$ scenarios allow org2 to utilize the average weight from the other two organizations

\(^2\)To make this clearer, the lines in the corresponding figures utilize lines of differing colors, and differing dash lengths.
• In all three scenarios, both A and B simulations have user3 with the highest priority.

As the graphs demonstrate, the metrics measured for user3 do not change between the two states of each simulation scenario.

While none of these scenarios represent any change in the total amount of time for all jobs to execute, they do represent significant differences in the order in which resources are allocated. Therefore, while this is not the limit to which the inherited values may be put, this represents one potential use of that information, one that represents many of the common scenarios and concerns in high-performance computational resource scheduling.

Of particular interest is the third scenario, as represented in Figures 4.11, 4.12, 4.13, and 4.14. While this is not precisely a real-world simulation, since the captured real-world data would have included many more than three users, and correspondingly more confusing graphs, the total job running times and resource requests are faithful to real-world data. Therefore, the inclusion of this scenario is valuable to demonstrate that the data model and inheritance algorithm shown and demonstrated here, is viable in real-world scenarios, as well as the contrived scenarios demonstrated in the first two sets of simulations.

4.5 Code Samples

Throughout this document, a number of passages refer to particular features of the program development. Although the example implementation is included in its entirety in Appendix B, characteristic examples of these features are reproduced here for clarity.

4.5.1 Daemonization

In the POSIX world, daemonizing a program effectively orphans a process, disconnecting its standard input and output streams, etc., and forces it to become a direct child of the primary, first-to-run process, usually known as init. The code in Code Segment 4.10 shows one method of doing this; no guarantees are made for this code on non-POSIX compliant operating systems. Note that this code is adapted from Jones (2008).
Figure 4.3: Scenario 1A Simulated Processors Allocated per User

Figure 4.4: Scenario 1B Simulated Processors Allocated Per User
Figure 4.5: Scenario 1A Simulated Processor-seconds Queued Per User

Figure 4.6: Scenario 1B Simulated Processor-seconds Queued Per User
Figure 4.7: Scenario 2A Simulated Processors Allocated Per User

Figure 4.8: Scenario 2B Simulated Processors Allocated Per User
Figure 4.9: Scenario 2A Simulated Processor-seconds Queued Per User

Figure 4.10: Scenario 2B Simulated Processor-seconds Queued Per User
Figure 4.11: Scenario 3A Simulated Processors Allocated Per User

Figure 4.12: Scenario 3B Simulated Processors Allocated Per User
Figure 4.13: Scenario 3A Simulated Processor-seconds Queued Per User

Figure 4.14: Scenario 3B Simulated Processor-seconds Queued Per User
The code is invoked by another program to become a daemon, as shown in Code Segment 4.11.
4.5.2 Checking for File Updates

The code utilized to check if the configuration file and structure model file have changed, relies heavily upon the `stat` function provided by POSIX operating systems, which allows one to compare the `mtime` attribute, which represents when the file was last modified. As before, this behavior is not guaranteed on non-POSIX operating systems. An example of how this is done, is found in Code Segment 4.12.

Code Segment 4.12: Checking for File Updates

```python
import os

cfgpaths_stats = {}

# get stats so I can check modify times later
for path in cfgpaths:
    cfgpaths_stats[path] = os.stat(path)

## if changes to config file
rereadconfig = False
for path in cfgpaths:
    print "Debugging: checking if cfg file ", path, " has changed"
    tmpstat = os.stat(path)
    if (tmpstat.st_mtime > cfgpaths_stats[path].st_mtime):
        print "Debugging: cfg file ", path, " has changed"
        cfgpaths_stats[path] = tmpstat
        rereadconfig=True

## re-read config file
if (rereadconfig):
    print "Debugging: rereading cfg files"
```
4.5.3 Reading Configuration File and Command-line Arguments

Command-line arguments are read and parsed using the `optparse` library provided by Python. This is illustrated in Code Segment 4.13.

Code Segment 4.13: Reading and Parsing Command-line Arguments

```python
from optparse import OptionParser

# Instantiate Parser
cliparser = OptionParser()

# Set up options
# short "-c" and long "--configfile" are both accepted
# Value is stored as a string in "CONFIG_PATH"
# Help string describing the option is provided as well
cliparser.add_option(’−c’,’−−configfile’, action=’store’,
    type=’string’, dest=’CONFIG_PATH’,
    default=’./multiparent.conf’,
    help=’Path to the local configuration file (defaults to
    
    
    ”./multiparent.conf”)’)

# Extract parsed data
(options, args) = cliparser.parse_args()

# Utilize the value in options.CONFIG_PATH
cfgpaths = (DEFAULT_CONFIGDEFAULT_PATH, options.CONFIG_PATH)
```

Configuration file parsing is handled by the `ConfigParser` library, also provided by Python, as illustrated in Code Segment 4.14.
import ConfigParser

config = ConfigParser.ConfigParser()
DEFAULT_CONFIG_DEFAULT_PATH = './default.conf'

#Read Configuration from list of paths
configpaths = (DEFAULT_CONFIG_DEFAULT_PATH, options.CONFIG_PATH)
config.read(configpaths)

#Output current configuration
print "Debugging: new config: ", config.items('multiparent')

Example output:
[ ("conflictresolverfunction", 'average'), 
 ("exportfunction", 'export_graph'), 
 ("structuremodelfile", 'mymodelfile.xml'), 
 ("exportmodule", 'exporter'), 
 ("iterationsleepinterval", '5' ), 
 ("reconfigtime", '400' ), 
 ("conflictresolvermodule", 'conflictresolver') ]

4.5.4 Dynamic Loading of Functions

The dynamic loading of both the conflict resolution mechanism, and the model export mechanism, is accomplished by some unique applications of the import and getattr functions in Python. The former is used to load a module, passing the module’s name as a parameter to the function, and assigning the result to a variable name. The latter is used to extract a specific function by name, from the module in question, and assign that into a variable. An example of this, and how those functions are used, are found in Code Segment 4.15.

code_segment_4_15.py

conflictresolver_mod_name = getconfig(config, 'conflictresolvermodule')

76
4.5.5 Graph Traversal

The graph traversal algorithm utilized here is described in Section 3.5.1, and is shown in Code Segment 4.16.

Code Segment 4.16: Graph Traversal Algorithm

```python
def traverse_graph(g):
    for node in g:
        # Clear all "traversed" flags
        node.traversed = False
        # and clear all inherited attrs
        node.inheritedattrs = {}

    # loop through nodes separately, after all have been cleared
    for node in g:
        # loop through each node
        traverse_node(node, g)

def traverse_node(node, graph):
```

```
tempattribs = {}

if(not node.traversed):
    for pn in graph.predecessors_iter(node):
        pnvals = traverse_node(pn, graph)
        for pnattrib in pnvals:
            if( pnattrib in tempattribs ):
                tempattribs[pnattrib].append( pnvals[pnattrib] )
            else:
                tempattribs[pnattrib] = [ pnvals[pnattrib] ]
        node.traversed=True
    # else:
    # already traversed

for attrib in tempattribs:
    if(len(tempattribs[attrib]) == 1):
        node.inheritedattrs[attrib] = tempattribs[attrib][0]
    else:
        # conflict resolution
        node.inheritedattrs[attrib] = conflict_resolve(tempattribs[attrib])

retval = {}

for attr in node.inheritedattrs:
    retval[attr] = node.inheritedattrs[attr]
for attr in node.attrs:
    retval[attr] = node.attrs[attr]

return retval

... if(traversal_needed):
    # traverse model
    traverse_graph(graph)
    traversal_needed = False
4.5.6 Data Serialization

The graph data serialization algorithm utilizes the GraphML standard for representing a graph in an XML format, utilizing the xml.dom.minidom library included with Python. The algorithm is outlined in Code Segment 4.17.

Code Segment 4.17: Data Serialization

```python
def serialize_graph(graph, graph_output_filename =  "serialized.xml"):
    import os
    graph_text = export_graphml(graph, False, False)
    graph_output_file = open(graph_output_filename + " .tmp", "w")
    graph_output_file.write(graph_text)
    graph_output_file.close()
    os.rename(graph_output_filename + " .tmp", graph_output_filename)

def export_graphml(g, exportinherited=False, prettyoutput=True):
    import copy as cp
    import xml.dom.minidom as md
    xmldoc = md.Document()
    graph = xmldoc.createElement('graph')
    graphml = xmldoc.createElement('graphml')
    graphml.appendChild(graph)
    xmldoc.appendChild(graphml)
    graph.setAttribute('edgedefault', 'directed')
    graphml.setAttribute('xmlns', 'http://graphml.graphdrawing.org/xmlns')
    graphml.setAttribute('xmlns:xsi', 'http://www.w3.org/2001/XMLSchema-instance')
    graphml.setAttribute('xsi:schemaLocation', "http://graphml.graphdrawing.org/xmlns http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd")
    for node in g:
        localattribs = {}
        if (exportinherited):
            localattribs = cp.copy(node.inheritedattributes)
        localattribs.update(node.attributes)
```
else:
    localattrs = cp.copy(node.attrs)
xmlnode = xmldoc.createElement('node')
xmlnode.setAttribute('id', node.name)
graph.appendChild(xmlnode)
for key, val in localattrs.iteritems():
    dataattrib = xmldoc.createElement('data')
    dataattrib.setAttribute('key', key)
    valueattrib = xmldoc.createTextNode(str(val))
    dataattrib.appendChild(valueattrib)
    xmlnode.appendChild(dataattrib)

for edge in g.edges():
    xmledge = xmldoc.createElement('edge')
    xmledge.setAttribute('source', edge[0].name)
    xmledge.setAttribute('target', edge[1].name)
    graph.appendChild(xmledge)
if (prettyoutput):
    return xmldoc.toprettyxml("UTF-8")
else:
    return xmldoc.toxml("UTF-8")

...serialize_graph(graph)

4.5.7 Conflict Resolution Functions

The conflict resolution functions must conform to a simple prototype. The code must provide a simple array of values as a parameter, and return the result. Each time a conflict occurs, the conflict resolution function will be called, and pass the conflicting values to the function. In listing 4.18, we see examples of a keep-sum function, a keep-average function, a keep-
maximum function, and a keep-minimum function. Note that these functions are also found in Section B.2.1.
Code Segment 4.18: Conflict Resolution Function Examples

```python
def sum( attribs ):
    total=0
    for i in attribs:
        total += float(i)
    return total

def average( attribs ):
    return sum(attribs)/len(attribs)

def max( attribs ):
    maximum = -1
    for i in attribs:
        if (maximum == -1 or i > maximum):
            maximum = i
    return float(maximum)

def min( attribs ):
    minimum = -1
    for i in attribs:
        if (minimum == -1 or i < minimum):
            minimum = i
    return float(minimum)
```

4.5.8 Data Export

The externally-configurable data export functionality bears a significant resemblance to the data serialization code described in Section 4.5.6. This particular code sample, though, outputs information appropriate for utilization as the `chargerate` parameter for the `account` (or `project`) credential of the Moab™-scheduling software, through the Identity Manager interface. (Adaptive Computing, Inc., 2009, 24.4)

Code Segment 4.19: Data Export

```python
def export_graph(graph):
    import copy as cp
    for node in graph:
```
localattrs = cp.copy(node.inheritedattrs)
localattrs.update(node.attribs)
for key, val in localattrs.items():
    if (key=="Val"):
        print "acct:%s chargerate=%f" % (node.name, val)
5 CONCLUSIONS

5.1 Research Summary

This thesis presents the need for a new computer model for organizations, one which closely represents the multi-parent nature of many modern organizations. In many situations, an organization can be conceptually linked to multiple parent organizations, and therefore if the organizational structure is to be used in resource allocation methodologies, including those utilized in high-performance computing systems, some method of arbitrating that multi-parent relationship is essential.

In the context of high-performance computing systems, resource scheduling algorithms can interact with external information sources to provide relative weights for different organizations, which are used to calculate the priority of resource requests, as demonstrated in Section 4.4. If these values are to be inherited from parent to child organizations, a situation in which multiple parent organizations exist provides an interesting challenge. In particular, as outlined in Sections 1.1, 2.2 and 2.4.1, existing approaches to modeling organizational structures either ignore the inter-organization structure entirely, or restrict each organizational structure to a simple tree data model (as described in Section 2.5.2), allowing only one parent organization to be designated. Therefore, this research illustrates the necessity of a framework, to allow an implementer to model those multi-parent relationships, inherit arbitrary attributes from those parents, and appropriately (as defined by the implementer) deal with inheritance conflicts. The inheritance conflict resolution functionality, in particular, must be installation specific, and therefore the framework must be sufficiently modular to easily load that conflict resolution policy, even dynamically changing that policy
as the corresponding configuration information is changed. An individual implementer may implement any number of inheritance conflict resolution policies, including *keep-maximum*, *keep-minimum*, *keep-average*, or any others as appropriate for the organizational structure and policies. For example, the organizational structure shown in Figure 3.7 on page 45 demonstrates the usefulness of a complex structure, coupled with a simple inheritance conflict resolution policy, a *keep-average* policy in this case.

This research also provides a reference implementation of this framework, implementing those data and inheritance models, including code samples for managing organization model files and periodically serializing the data, both using the *GraphML* standard (see Section 3.5.1 and Appendix E), and providing a mechanism for external tools and formats to be integrated. The performance of the traversal algorithm (see Sections 3.5.1 and 4.3) used to create the critical multiple-inheritance mechanism, provides exceptional scalability, certainly sufficient for the graph sizes and node counts anticipated.

When considering the results of this research, the scalability of the traversal algorithm in particular should be emphasized. It is anticipated that the organizational structures utilized in this context will be maintained manually by the implementer, by editing the data model file, as defined in the configuration file syntax documented in Appendix C. Therefore, the arbitrary limit of 4000 organizations utilized for the scalability tests in Section 4.3 seemed reasonable. However, as illustrated in that same section, the performance of the algorithm is linear, or \( O(n) \), even on differing hardware resources. Given this, it is reasonable to assume that similar scalability would continue, with the total time being determined by the number of nodes traversed, should even larger organizations be modeled. As shown in Table 4.2, and discussed in Section 4.3.3, this traversal time on relatively meager system was less than \( \frac{1}{10} \) of one second. Within that arbitrary 4000 node limitation, therefore, the average traversal time is so minimal as to present no real difficulty in implementing this in a server or virtual machine of relatively minuscule capabilities, thereby demonstrating the versatility for installing this implementation.
5.2 Value of Research

The data model refinement described in this research represents a subtle change to existing organizational modeling facilities. In particular, the existing facilities and approaches either ignore the structure of the organizations entirely (see Section 1.1), providing no inheritance at all, or they provide a simple tree-based model (see Sections 2.2 and 2.4.1), which only provides for single-parent structures. Tree-based models, as seen in x.500 style directory services, in particular are insufficient since the lack of multi-parent association precludes the ability to inherit arbitrary attributes from those parents. In addition, these tree-based systems allow for schema or node structure inheritance, but fail to allow for attribute inheritance, a necessary prerequisite for the inheritance model described here.

As a result of the subtlety, the value of this research may not necessarily be immediately apparent. However, as described in Section 5.1, this research provides a mechanism by which complex organizations can be more accurately modeled, than can be accomplished using existing tools. In both the context of high-performance computing, and elsewhere, administrative staff have often been forced to compromise, finding alternatives to applying the desired weights or values to each organization. Commonly, for example, the organizations at the edges of the structure, such as individual research groups, are the only ones considered, and the result is a disconnected, flat organizational structure, like that shown in Figure 1.1 on page 3. The new data model proposed by this research, and its associated algorithms and reference implementation, provide a mechanism by which the organization can be modeled as it actually exists. The data and inheritance models allow for all practical organizational structures to be modeled, and therefore organizations can have resources assigned according to an appropriate concept of fairness, as defined within that framework.

Throughout this text, examples of the usefulness of this research have focused on those relating to high-performance computing. Nevertheless, this data model and associated inheritance could be utilized in any number of situations in which some resource is shared among organizations that relate to each other in complex ways. Any similar structure could
be modeled in a similar way, and similar concepts applied. For example, perhaps a corporation has some limited resource, say time in one of a small set of conference rooms, and the corporation’s management determines that certain large divisions within that organization need to be assigned differing priorities, but at the edges of that organization’s structure, collaborative teams have begun to form, and therefore the multi-parent organization still exists, and the problem then becomes one of priority based scheduling, based on values that may be multiply inherited. Though this application differs, the underlying problem and concepts are still the same. In essence, any time in which a complex, multi-parent organization exists, with uneven resource allocation levels or weights assigned to organizations at any level, the data model described here will allow that model to accurately reflect the real relationships between those organizations, and still preserve the allocation values in a consistent, deterministic way.

5.3 Further Research

As is often the case with academic research, while this research provides a solution to the described problem, additional research and development are still warranted, as described below.

5.3.1 Graph Algorithm Limitations

While the graph correctness examples described in Sections 3.6.1 and 4.1 provide some basic scenarios to verify graph correctness, they do have some shortcomings. In particular, the test utilized in multiparent.py (see Listing B.11) on lines 68-69 converts the graph temporarily from a directed graph to an undirected graph, in order to utilize the pre-built test for connectivity, contained in the NetworkX library. (Hagberg, 2009) While this is sufficient for the examples shown, one can certainly build directed graphs that are considered disconnected, that when converted to undirected graphs are considered connected. An example of this is shown in Figure 5.1. By the strict definition, a graph is connected, “if every pair of vertices
is connected by a path [or series of edges]" (Cormen et al., 1990, 88) Therefore, while the undirected graph in Figure 5.3.1 can be considered connected, its partner in Figure 5.3.1 is not, since directional edges can only be traversed in one direction; in this case, for example, no path connects node B to A, although the reverse is true.

In order to alleviate this shortcoming, two things become necessary. First, one needs to consider whether or not the case of the disconnected graph presents a particular problem. For the purposes of this research, that particular facet was not explored, allowing a more simplistic subset of possible structures to be evaluated. Second, assuming that future research indicates that disconnected graphs are still to be disallowed, a more complete algorithm for detecting disconnected graphs must be devised. A simple, admittedly naive approach might be to evaluate the graph, attempting to traverse between pairs of points. Unfortunately, the number of pairs of nodes, and therefore potential paths, grows with the factorial of the number of nodes. In fact, since this would mean traversing both directions between pairs of nodes, these pairs are considered unordered, and at the arbitrary level of 4000 nodes defined as “reasonable” in Section 4.3.1, the total number of node pairs is approximately 8 million, as shown in Equation 5.1, a standard combinatorics equation.

Given this, we see how quickly the number of potential pairs of nodes increases. Even neglecting the non-trivial time required to actually traverse between them, the number of nodes increases quite quickly, and may overwhelm such a naive algorithm. Therefore, while
there may be more efficient ways accomplish the same task, that was considered beyond the scope of this research.

\[
\binom{n}{k} = \frac{n!}{k!(n-k)!}
\]

\[
\binom{4000}{2} = \frac{4000!}{2!3998!}
\]

\[
\binom{4000}{2} = \left( \frac{4000}{2} \right) \left( \frac{3999}{1} \right)
\]

\[
\binom{4000}{2} = (2000)(3999)
\]

\[
\binom{4000}{2} = 7998000
\]

5.3.2 Users’ Relationship to Organizations

As discussed in Section 1.4, this research has deliberately excluded any mechanism for associating users with the organizations modeled. Presumably this can be managed without much difficulty with the current many-to-one relationship mechanisms that most credential systems provide, including the POSIX-style user group credential, where a user may be a part of many groups. However, if a different relationship between users and organizations is desired, there might presumably be some implications for the organizational structure mechanisms as well. Additionally, depending on use cases, an inheritance conflict similar to that resolved by this research may occur as users are associated with different organizations. This research could potentially be extended to resolve those conflicts as well.

5.3.3 User Interface

Because this document’s research and associated software development represent a backend framework, no attempt was made to develop a full product, in which an easy-to-use interface
would be included to administer both the system preferences, and perhaps more importantly, the organizational structure being developed. As it stands, the framework designed would require that the individual administrator simply model the organization as appropriate using the GraphML format, as described in Appendix E, and the software configuration with the configuration file, as described in Appendix C. The example implementation provided with this document does allow these files to be automatically re-read and applied during the periodic execution, but additional tools for managing those files or corresponding contents, would likely be necessary before the framework described here would become a marketable product.

5.4 Conclusion

As described in Sections 5.1 and 5.2, this research represents a subtle tuning to the organizational modeling tools in common use. However, this research has shown the viability of that adjusted data model, demonstrated the scalability of the traversal algorithms described, and illustrated real-world uses of the technology, thereby showing the data model and associated algorithms to be a success.

The reference implementation included with this research, while not the most robust or well-designed code, was successful in demonstrating all the facets of the data model and associated inheritance. The performance of the traversal algorithm was shown to have excellent scalability, and the average traversal time was shown to be trivial within the node counts tested. There are certainly other aspects to explore, and as discussed in Section 5.3, more should be done before this technology is ready for full deployment as a commercial application. However, the performance and correctness tests show that this reference implementation was a significant success in the development and demonstration of the algorithms discussed.
References


A SOFTWARE LICENSING

A.1 Software License Association

The software utilized in developing this thesis and associated reference implementation, is subject to the licenses listed in Table A.1. Unless otherwise noted here, any software built for this thesis is licensed under the GNU General Production License v2.0 (see Section A.2.1).

<table>
<thead>
<tr>
<th>Software</th>
<th>License Name</th>
<th>License Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>PSF License v2</td>
<td>Section A.2.4 on page 106</td>
</tr>
<tr>
<td>NetworkX Library</td>
<td>GNU LGPL v 2.1</td>
<td>Section A.2.2 on page 100</td>
</tr>
<tr>
<td>GraphML Specification</td>
<td>Creative Commons Attrib. 3.0</td>
<td>Section A.2.5 on page 110</td>
</tr>
<tr>
<td>LXML Python Library</td>
<td>BSD (3 Clause)</td>
<td>Section A.2.3 on page 106</td>
</tr>
</tbody>
</table>

A.2 License Texts

A.2.1 GNU GPL v2.0

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software— to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

95
When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors’ reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone’s free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
(c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

(a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
(b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
(c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these
terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

**NO WARRANTY**

11. **Because the Program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law.** Except when otherwise stated in writing the copyright holders and/or other parties provide the program “as is” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. **Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.**

12. **In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited**
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

one line to give the program’s name and a brief idea of what it does.

Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) yyyy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands show w and show c should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than show w and show c; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.
A.2.2 GNU LGPL v2.1

GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author’s reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the “Lesser” General Public License because it does Less to protect the user’s freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over
competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users’ freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0 This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1 You may copy and distribute verbatim copies of the Library’s complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2 You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. The modified work must itself be a software library.

b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
c You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3 You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4 You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5 A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted,
regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6 As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7 You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8 You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who
have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9 You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10 Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11 If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12 If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13 The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14 If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15 BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PRO-
VIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the library’s name and a brief idea of what it does.
Copyright (C) year name of author
This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.
This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.
You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.
You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library ‘Frob’ (a library for tweaking knobs) written by James Random Hacker.
¡signature of Ty Coon¿, 1 April 1990 Ty Coon, President of Vice

That’s all there is to it!
A.2.3 Modified BSD ("3 clause") License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

A.2.4 Python Software Foundation License, v2

A HISTORY OF THE SOFTWARE

Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands as a successor of a language called ABC. Guido remains Python’s principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see http://www.cnri.reston.va.us) in Reston, Virginia where he released several versions of the software.

In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations (now Zope Corporation, see http://www.zope.com). In 2001, the Python Software Foundation (PSF, see http://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation is a sponsoring member of the PSF.

All Python releases are Open Source (see http://www.opensource.org for the Open Source Definition). Historically, most, but not all, Python releases have also been GPL-compatible; the table below summarizes the various releases.
<table>
<thead>
<tr>
<th>Release</th>
<th>Derived from</th>
<th>Year</th>
<th>Owner</th>
<th>GPL-compatible?</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9.0 thru 1.2</td>
<td></td>
<td>1991-1995</td>
<td>CWI</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>1.3 thru 1.5.2</td>
<td>1.2</td>
<td>1995-1999</td>
<td>CNRI</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>1.5.2</td>
<td>2000</td>
<td>CNRI</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>2.0</td>
<td>1.6</td>
<td>2000</td>
<td>BeOpen.com</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>1.6.1</td>
<td>1.6</td>
<td>2001</td>
<td>CNRI</td>
<td>yes</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>2.0+1.6.1</td>
<td>2001</td>
<td>PSF</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>2.0.1</td>
<td>2.0+1.6.1</td>
<td>2001</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.1.1</td>
<td>2.1+2.0.1</td>
<td>2001</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>2.1.1</td>
<td>2002</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.1.2</td>
<td>2.1.1</td>
<td>2002</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.1.3</td>
<td>2.1.2</td>
<td>2002</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.2.1</td>
<td>2.2</td>
<td>2002</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.2.2</td>
<td>2.2.1</td>
<td>2002</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.2.3</td>
<td>2.2.2</td>
<td>2003</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>2.2.2</td>
<td>2002-2003</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.3.1</td>
<td>2.3</td>
<td>2002-2003</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.3.2</td>
<td>2.3.1</td>
<td>2002-2003</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.3.3</td>
<td>2.3.2</td>
<td>2002-2003</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.3.4</td>
<td>2.3.3</td>
<td>2004</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.3.5</td>
<td>2.3.4</td>
<td>2005</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>2.3</td>
<td>2004</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.4.1</td>
<td>2.4</td>
<td>2005</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.4.2</td>
<td>2.4.1</td>
<td>2005</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.4.3</td>
<td>2.4.2</td>
<td>2006</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>2.4</td>
<td>2006</td>
<td>PSF</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

Thanks to the many outside volunteers who have worked under Guido’s direction to make these releases possible.

B TERMS AND CONDITIONS FOR ACCESSING OR OTHERWISE USING PYTHON

PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2

1 This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using this software ("Python") in source or binary form and its associated documentation.

2 Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python alone or in any derivative version, provided, however, that PSF’s License Agreement and PSF’s notice of copyright, i.e., "Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Python Software Foundation; All Rights Reserved" are retained in Python alone or in any derivative version prepared by Licensee.

3 In the event Licensee prepares a derivative work that is based on or incorporates Python or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python.

4 PSF is making Python available to Licensee on an "AS IS" basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT

1GPL-compatible doesn’t mean that we’re distributing Python under the GPL. All Python licenses, unlike the GPL, let you distribute a modified version without making your changes open source. The GPL-compatible licenses make it possible to combine Python with other software that is released under the GPL; the others don’t.

2According to Richard Stallman, 1.6.1 is not GPL-compatible, because its license has a choice of law clause. According to CNRI, however, Stallman’s lawyer has told CNRI’s lawyer that 1.6.1 is ”not incompatible” with the GPL.
NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WAR- 
RANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR 
THAT THE USE OF PYTHON WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5 PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON FOR 
ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT 
OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON, OR ANY DERIVA-

TIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6 This License Agreement will automatically terminate upon a material breach of its terms and 
conditions.

7 Nothing in this License Agreement shall be deemed to create any relationship of agency, part-
nership, or joint venture between PSF and Licensee. This License Agreement does not grant 
permission to use PSF trademarks or trade name in a trademark sense to endorse or promote 
products or services of Licensee, or any third party.

8 By copying, installing or otherwise using Python, Licensee agrees to be bound by the terms and 
conditions of this License Agreement.

BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0

BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

1 This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an office at 160 
Saratoga Avenue, Santa Clara, CA 95051, and the Individual or Organization ("Licensee") access-
ing and otherwise using this software in source or binary form and its associated documentation 
("the Software").

2 Subject to the terms and conditions of this BeOpen Python License Agreement, BeOpen hereby 
grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform 
and/or display publicly, prepare derivative works, distribute, and otherwise use the Software alone 
or in any derivative version, provided, however, that the BeOpen Python License is retained in the 
Software, alone or in any derivative version prepared by Licensee.

3 BeOpen is making the Software available to Licensee on an "AS IS" basis. BEOPEN MAKES 
NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAM-

PLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND DISCLAIMS ANY REPRESEN-
TATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR 
PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY THIRD 
PARTY RIGHTS.

4 BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE SOFT-
WARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS 
AS A RESULT OF MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY DERIVA-
TIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

5 This License Agreement will automatically terminate upon a material breach of its terms and 
conditions.

6 This License Agreement shall be governed by and interpreted in all respects by the law of 
the State of California, excluding conflict of law provisions. Nothing in this License Agree-

ment shall be deemed to create any relationship of agency, partnership, or joint venture be-

tween BeOpen and Licensee. This License Agreement does not grant permission to use BeOpen 
trademarks or trade names in a trademark sense to endorse or promote products or services 
of Licensee, or any third party. As an exception, the "BeOpen Python" logos available at 
http://www.pythonlabs.com/logos.html may be used according to the permissions granted on that 
web page.

7 By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms 
and conditions of this License Agreement.

CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1

1 This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, hav-
ing an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or 
Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary 
form and its associated documentation.
Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright (c) 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement, Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the Internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the Internet using the following URL: http://hdl.handle.net/1895.22/1013".

In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.

CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

This License Agreement will automatically terminate upon a material breach of its terms and conditions.

This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

By clicking on the "ACCEPT" button where indicated, or by copying, installing or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

ACCEPT

CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2

Copyright (c) 1991 - 1995, Stichting Mathematisch Centrum Amsterdam, The Netherlands. All rights reserved.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Stichting Mathematisch Centrum or CWI not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF
A.2.5 Creative Commons Attribution 3.0

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE (“CCPL” OR “LICENSE”). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

   a. “Collective Work” means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with one or more other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

   b. “Derivative Work” means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image (“synching”) will be considered a Derivative Work for the purpose of this License.

   c. “Licensor” means the individual, individuals, entity or entities that offers the Work under the terms of this License.

   d. “Original Author” means the individual, individuals, entity or entities who created the Work.

   e. “Work” means the copyrightable work of authorship offered under the terms of this License.

   f. “You” means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

   b. to create and reproduce Derivative Works provided that any such Derivative Work, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked “The original work was translated from English to Spanish,” or a modification could indicate “The original work has been modified.”;

   c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

   d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works.
e. For the avoidance of doubt, where the Work is a musical composition:
   i. Performance Royalties Under Blanket Licenses. Licensor waives the exclusive right to collect,
      whether individually or, in the event that Licensor is a member of a performance rights society
      (e.g. ASCAP, BMI, SESAC), via that society, royalties for the public performance or public
      digital performance (e.g. webcast) of the Work.
   ii. Mechanical Rights and Statutory Royalties. Licensor waives the exclusive right to collect,
       whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency),
       royalties for any phonorecord You create from the Work (“cover version”) and distribute,
       subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act
       (or the equivalent in other jurisdictions).

f. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound
   recording, Licensor waives the exclusive right to collect, whether individually or via a performance-
   rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of
   the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright
   Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised.
The above rights include the right to make such modifications as are technically necessary to exercise
the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the
   following restrictions:

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work
   only under the terms of this License, and You must include a copy of, or the Uniform Resource
   Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly
   display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on
   the Work that restrict the terms of this License or the ability of a recipient of the Work to exercise
   the rights granted to that recipient under the terms of the License. You may not sublicense
   the Work. You must keep intact all notices that refer to this License and to the disclaimer of
   warranties. When You distribute, publicly display, publicly perform, or publicly digitally perform
   the Work, You may not impose any technological measures on the Work that restrict the ability of
   a recipient of the Work from You to exercise the rights granted to that recipient under the terms
   of the License. This Section 4(a) applies to the Work as incorporated in a Collective Work, but
   this does not require the Collective Work apart from the Work itself to be made subject to the
   terms of this License. If You create a Collective Work, upon notice from any Licensor You must,
   to the extent practicable, remove from the Collective Work any credit as required by Section 4(b),
   as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the
   extent practicable, remove from the Derivative Work any credit as required by Section 4(b), as
   requested.

b. If You distribute, publicly display, publicly perform, or publicly digitally perform the Work (as
   defined in Section 1 above) or any Derivative Works (as defined in Section 1 above) or Collective
   Works (as defined in Section 1 above), You must, unless a request has been made pursuant to
   Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium
   or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if
   supplied, and/or (ii) if the Original Author and/or Licensor designate another party or parties (e.g.
   a sponsor institute, publishing entity, journal) for attribution (“Attribution Parties”) in Licensor’s
   copyright notice, terms of service or by other reasonable means, the name of such party or parties;
   the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource
   Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not
   refer to the copyright notice or licensing information for the Work; and, consistent with Section
   3(b) in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative
   Work (e.g., “French translation of the Work by Original Author,” or “Screenplay based on original
   Work by Original Author”). The credit required by this Section 4(b) may be implemented in any
   reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at
   a minimum such credit will appear, if a credit for all contributing authors of the Derivative Work
   or Collective Work appears, then as part of these credits and in a manner at least as prominent as
the credits for the other contributing authors. For the avoidance of doubt, You may only use the
credit required by this Section for the purpose of attribution in the manner set out above and, by
exercising Your rights under this License, You may not implicitly or explicitly assert or imply any
connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution
Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written
permission of the Original Author, Licensor and/or Attribution Parties.

5. Representations, Warranties and Disclaimer
UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR
OFFERS THE WORK AS-IS AND ONLY TO THE EXTENT OF ANY RIGHTS HELD IN THE
LICENSED WORK BY THE LICENSOR. THE LICENSOR MAKES NO REPRESENTATIONS
OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATU-
TORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MAR-
KETABILITY, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT,
OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRE-
SENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDI-
CTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION
MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO
EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL,
INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF
THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF
THE POSSIBILITY OF SUCH DAMAGES.

7. Termination
a. This License and the rights granted hereunder will terminate automatically upon any breach by You
of the terms of this License. Individuals or entities who have received Derivative Works (as defined
in Section 1 above) or Collective Works (as defined in Section 1 above) from You under this License,
however, will not have their licenses terminated provided such individuals or entities remain in full
compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this
License.
b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration
of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right
to release the Work under different license terms or to stop distributing the Work at any time;
provided, however that any such election will not serve to withdraw this License (or any other
license that has been, or is required to be, granted under the terms of this License), and this
License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous
a. Each time You distribute or publicly digitally perform the Work (as defined in Section 1 above) or
a Collective Work (as defined in Section 1 above), the Licensor offers to the recipient a license to
the Work on the same terms and conditions as the license granted to You under this License.
b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the
recipient a license to the original Work on the same terms and conditions as the license granted to
You under this License.
c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect
the validity or enforceability of the remainder of the terms of this License, and without further
action by the parties to this agreement, such provision shall be reformed to the minimum extent
necessary to make such provision valid and enforceable.
d. No term or provision of this License shall be deemed waived and no breach consented to unless
such waiver or consent shall be in writing and signed by the party to be charged with such waiver
or consent.
e. This License constitutes the entire agreement between the parties with respect to the Work licensed
here. There are no understandings, agreements or representations with respect to the Work not
specified here. Licensor shall not be bound by any additional provisions that may appear in any
communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
B CODE SEGMENTS

B.1 XML Code

B.1.1 XML Document Type Definition

The following XML Document Type Definition, or DTD, is used to validate the serialized form of the XML structure file when read. If the file does not validate, an error message is logged. Also, if the file is invalid, and this is the initial program start, the program will abort. If this is a re-read of the file and the file is invalid, no changes to the memory-internal structure will be made.

Code Segment B.1: XML DTD for Data Structure

```
1  <!ELEMENT graphml (graph)>  
2  <!ELEMENT graph (node+,edge+)>  
3      <!ATTLIST graph id ID #REQUIRED  
4          edgedefault CDATA #FIXED "directed">  
5  <!ELEMENT node (data*)>  
6      <!ATTLIST node id ID #REQUIRED>  
7  <!ELEMENT edge EMPTY>  
8      <!ATTLIST edge source IDREF #REQUIRED  
9          target IDREF #REQUIRED>  
10  <!ELEMENT data (#PCDATA)>  
11      <!ATTLIST data key CDATA #REQUIRED>  
```

B.1.2 GraphML for Inheritance Correctness Testing

The following XML document is the full graph definition used for the inheritance correctness test discussed in Section 4.2.
Code Segment B.2: XML Code for Inheritance Testing Graph

```xml
<?xml version="1.0" encoding="UTF-8"?>
<graphml>
  <graph id="organization_name" edgedefault="directed">
    <node id="A">
      <data key="Val">1</data>
    </node>
    <node id="B">
      <data key="Val">2</data>
    </node>
    <node id="C"/>
    <node id="D"/>
    <node id="E"/>
    <node id="F"/>
    <node id="G"/>
    <node id="H">
      <data key="Val">3</data>
    </node>
    <node id="I"/>
    <node id="J"/>
    <node id="K"/>
    <node id="L"/>
    <node id="M"/>
    <node id="N"/>
    <edge source="A" target="B"/>
    <edge source="A" target="C"/>
    <edge source="B" target="D"/>
    <edge source="B" target="E"/>
    <edge source="B" target="F"/>
    <edge source="C" target="F"/>
    <edge source="C" target="G"/>
    <edge source="C" target="H"/>
    <edge source="D" target="I"/>
    <edge source="D" target="J"/>
    <edge source="E" target="K"/>
    <edge source="F" target="L"/>
  </graph>
</graphml>
```
B.2 Python Code

B.2.1 conflictresolver.py

Code Segment B.3: Conflict Resolution Example Code

```python
1 def sum( attribs ):
2     total=0
3     for i in attribs:
4         total += float(i)
5     return total

6 def average( attribs ):
7     return sum(attribs)/len(attribs)

10 def max( attribs ):
11     maximum = -1
12     for i in attribs:
13         if (maximum == -1 or i > maximum):
14             maximum = i
15     return float(maximum)

17 def min( attribs ):
18     minimum = -1
19     for i in attribs:
20         if (minimum == -1 or i < minimum):
21             minimum = i
```

117
```python
    return float(minimum)

if __name__ == '__main__':
    test = ['1', '2', '3', '3', '4']
    print "Test input: ", test, "
    "; average: ", average(test), "
    "; sum: ", sum(test), "; max: ", "
    max(test), "; min: ", min(test)
```

### B.2.2 correctnesstests.py

#### Code Segment B.4: Correctness Tests

```python
import multiparent as mp
import networkx as nx
import graphnode as gr
import mp_exceptions as mpe
import graphml_readwrite as grw
import conflictresolver as cr

def do_graph_correctness_tests():
    graphs = []

    # "Valid, moderately complex test case"
    graph = nx.DiGraph()
    a = gr.graphnode("A", {'attrib': '1'})
    b = gr.graphnode("B", {'attrib': '1'})
    c = gr.graphnode("C", {'attrib': '1'})
    d = gr.graphnode("D", {'attrib': '1'})
    e = gr.graphnode("E", {'attrib': '1'})
    f = gr.graphnode("F", {'attrib': '1'})
    g = gr.graphnode("G", {'attrib': '1'})
    h = gr.graphnode("H", {'attrib': '1'})
```
graph.add_nodes_from((a, b, c, d, e, f, g, h))
graph.add_edges_from(((a, b), (a, c), (a, d), (b, c), (d, e), (d, f), (e, g), (e, h), (f, g)))
graphs.append({
    'graph': graph,
    'expected_result': True,
    'description': 'Valid, moderately complex test case'
})

# "Valid organizational structure with non-single root"
graph = nx.DiGraph()
a = gr.graphnode("A", {'attrib': '1'})
b = gr.graphnode("B", {'attrib': '1'})
c = gr.graphnode("C", {'attrib': '1'})
d = gr.graphnode("D", {'attrib': '1'})
e = gr.graphnode("E", {'attrib': '1'})
f = gr.graphnode("F", {'attrib': '1'})
g = gr.graphnode("G", {'attrib': '1'})
h = gr.graphnode("H", {'attrib': '1'})
i = gr.graphnode("I", {'attrib': '1'})
j = gr.graphnode("J", {'attrib': '1'})
graph.add_nodes_from((a, b, c, d, e, f, g, h, i, j))
graph.add_edges_from(((a, c), (b, c), (c, d), (c, e), (c, f), (c, g), (e, h), (f, i), (f, j)))
graphs.append({
    'graph': graph,
    'expected_result': True,
    'description': 'Valid organizational structure with non-single root'
})

# "Invalid, disconnected organizational structure"
graph = nx.DiGraph()
a = gr.graphnode("A", {'attrib': '1'})
b = gr.graphnode("B", {'attrib': '1'})
c = gr.graphnode("C", {'attrib': '1'})
d = gr.graphnode("D", {'attrib': '1'})
e = gr.graphnode("E", {'attrib': '1'})
f = gr.graphnode("F", {'attrib': '1'})
g = gr.graphnode("G", {'attrib': '1'})
h = gr.graphnode("H", {'attrib': '1'})

graph.add_nodes_from((a, b, c, d, e, f, g, h))
graph.add_edges_from(((a, b), (a, c), (b, c), (d, e), (d, f), (e, g), (e, h)))

graphs.append({'graph': graph, 'expected_result': False, 'description': "Invalid, disconnected organizational structure"})

# "Simple Invalid, Cyclic organizational structure"

graph = nx.DiGraph()
a = gr.graphnode("A", {'attrib': '1'})
b = gr.graphnode("B", {'attrib': '1'})
c = gr.graphnode("C", {'attrib': '1'})

graph.add_nodes_from((a, b, c))
graph.add_edges_from(((a, b), (a, c), (b, c), (c, b)))

graphs.append({'graph': graph, 'expected_result': False, 'description': "Simple Invalid, Cyclic organizational structure"})

# "Complex Invalid, Cyclic organizational structure"

graph = nx.DiGraph()
a = gr.graphnode("A", {'attrib': '1'})
b = gr.graphnode("B", {'attrib': '1'})
c = gr.graphnode("C", {'attrib': '1'})
d = gr.graphnode("D", {'attrib': '1'})
e = gr.graphnode("E", {'attrib': '1'})
f = gr.graphnode("F", {'attrib': '1'})
g = gr.graphnode("G", {'attrib': '1'})
graph.add_nodes_from( (a,b,c,d,e,f,g) )
graph.add_edges_from( ( (a,b), (a,c), (b,c), (b,d), (b,e),
                      (c,d), (d,g), (e,f), (g,c) ) )
graphs.append( {
    'graph': graph,
    'expected_result': False,
    'description': "Complex Invalid, Cyclic organizational structure"
})

for trial in graphs:
    print "Checking graph: ", trial['description']
    result = False
    try:
        result = mp.check_structure(trial['graph'])
        if(trial['expected_result']):
            print "\tTest Success!"
        else:
            print "\tTest Failure"
    except mpe.graphStructureException, e:
        print "Structure problem: ", e
        if(not trial['expected_result']):
            print "\tTest Success!"
        else:
            print "\tTest Failure"

def do_inheritance_correctness_tests():
    mp.resolver_func = cr.average
    print "Inheritance correctness test:\n"
    graph = grw.do_import_test()
    print "Before graph traversal:"
    for node in graph.nodes():
        print "  Node: name=" , node.name, \
        "\n  \tAttributes: ", node.attrs, \
        "\n"
import sys, os

def daemonize (stdin='\dev/null', \
stdout='\dev/null',\n stderr='\dev/null'):
    # Perform first fork.
    try:
        pid = os.fork(
        if pid > 0:
            sys.stderr.write("exiting first parent\n")
            sys.exit(0) # Exit first parent.
        except OSError, e:
            sys.stderr.write("fork #1 failed: (%d) %s\n" \
(e.errno, e.strerror))
        sys.exit(1)
    # Decouple from parent environment.
ochdir("/")
    os.umask(0)

"Inherited attrs": node.inheritedattrs
mp.traverse_graph(graph)
print "After graph traversal:"
for node in graph.nodes():
    print "Node: name=", node.name, \
    "Attrs": node.attrs, \
    "Inherited attrs": node.inheritedattrs

if _name_ == "__main__":
do_graph_correctness_tests()
do_inheritance_correctness_tests()
os.setsid()

# Perform second fork.
try:
    pid = os.fork()
    if pid > 0:
        sys.stderr.write("exiting second parent\n")
        sys.exit(0)  # Exit second parent.
except OSError, e:
    sys.stderr.write("fork #2 failed: (%d) %sn " %
                    (e.errno, e.strerror))
    sys.exit(1)
sys.stderr.write("process is daemonized\n")
# The process is now daemonized,
# redirect standard file descriptors.
for f in sys.stdout, sys.stderr: f.flush()
si = file(stdin, 'r')
so = file(stdout, 'a+')
se = file(stderr, 'a+', 0)
os.dup2(si.fileno(), sys.stdin.fileno())
os.dup2(so.fileno(), sys.stdout.fileno())
os.dup2(se.fileno(), sys.stderr.fileno())

B.2.4 dtdvalidator.py

Code Segment B.6: DTD Validation Code
default.dtd = """"<ELEMENT graphml (graph)>
        <!ELEMENT graph (node+,edge+)>
        <!ATTLIST graph id ID #REQUIRED
            edgedefault CDATA #FIXED "directed">
        <!ELEMENT node (data*)>
        <!ATTLIST node id ID #REQUIRED>
        <!ELEMENT edge EMPTY>
```python
def dtd_validation_tests():
    xml_tests = {
        "xmlcode" : """"""xml version="1.0" /
        encoding="UTF-8"?"
    }
    <graphml>
        <graph id="organization_name" edgedefault="directed">
            <node id="A">
                <data key="Val">1</data>
            </node>
            <node id="B">
                <data key="Val">2</data>
            </node>
            <node id="C" />
            <node id="D" />
            <node id="E" />
            <node id="F" />
            <node id="G" />
            <node id="H">
                <data key="Val">3</data>
            </node>
            <node id="I" />
            <node id="J" />
            <node id="K" />
            <node id="L" />
            <node id="M" />
            <node id="N" />
            <edge source="A" target="B" />
            <edge source="A" target="C" />
            <edge source="B" target="D" />
            <edge source="B" target="E" />
            <edge source="B" target="F" />
```
43 <edge source="C" target="F" />
44 <edge source="C" target="G" />
45 <edge source="C" target="H" />
46 <edge source="D" target="I" />
47 <edge source="D" target="J" />
48 <edge source="E" target="K" />
49 <edge source="F" target="L" />
50 <edge source="G" target="L" />
51 <edge source="H" target="L" />
52 <edge source="H" target="M" />
53 <edge source="H" target="N" />
54 </graph>
55 </graphml>
56 ", "expected_result": True},
57 {"xmlcode": """"<?xml version="1.0" /
 encoding="UTF-8"?>
58 <graphml>
59 <graph id="organization_name" edgedefault="directed">
60 <node id="A">
61 <data key="Val">1</data>
62 </node>
63 <node id="B">
64 <data key="Val">2</data>
65 </node>
66 <node id="C" />
67 <node id="D" name="value" />
68 <node id="E" />
69 <node id="F" />
70 <node id="G" />
71 <node id="H">
72 <data key="Val">3</data>
73 <data>value</data>
74 </node>
75 <node id="I" />
76 <node id="J" />
77 <node id="K" />
<node id="L" />
<node id="M" />
<node id="N" />
<edge source="A" target="B" />
<edge source="A" target="C" />
<edge source="B" target="D" />
<edge source="B" target="E" />
<edge source="B" target="F" />
<edge source="C" target="F" />
<edge source="C" target="G" />
<edge source="C" target="H" />
<edge source="D" target="I" />
<edge source="D" target="J" />
<edge source="E" target="K" />
<edge source="F" target="L" />
<edge source="G" target="L" />
<edge source="H" target="L" />
<edge source="H" target="M" />
<edge source="H" target="N" />
</graph>
</graphml>

""", "expected_result": False}
)
for test in xml_tests:
    result = isvalid(test["xmlcode"])
    if result["success"] == test["expected_result"]:
        print "Test successful: expected result: ", \
    test["expected_result"], "; actual result: ", \
    result["success"]
else:
    print "Test failed: expected result: ", \
    test["expected_result"], \
    "; actual result: ", result["success"]

126
if not result['success']:
    print "Errors found:\n", result['errors'], "\n"
    print "___________________"

def is_valid(xml, dtd = defaultdtd):
    from StringIO import StringIO
    from lxml import etree
    localdtd = etree.DTD(StringIO(dtd))
    documentroot = etree.XML(xml)
    validation_result = localdtd.validate(documentroot)
    if validation_result:
        return {'success': validation_result, "errors": ()}
    else:
        return {'success': validation_result, "errors": localdtd.error_log }

def is_valid_file(xmlfilepath, dtd = defaultdtd):
    xmlfile = open(xmlfilepath, "r")
    xml=xmlfile.read()
    return is_valid(xml, dtd)

if __name__ == "__main__":
    dtd_validation_tests()
for key, val in localattrs.items():
    if(key=="Val"):
        print "acct:%s chargerate=%f" % (node.name, val)

if __name__ == '__main__':
    import networkx as nx
    import multiparent as mp
    import graphml_readwrite as grw
    import conflictresolver as cr
    mp.resolver_func = cr.average
    g = grw.do_import_test()
    mp.traverse_graph(g)
    export_graph(g)

B.2.6  graphml_readwrite.py

from graphnode import graphnode
import networkx
import mp.exceptions as mpe

mynodelist = {}  #generic node list to keep from creating too many duplicates

def parse_graphml_file(filename):
    file = open(filename, "r")
    graphml_contents = file.read()
    file.close()
    return parse_graphml(graphml_contents)

def parse_graphml(input):
    import xml.dom.minidom
    dom = xml.dom.minidom.parseString(input)
graph = networkx.DiGraph()
for node in dom.getElementsByTagName('node'):
    if node.hasAttribute('id'):
        graphnode = getNode(node.getAttribute('id'))
        graph.add_node(graphnode)
for edge in dom.getElementsByTagName('edge'):
    graph.add_edge(getNode(edge.getAttribute('source')),
                    getNode(edge.getAttribute('target')))
for data in dom.getElementsByTagName('data'):
    textdata = ""
    parentnode = data.parentNode
    nodekey = data.getAttribute('key')
    for textnode in data.childNodes:
        if textnode.nodeType == textnode.TEXT_NODE:
            textdata += textnode.data
        getNode(parentnode.getAttribute('id')).attrs[nodekey] = int(textdata)
return graph

def export_graphml(g, exportinherited=False, prettyoutput=True):
    import copy as cp
    import xml.dom.minidom as md
    xmldoc = md.Document()
    graph = xmldoc.createElement('graph')
    graphml = xmldoc.createElement('graphml')
    graphml.appendChild(graph)
    xmldoc.appendChild(graphml)
    graph.setAttribute('edgedefault', 'directed')
    graphml.setAttribute('xmlns', 'http://graphml.graphdrawing.org/xmlns')
    graphml.setAttribute('xmlns:mx', 'http://www.w3.org/2001/XMLSchema-instance')
    graphml.setAttribute('xsi:schemaLocation', "http://www.w3.org/2001/XMLSchema-instance")
    graphml.setAttribute('xsi:schemaLocation', "http://www.w3.org/2001/XMLSchema-instance")
for node in g:
    localattrs = {}
    if (exportinherited):
        localattrs = cp.copy(node.inheritedattrs)
        localattrs.update(node.attrs)
    else:
        localattrs = cp.copy(node.attrs)
    xmlnode = xmldoc.createElement('node')
    xmlnode.setAttribute('id', node.name)
    graph.appendChild(xmlnode)
    for key, val in localattrs.items():
        dataattrib = xmldoc.createElement('data')
        dataattrib.setAttribute('key', key)
        valueattrib = xmldoc.createTextNode(str(val))
        dataattrib.appendChild(valueattrib)
        xmlnode.appendChild(dataattrib)
    for edge in g.edges():
        xmledge = xmldoc.createElement('edge')
        xmledge.setAttribute('source', edge[0].name)
        xmledge.setAttribute('target', edge[1].name)
        graph.appendChild(xmledge)
        if (prettyoutput):
            return xmldoc.toprettyxml("UTF-8")
        else:
            return xmldoc.toxml("UTF-8")

def getNode(nodename):
    global mynodelist
    if nodename not in mynodelist:
        mynodelist[nodename] = graphnode(nodename, {})
    return mynodelist[nodename]

def do_tests():
    g = do_import_test()
print "Import test done"

do_export_test(g)
print "Export test done"

def do_export_test(g, exportinherited=False, 
                    prettyoutput=False):
    xml = export_graphml(g, exportinherited, prettyoutput)
    print "Returned xml:"
    print xml

def do_import_test():
    import dtdvalidator as dv
    graphml_input="""<xml version="1.0" encoding="UTF-8">"
    <graphml>
        <graph id="organization_name" edgedefault="directed">
            <data key="Val">1</data>
        </node>
        <node id="B">
            <data key="Val">2</data>
        </node>
        <node id="C"/>
        <node id="D"/>
        <node id="E"/>
        <node id="F"/>
        <node id="G"/>
        <node id="H">
            <data key="Val">3</data>
        </node>
        <node id="I"/>
        <node id="J"/>
        <node id="K"/>
        <node id="L"/>
        <node id="M"/>
        <node id="N"/>
        <edge source="A" target="B" />
    </graph>
</graphml>
<edge source="A" target="C" />
<edge source="B" target="D" />
<edge source="B" target="E" />
<edge source="B" target="F" />
<edge source="C" target="F" />
<edge source="C" target="G" />
<edge source="C" target="H" />
<edge source="D" target="I" />
<edge source="D" target="J" />
<edge source="E" target="K" />
<edge source="F" target="L" />
<edge source="G" target="L" />
<edge source="H" target="L" />
<edge source="H" target="M" />
<edge source="H" target="N" />

</graph>
</graphml>

dtdresult = dv.isvalid(graphml_input)
if dtdresult['success']:
    return parse_graphml(graphml_input)
else:
    raise mpe.xmlStructureException("Error validating XML: \
    " + str(dtdresult['errors']))

if __name__ == "__main__":
do_tests()
Used to encapsulate the critical values of the graph node. Not really much more than the equivalent of a struct, but unlike a dict, this is hashable.

```

def __init__(self, name, attrs):
    self.name=name
    self.attrs = attrs
    self.inheritedattrs = {}
    self.istraversed = False # for traversal algorithms later

B.2.8 mp_exceptions.py

Code Segment B.10: Exceptions Specific to Multiparent Application

```
B.2.9  multiparent.py

Code Segment B.11: Main Multiparent Application Code

```python
import graphnode as gn
import networkx as nx
import os
import mp_exceptions as mpe
import ConfigParser
from optparse import OptionParser
import dtdvalidator as dv
import graphml_readwrite as rw
import signal
time
daemonize
import serializer

resolver_func = lambda x: 0

def traverse_graph(g):
    for node in g:
        # Clear all "traversed" flags
        node.traversed=False
        # and clear all inherited attribs
        node.inheritedattrs = {}
        # Loop through nodes separately, after all have been cleared
        for node in g:
            # Loop through each node
            traverse_node(node, g)

def traverse_node(node, graph):
    tempattrs = {}
    if(not node.traversed):
        for pn in graph.predecessors_iter(node):
            pnvals = traverse_node(pn, graph)
            for pnattr in pnvals:
```
if( pnattrib in tempattribs ):
    tempattribs[pnattrib].append( ∪
        pnvals[pnattrib] )
else:
    tempattribs[pnattrib] = [ pnvals[pnattrib] ]
    node.traversed=True
   # else:
   # already traversed

for attrib in tempattribs:
    if(len(tempattribs[attrib]) == 1):
        node.inheritedattribs[attrib] = ∪
            tempattribs[attrib][0]
    else:
        #conflict resolution
        node.inheritedattribs[attrib] = ∪
            conflict_resolve(tempattribs[attrib])
        retval = {}
        for attr in node.inheritedattribs:
            retval[attr] = node.inheritedattribs[attr]
        for attr in node.attribs:
            retval[attr] = node.attribs[attr]
        return retval

def conflict_resolve(attributes):
    global resolver_func
    return resolver_func(attributes)

def check_structure(gr):
    #returns true if structure is acceptable, and RAISES /
    #EXCEPTION otherwise;
    #An exception is raised containing the error strings if /
    #they exist
    errstrs = []
    if( not nx.is_directed(gr) ):
errstrs.append("Graph must be directed.")
else:
    if (not nx.is_directed_acyclic_graph(gr)):
        errstrs.append("Graph must not contain any cycles.")
    if (not nx.is_connected(gr.to_undirected())) :
        errstrs.append("Graph must be connected.")
if (len(errstrs) > 0 ):
    retstring=""
    for i in range(0, len(errstrs)):
        retstring = retstring + errstrs[i] + " "
    raise mpe.graphStructureException(retstring)
return True #If we've gotten this far, it must be good

def getconfig(parser, field, category='multiparent'):
    return parser.get(category, field)

def main():
    #startup
    # Read configuration and cli parameters
    global config
    config = ConfigParser.ConfigParser()
    DEFAULT_CONFIGDEFAULT_PATH='./default.conf'
    #DEFAULT_CONFIG_PATH='./multiparent.conf'
    cliparser = OptionParser()
    cliparser.add_option( '-c', '--configfile', action='store', type='string', dest='CONFIG_PATH', default="./multiparent.conf", \
     help="Path to the local configuration file (defaults to \\
     "./multiparent.conf")")
    (options, args) = cliparser.parse_args()
    cfgpaths = (DEFAULT_CONFIGDEFAULT_PATH, options.CONFIG_PATH,)
    config.read(cfgpaths)
    cfgpaths_stats = {}
#get stats so I can check modify times later

```python
for path in cfgpaths:
    cfgpaths_stats[path] = os.stat(path)
```

#set up signal handlers

```python
signal.signal(signal.SIGHUP, sighandle_HUP)
signal.signal(signal.SIGINT, sighandle_INT)
signal.signal(signal.SIGTERM, sighandle_TERM)
signal.signal(signal.SIGABRT, sighandle_ABRT)
```

##daemonize

```python
daemonize.daemonize()
```

#read model file

```python
modelfile_path = getconfig(config, 'structuremodelfile')
if (dv.isvalid_file(modelfile_path)):
    graph = rw.parse_graphml_file(modelfile_path)
    modelfile_stat = os.stat(modelfile_path)
else:
    raise mpe.xmlStructureException("XML at " + modelfile_path + 
                                     "Fails to validate against DTD")
```

#read conflict resolution mechanism initially:

```python
global resolver_func
conflictreresolver_mod_name = getconfig(config, 'conflictreolvermodule')
conflictreresolver_mod = __import__(conflictreolver_mod_name)
resolver_func = getattr(conflictreolver_mod, getconfig(config, 'conflictreolverfunction'))
```

# global export_func

```python
export_mod_name = getconfig(config, 'exportmodule')
export_mod = __import__(export_mod_name)
export_func = getattr(export_mod, getconfig(config, 'exportfunction'))
```
#always traverse the first time
traversal_needed=True

#Main Loop
while(True):
    if(traversal_needed):
        # traverse model
        traverse_graph(graph)
        traversal_needed = False
        export_func(graph)
        serializer.serialize_graph(graph)
        
        # sleep some interval
        time.sleep(float(getconfig(config,'iterationsleepinterval')))
    
        # if changes to config file
        rereadconfig = False
        for path in cfgpaths:
            print("Debugging: checking if cfg file ", path, " has changed")
            tmpstat = os.stat(path)
            if(tmpstat.st_mtime >
               cfgpaths_stats[path].st_mtime):
                print("Debugging: cfg file ", path, " has changed")
                cfgpaths_stats[path] = tmpstat
                rereadconfig=True
        
        if(rereadconfig):
            #copy old conflictresolver stuff for comparison
            conflictresolver_mod_name_bak = getconfig(config, 'conflictresolvermodule')
            conflictresolver_func_name_bak = getconfig(config, 'conflictresolverfunction')
            export_mod_name_bak = getconfig(config, 'exportmodule')
export_func_name_bak = getconfig(config, 'exportfunction')

### re-read config file

print "Debugging: rereading cfg files"
config.read(cfgpaths)
print "Debugging: new config: ",
    config.items('multiparent')
conflictresolver_func_name_new = getconfig(config, 'conflictresolverfunction')
conflictresolver_mod_name = getconfig(config, 'conflictresolvermodule')
if (conflictresolver_mod_name_bak != conflictresolver_mod_name or \
    conflictresolver_func_name_bak != conflictresolver_func_name_new):
    print "Debugging: need to re-read conflict resolver function"
    #reread conflict resolver function
    conflictresolver_mod =
        __import__(conflictresolver_mod_name)
    resolver_func = getattr(conflictresolver_mod, 'conflictresolver_func_name_new')
    traversal_needed = True
export_func_name_new = getconfig(config, 'exportfunction')
export_mod_name = getconfig(config, 'exportmodule')
if (export_mod_name_bak != export_mod_name or \
    export_func_name_bak != export_func_name_new):
    print "Debugging: need to re-read export function name"
    export_mod = __import__(export_mod_name)
    export_func = getattr(export_mod, 'export_func_name_new')
    traversal_needed = True

139
if changes to model file (including different path to model file in config)

print "Debugging: checking if model file needs re-reading"

modelfile_path = getconfig(config, 'structuremodelfile')
tmpstat = os.stat(modelfile_path)
if(tmpstat.st.ino != modelfile_stat.st.ino or
   tmpstat.st.mtime > modelfile_stat.st.mtime):
   print "Debugging: re-reading model file"

if(dv.isvalid_file(modelfile_path)):
    graph = rw.parse_graphml_file(modelfile_path)
else:
    raise mpe.xmlStructureException("XML at " + modelfile_path + "Fails to validate against DTD")

prepare for traversal
traversal_needed=True
modelfile_stat = tmpstat

#Signal Handlers:
# SIGHUP
def sighandle_HUP(signum, frame):
    print "Debugging: handling SIGHUP"
    # still not sure the best thing to do here. re-read config/file?
    pass

# SIGINT
# clean up memory (if necessary)
# exit
def sighandle_INT(signum, frame):
    print "Debugging: handling SIGINT"
    exit()

# SIGTERM
# clean up memory (if necessary)
# exit

def sighandle_TERM(signum, frame):
    print "Debugging: handling SIGTERM"
    # Note that this is a standard kill, without args
    exit()

# SIGABRT
# Not sure; maybe same as SIGINT and SIGTERM

def sighandle_ABRT(signum, frame):
    print "Debugging: handling SIGABRT"
    pass

if __name__ == '__main__':
    main()

B.2.10 performancetests.py

Code Segment B.12: Performance Test Coding

```python
import mp.exceptions as mpe
import networkx as nx
import graphnode as gn
import time
import copy as cp
import multiparent as mp

def generate_graph(depth, branchingfactor, data = {'attrib1': 12}):
    if (depth <= 1):
        raise mpe.graphGenerationException("Depth must be greater than 1 for the generator")
    if (branchingfactor < 1):
```
raise mpe.graphGenerationException("Branching factor must be greater than or equal to 1 for generator")

graph = nx.DiGraph()
rootnode = gn.graphnode("rootnode", data)
lastgeneration = [rootnode]
graph.add_node(rootnode)
nodecount = 1
edgecount = 0
for depthcounter in range(0, depth):
    lastgeneration_tmp = []
    for parentnode in lastgeneration:
        for branchingfactorcounter in range(0, branchingfactor):
            node = gn.graphnode(str(depthcounter) + "," + str(branchingfactorcounter), {})
            cp.copy({})
            nodecount += 1
            edgecount += 1
            lastgeneration_tmp.append(node)
            graph.add_node(node)
            graph.add_edge(parentnode, node)
    lastgeneration = cp.copy(lastgeneration_tmp)
return graph

def do_performance_test(depth, branchingfactor, iterationcount = 1000):
    graph = generate_graph(depth, branchingfactor)
    time_a = time.time()
    for iter in range(0, iterationcount):
        mp.traverse_graph(graph)
    time_b = time.time()
    return (depth, branchingfactor, iterationcount, time_b-time_a)
def do_performance_tests():
print "depth, branchingfactor, iterationcount, elapsed_sec"
for (depth, branchingfactor) in (2, 2), (3, 2), (4, 2), (5, 2), (6, 2), (7, 2), (8, 2), (9, 2), (10, 2), (2, 3), (3, 3), (4, 3), (5, 3), (6, 3), (7, 3), (2, 4), (3, 4), (4, 4), (5, 4), (2, 5), (3, 5), (4, 5), (5, 5), (2, 6), (3, 6), (4, 6), (2, 7), (3, 7), (4, 7), (2, 8), (3, 8), (2, 9), (3, 9), (2, 10),
def test_generator():
    try:
        g = generate_graph(3, 4)
    except mpe.graphGenerationException, e:
        print e
        exit(-1)
    print "complete graph generation"
    print "Node list:"
    for node in g.nodes():
        print "Node: ", node.name
    for edge in g.edges():
        print "Edge: ", edge[0].name, "->", edge[1].name

if __name__ == "__main__":
    do_performance_tests()
if __name__ == '__main__':
    import networkx as nx
    g = nx.read_gpickle("./inheritance_example.pkl")
    serialize_graph(g, "test_serialize.xml")
    pass

B.3 Captured Output

B.3.1 Inheritance Correctness Validation Output

Code Segment B.14: Inheritance Correctness Validation Output (see Section 4.2)

Inheritance correctness test:

Before graph traversal:

Node: name= A
  Attri bs: {u'Val': 1}
  Inherited attrs: {}

Node: name= N
  Attri bs: {}
  Inherited attrs: {}

Node: name= B
  Attri bs: {u'Val': 2}
  Inherited attrs: {}

Node: name= F
  Attri bs: {}
  Inherited attrs: {}

Node: name= J
  Attri bs: {}
  Inherited attrs: {}

Node: name= E
  Attri bs: {}
  Inherited attrs: {}
Node: name= C
    Attribs: {}
    Inherited attribs: {}
Node: name= D
    Attribs: {}
    Inherited attribs: {}
Node: name= I
    Attribs: {}
    Inherited attribs: {}
Node: name= H
    Attribs: {u'Val': 3}
    Inherited attribs: {}
Node: name= M
    Attribs: {}
    Inherited attribs: {}
Node: name= K
    Attribs: {}
    Inherited attribs: {}
Node: name= G
    Attribs: {}
    Inherited attribs: {}
Node: name= L
    Attribs: {}
    Inherited attribs: {}

After graph traversal:
Node: name= A
    Attribs: {u'Val': 1}
    Inherited attribs: {}
Node: name= N
    Attribs: {}
    Inherited attribs: {u'Val': 3}
Node: name= B
    Attribs: {u'Val': 2}
    Inherited attribs: {u'Val': 1}
Node: name= F
    Attribs: {}
Inherited attrs: {u'Val': 1.5}

Node: name= J
  Attribs: {}
  Inherited attrs: {u'Val': 2}

Node: name= E
  Attribs: {}
  Inherited attrs: {u'Val': 2}

Node: name= C
  Attribs: {}
  Inherited attrs: {u'Val': 1}

Node: name= D
  Attribs: {}
  Inherited attrs: {u'Val': 2}

Node: name= I
  Attribs: {}
  Inherited attrs: {u'Val': 2}

Node: name= H
  Attribs: {u'Val': 3}
  Inherited attrs: {u'Val': 1}

Node: name= M
  Attribs: {}
  Inherited attrs: {u'Val': 3}

Node: name= K
  Attribs: {}
  Inherited attrs: {u'Val': 2}

Node: name= G
  Attribs: {}
  Inherited attrs: {u'Val': 1}

Node: name= L
  Attribs: {}
  Inherited attrs: {u'Val': 1.8333333333333333}
C CONFIGURATION FILE SYNTAX

C.1 Configuration File Introduction

The configuration file utilized in the reference implementation (see Sections 4.5.3 on page 75, and B.2.9 on page 134) utilizes the syntax provided by the ConfigParser library provided by Python. In Sections C.2, and C.3, examples of this syntax are given, and the syntax is explained.

C.2 Configuration File Syntax

The syntax utilized uses sections, combined with name-value pairs. The reference implementation uses the multiparent section, as well as the special DEFAULT section. The values in the multiparent section override any corresponding values in the DEFAULT section. Sections are denoted by the section name enclosed in square brackets, by itself on a line. All name-value pairs listed in the file after this are considered part of that group, until a new section header is encountered. For complete format examples, refer to Section C.3.

The following names are used for the associated parameters in the reference implementation:

- structuremodelfile
  - This contains a path to the structure model file that represents the organization being modeled. This file uses the GraphML standard outlined in Appendix E.

- iterationsleepinterval
  - This contains the number of seconds that the software will sleep, before trying to re-read the configuration and model files, and re-traverse the organizational structure.

- conflictresolvermodule
  - This is the name of the Python module used to supply the conflict resolution mechanism. This module must be in the local Python module path, exactly as
if it were being imported using the `import` modulename command. There must also be a `conflictresolverfunction` defined.

- `conflictresolverfunction`
  - This is the name of the Python function used for conflict resolution. This function must exist inside the module defined by the `conflictresolvermodule` command.

- `exportmodule`
  - This is the name of the Python module used to supply the data export mechanism. This module must be in the local Python module path, exactly as if it were being imported using the `import` modulename command. There must also be an `exportfunction` defined.

- `exportfunction`
  - This is the name of the Python function used for the data export mechanism. This function must exist inside the module defined by the `exportmodule` command.

### C.3 Configuration File Examples

Listings C.1 and C.2 provide examples of the configuration file syntax. As shown in Section 4.5.3 on page 75, multiple files can be imported in the same program. This way, an administrator may define a generic set of defaults, while allowing other administrators to locally override the parameters.

#### Code Segment C.1: Default Configuration File

```plaintext
[DEFAULT]
structuremodelfile=mymodelfile.xml
iterationsleepinterval=60
conflictresolvermodule=conflictresolver
conflictresolverfunction=average
exportmodule=exporter
exportfunction=export_graph
```
Code Segment C.2: Specific Configuration File

1  [multiparent]
2  structuremodelfile=mymodelfile.xml
3  iterationsleeptimeinterval=5
## D REAL-WORLD SIMULATION SCENARIO

The following table shows the jobs simulated for the modified real-world scenario discussed in Sections 3.6.3 and 4.4.

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Nodes Requested</th>
<th>Processors Requested</th>
<th>User</th>
<th>Organization</th>
<th>Time Requested (sec)</th>
<th>Actual Running Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>user3</td>
<td>org3</td>
<td>19800</td>
<td>240</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>3600</td>
<td>1020</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>8</td>
<td>user3</td>
<td>org3</td>
<td>19800</td>
<td>1590</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------</td>
<td>----------------------</td>
<td>-------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>42</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>43</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>44</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>45</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>47</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>48</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>49</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>51</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>52</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>53</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>54</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>55</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>56</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>57</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>58</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>59</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>61</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>----------------------</td>
<td>------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>62</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>63</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>65</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>66</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>67</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>68</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>69</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>70</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>71</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>72</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>73</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>74</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>75</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>76</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>77</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>78</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>79</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>80</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>81</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>82</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>83</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>84</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>85</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>86</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>87</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>88</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>89</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>90</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>91</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>92</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>93</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>94</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>95</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>96</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------</td>
<td>----------------------</td>
<td>--------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>97</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>98</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>99</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>102</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>103</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>104</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>105</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>106</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>107</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>108</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>109</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>112</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>113</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>114</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>115</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>116</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>117</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>118</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>119</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>120</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>121</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>122</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>123</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>124</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>125</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>126</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>127</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>128</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>129</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>130</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>131</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>132</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>133</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>134</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>135</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>136</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>137</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>138</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>139</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>140</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>141</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>142</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>143</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>144</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>145</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>146</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>147</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>148</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>149</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>150</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>19800</td>
<td>2580</td>
</tr>
<tr>
<td>151</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>152</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>153</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>154</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>155</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>156</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>157</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>158</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>159</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>160</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>161</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>162</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>163</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>164</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>165</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>166</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>----------------------</td>
<td>----------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>167</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>168</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>169</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>170</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>171</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>172</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>173</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>174</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>175</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>176</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>177</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>178</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>179</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>180</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>181</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>182</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>183</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>184</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>185</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>186</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>187</td>
<td>1</td>
<td>8</td>
<td>user3</td>
<td>org3</td>
<td>19800</td>
<td>2940</td>
</tr>
<tr>
<td>188</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>189</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>190</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>191</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>192</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>193</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>194</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>195</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>196</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>197</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>198</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>199</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>201</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------</td>
<td>----------------------</td>
<td>------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>202</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>203</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>204</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>205</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>206</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>207</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>208</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>209</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>210</td>
<td>1</td>
<td>8</td>
<td>user3</td>
<td>org3</td>
<td>172800</td>
<td>30</td>
</tr>
<tr>
<td>211</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>212</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>213</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>214</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>215</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>216</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>217</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>218</td>
<td>1</td>
<td>8</td>
<td>user3</td>
<td>org3</td>
<td>172800</td>
<td>30</td>
</tr>
<tr>
<td>219</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>220</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>221</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>222</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>223</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>224</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>225</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>226</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>227</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>228</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>229</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>230</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>231</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>232</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>233</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>234</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>235</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>236</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>-------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>237</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>238</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>239</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>240</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>241</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>242</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>243</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>244</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>245</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>246</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>247</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>248</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>249</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>250</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>251</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>252</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>253</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>254</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>255</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>256</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>257</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>258</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>259</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>260</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>261</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>262</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>263</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>264</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>265</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>266</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>267</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>268</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>269</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>270</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>271</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------</td>
<td>----------------------</td>
<td>--------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>272</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>273</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>274</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>275</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>276</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>277</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>278</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>279</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>280</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>281</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>282</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>283</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>284</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>285</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>286</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>287</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>288</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>289</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>290</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>291</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>292</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>293</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>294</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>295</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>296</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>297</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>298</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>299</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>3600</td>
<td>900</td>
</tr>
<tr>
<td>300</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>301</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>302</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>303</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>304</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>305</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>306</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------</td>
<td>----------------------</td>
<td>--------</td>
<td>--------------</td>
<td>----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>307</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>308</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>309</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>310</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>311</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>312</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>313</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>314</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>315</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>316</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>317</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>318</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>319</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>320</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>321</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>322</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>323</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>324</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>325</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>326</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>327</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>328</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>329</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>330</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>331</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>37800</td>
<td>30</td>
</tr>
<tr>
<td>332</td>
<td>1</td>
<td>8</td>
<td>user3</td>
<td>org3</td>
<td>19800</td>
<td>4350</td>
</tr>
<tr>
<td>333</td>
<td>1</td>
<td>8</td>
<td>user3</td>
<td>org3</td>
<td>19800</td>
<td>5940</td>
</tr>
<tr>
<td>334</td>
<td>1</td>
<td>8</td>
<td>user3</td>
<td>org3</td>
<td>19800</td>
<td>7740</td>
</tr>
<tr>
<td>335</td>
<td>1</td>
<td>8</td>
<td>user3</td>
<td>org3</td>
<td>172800</td>
<td>22290</td>
</tr>
<tr>
<td>336</td>
<td>1</td>
<td>4</td>
<td>user3</td>
<td>org3</td>
<td>172800</td>
<td>55590</td>
</tr>
<tr>
<td>337</td>
<td>16</td>
<td>128</td>
<td>user3</td>
<td>org3</td>
<td>86400</td>
<td>60450</td>
</tr>
<tr>
<td>338</td>
<td>1</td>
<td>8</td>
<td>user1</td>
<td>org1</td>
<td>19800</td>
<td>210</td>
</tr>
<tr>
<td>339</td>
<td>1</td>
<td>8</td>
<td>user2</td>
<td>org2</td>
<td>19800</td>
<td>240</td>
</tr>
<tr>
<td>340</td>
<td>1</td>
<td>8</td>
<td>user2</td>
<td>org2</td>
<td>19800</td>
<td>270</td>
</tr>
<tr>
<td>341</td>
<td>1</td>
<td>8</td>
<td>user1</td>
<td>org1</td>
<td>19800</td>
<td>780</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>----------------------</td>
<td>--------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>342</td>
<td>1</td>
<td>8</td>
<td>user1</td>
<td>org1</td>
<td>19800</td>
<td>930</td>
</tr>
<tr>
<td>343</td>
<td>1</td>
<td>8</td>
<td>user1</td>
<td>org1</td>
<td>19800</td>
<td>240</td>
</tr>
<tr>
<td>344</td>
<td>1</td>
<td>8</td>
<td>user1</td>
<td>org1</td>
<td>19800</td>
<td>210</td>
</tr>
<tr>
<td>345</td>
<td>1</td>
<td>8</td>
<td>user1</td>
<td>org1</td>
<td>19800</td>
<td>1320</td>
</tr>
<tr>
<td>346</td>
<td>1</td>
<td>8</td>
<td>user1</td>
<td>org1</td>
<td>172800</td>
<td>30</td>
</tr>
<tr>
<td>347</td>
<td>1</td>
<td>8</td>
<td>user2</td>
<td>org2</td>
<td>86400</td>
<td>60</td>
</tr>
<tr>
<td>348</td>
<td>1</td>
<td>4</td>
<td>user3</td>
<td>org3</td>
<td>172800</td>
<td>56160</td>
</tr>
<tr>
<td>349</td>
<td>1</td>
<td>8</td>
<td>user1</td>
<td>org1</td>
<td>172800</td>
<td>30</td>
</tr>
<tr>
<td>350</td>
<td>1</td>
<td>8</td>
<td>user2</td>
<td>org2</td>
<td>19800</td>
<td>2130</td>
</tr>
<tr>
<td>351</td>
<td>1</td>
<td>8</td>
<td>user2</td>
<td>org2</td>
<td>19800</td>
<td>3300</td>
</tr>
<tr>
<td>352</td>
<td>1</td>
<td>8</td>
<td>user2</td>
<td>org2</td>
<td>19800</td>
<td>2190</td>
</tr>
<tr>
<td>353</td>
<td>1</td>
<td>8</td>
<td>user2</td>
<td>org2</td>
<td>19800</td>
<td>3810</td>
</tr>
<tr>
<td>354</td>
<td>1</td>
<td>8</td>
<td>user1</td>
<td>org1</td>
<td>19800</td>
<td>4230</td>
</tr>
<tr>
<td>355</td>
<td>1</td>
<td>8</td>
<td>user2</td>
<td>org2</td>
<td>19800</td>
<td>7560</td>
</tr>
<tr>
<td>356</td>
<td>1</td>
<td>8</td>
<td>user2</td>
<td>org2</td>
<td>19800</td>
<td>10020</td>
</tr>
<tr>
<td>357</td>
<td>1</td>
<td>8</td>
<td>user2</td>
<td>org2</td>
<td>172800</td>
<td>15720</td>
</tr>
<tr>
<td>358</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>79410</td>
</tr>
<tr>
<td>359</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>92040</td>
</tr>
<tr>
<td>360</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>104160</td>
</tr>
<tr>
<td>361</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>45840</td>
</tr>
<tr>
<td>362</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>113730</td>
</tr>
<tr>
<td>363</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>112140</td>
</tr>
<tr>
<td>364</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>111960</td>
</tr>
<tr>
<td>365</td>
<td>1</td>
<td>4</td>
<td>user2</td>
<td>org2</td>
<td>172800</td>
<td>55380</td>
</tr>
<tr>
<td>366</td>
<td>1</td>
<td>4</td>
<td>user2</td>
<td>org2</td>
<td>172800</td>
<td>55380</td>
</tr>
<tr>
<td>367</td>
<td>1</td>
<td>4</td>
<td>user1</td>
<td>org1</td>
<td>432000</td>
<td>55590</td>
</tr>
<tr>
<td>368</td>
<td>1</td>
<td>4</td>
<td>user2</td>
<td>org2</td>
<td>172800</td>
<td>56400</td>
</tr>
<tr>
<td>369</td>
<td>1</td>
<td>4</td>
<td>user3</td>
<td>org3</td>
<td>172800</td>
<td>120330</td>
</tr>
<tr>
<td>370</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>119790</td>
</tr>
<tr>
<td>371</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>123000</td>
</tr>
<tr>
<td>372</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>125730</td>
</tr>
<tr>
<td>373</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>128970</td>
</tr>
<tr>
<td>374</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>130260</td>
</tr>
<tr>
<td>375</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>129600</td>
</tr>
<tr>
<td>376</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>106650</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>377</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>137280</td>
</tr>
<tr>
<td>378</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>140790</td>
</tr>
<tr>
<td>379</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>140970</td>
</tr>
<tr>
<td>380</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>142290</td>
</tr>
<tr>
<td>381</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>144630</td>
</tr>
<tr>
<td>382</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>143280</td>
</tr>
<tr>
<td>383</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>146190</td>
</tr>
<tr>
<td>384</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>149970</td>
</tr>
<tr>
<td>385</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>150690</td>
</tr>
<tr>
<td>386</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>150300</td>
</tr>
<tr>
<td>387</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>150390</td>
</tr>
<tr>
<td>388</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>150450</td>
</tr>
<tr>
<td>389</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>123510</td>
</tr>
<tr>
<td>390</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>150810</td>
</tr>
<tr>
<td>391</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>154200</td>
</tr>
<tr>
<td>392</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>156480</td>
</tr>
<tr>
<td>393</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>127110</td>
</tr>
<tr>
<td>394</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>97020</td>
</tr>
<tr>
<td>395</td>
<td>4</td>
<td>32</td>
<td>user1</td>
<td>org1</td>
<td>21600</td>
<td>1470</td>
</tr>
<tr>
<td>396</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>159480</td>
</tr>
<tr>
<td>397</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>130080</td>
</tr>
<tr>
<td>398</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>100410</td>
</tr>
<tr>
<td>399</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>163110</td>
</tr>
<tr>
<td>400</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>160950</td>
</tr>
<tr>
<td>401</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>166320</td>
</tr>
<tr>
<td>402</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>167670</td>
</tr>
<tr>
<td>403</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>112200</td>
</tr>
<tr>
<td>404</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>112380</td>
</tr>
<tr>
<td>405</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>173280</td>
</tr>
<tr>
<td>406</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>117360</td>
</tr>
<tr>
<td>407</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>118380</td>
</tr>
<tr>
<td>408</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>118470</td>
</tr>
<tr>
<td>409</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>119220</td>
</tr>
<tr>
<td>410</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>124860</td>
</tr>
<tr>
<td>411</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>185730</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------</td>
<td>----------------------</td>
<td>--------</td>
<td>--------------</td>
<td>----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>412</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>183510</td>
</tr>
<tr>
<td>413</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>157200</td>
</tr>
<tr>
<td>414</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>185340</td>
</tr>
<tr>
<td>415</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>129000</td>
</tr>
<tr>
<td>416</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>159630</td>
</tr>
<tr>
<td>417</td>
<td>8</td>
<td>32</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>129630</td>
</tr>
<tr>
<td>418</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>161430</td>
</tr>
<tr>
<td>419</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>133680</td>
</tr>
<tr>
<td>420</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>134460</td>
</tr>
<tr>
<td>421</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>134940</td>
</tr>
<tr>
<td>422</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>135690</td>
</tr>
<tr>
<td>423</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>136710</td>
</tr>
<tr>
<td>424</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>137460</td>
</tr>
<tr>
<td>425</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>138450</td>
</tr>
<tr>
<td>426</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>141090</td>
</tr>
<tr>
<td>427</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>145620</td>
</tr>
<tr>
<td>428</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>147990</td>
</tr>
<tr>
<td>429</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>150000</td>
</tr>
<tr>
<td>430</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>152520</td>
</tr>
<tr>
<td>431</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>153960</td>
</tr>
<tr>
<td>432</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>154350</td>
</tr>
<tr>
<td>433</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>154500</td>
</tr>
<tr>
<td>434</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>155490</td>
</tr>
<tr>
<td>435</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>156570</td>
</tr>
<tr>
<td>436</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>187560</td>
</tr>
<tr>
<td>437</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>158310</td>
</tr>
<tr>
<td>438</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>158250</td>
</tr>
<tr>
<td>439</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>158910</td>
</tr>
<tr>
<td>440</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>164790</td>
</tr>
<tr>
<td>441</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>120120</td>
</tr>
<tr>
<td>442</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>170880</td>
</tr>
<tr>
<td>443</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>172920</td>
</tr>
<tr>
<td>444</td>
<td>2</td>
<td>16</td>
<td>user1</td>
<td>org1</td>
<td>172800</td>
<td>172860</td>
</tr>
<tr>
<td>445</td>
<td>2</td>
<td>16</td>
<td>user2</td>
<td>org2</td>
<td>172800</td>
<td>172830</td>
</tr>
<tr>
<td>446</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>188310</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>----------------------</td>
<td>------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>447</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>189270</td>
</tr>
<tr>
<td>448</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>189900</td>
</tr>
<tr>
<td>449</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>192630</td>
</tr>
<tr>
<td>450</td>
<td>1</td>
<td>4</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>196770</td>
</tr>
<tr>
<td>451</td>
<td>1</td>
<td>4</td>
<td>user1</td>
<td>org1</td>
<td>720000</td>
<td>205470</td>
</tr>
<tr>
<td>452</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>206910</td>
</tr>
<tr>
<td>453</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>78840</td>
</tr>
<tr>
<td>454</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>117990</td>
</tr>
<tr>
<td>455</td>
<td>1</td>
<td>1</td>
<td>user3</td>
<td>org3</td>
<td>1080000</td>
<td>214620</td>
</tr>
<tr>
<td>456</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>85620</td>
</tr>
<tr>
<td>457</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>118560</td>
</tr>
<tr>
<td>458</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>127410</td>
</tr>
<tr>
<td>459</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>99810</td>
</tr>
<tr>
<td>460</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>102930</td>
</tr>
<tr>
<td>461</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>109140</td>
</tr>
<tr>
<td>462</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>163890</td>
</tr>
<tr>
<td>463</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>141510</td>
</tr>
<tr>
<td>464</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>114060</td>
</tr>
<tr>
<td>465</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>119250</td>
</tr>
<tr>
<td>466</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>99480</td>
</tr>
<tr>
<td>467</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>123360</td>
</tr>
<tr>
<td>468</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>137250</td>
</tr>
<tr>
<td>469</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>142290</td>
</tr>
<tr>
<td>470</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>131010</td>
</tr>
<tr>
<td>471</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>127110</td>
</tr>
<tr>
<td>472</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>118590</td>
</tr>
<tr>
<td>473</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>139320</td>
</tr>
<tr>
<td>474</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>129600</td>
</tr>
<tr>
<td>475</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>163320</td>
</tr>
<tr>
<td>476</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>103530</td>
</tr>
<tr>
<td>477</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>144330</td>
</tr>
<tr>
<td>478</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>135960</td>
</tr>
<tr>
<td>479</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>138360</td>
</tr>
<tr>
<td>480</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>141570</td>
</tr>
<tr>
<td>481</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>144630</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------</td>
<td>----------------------</td>
<td>-------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>482</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>160500</td>
</tr>
<tr>
<td>483</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>720000</td>
<td>269580</td>
</tr>
<tr>
<td>484</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>154830</td>
</tr>
<tr>
<td>485</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>146430</td>
</tr>
<tr>
<td>486</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>148440</td>
</tr>
<tr>
<td>487</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>145260</td>
</tr>
<tr>
<td>488</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>147540</td>
</tr>
<tr>
<td>489</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>152250</td>
</tr>
<tr>
<td>490</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>125280</td>
</tr>
<tr>
<td>491</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>158580</td>
</tr>
<tr>
<td>492</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>126570</td>
</tr>
<tr>
<td>493</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>162690</td>
</tr>
<tr>
<td>494</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>95850</td>
</tr>
<tr>
<td>495</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>166680</td>
</tr>
<tr>
<td>496</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>160200</td>
</tr>
<tr>
<td>497</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>168420</td>
</tr>
<tr>
<td>498</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>168420</td>
</tr>
<tr>
<td>499</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>167250</td>
</tr>
<tr>
<td>500</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>153570</td>
</tr>
<tr>
<td>501</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>150600</td>
</tr>
<tr>
<td>502</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>185160</td>
</tr>
<tr>
<td>503</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>127650</td>
</tr>
<tr>
<td>504</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>150270</td>
</tr>
<tr>
<td>505</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>188130</td>
</tr>
<tr>
<td>506</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>166560</td>
</tr>
<tr>
<td>507</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>175350</td>
</tr>
<tr>
<td>508</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>180960</td>
</tr>
<tr>
<td>509</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>188100</td>
</tr>
<tr>
<td>510</td>
<td>16</td>
<td>128</td>
<td>user2</td>
<td>org2</td>
<td>86400</td>
<td>55590</td>
</tr>
<tr>
<td>511</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>171840</td>
</tr>
<tr>
<td>512</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>175920</td>
</tr>
<tr>
<td>513</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>163230</td>
</tr>
<tr>
<td>514</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>109380</td>
</tr>
<tr>
<td>515</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>188040</td>
</tr>
<tr>
<td>516</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>129300</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>----------------------</td>
<td>--------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>517</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>179790</td>
</tr>
<tr>
<td>518</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>186810</td>
</tr>
<tr>
<td>519</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>212700</td>
</tr>
<tr>
<td>520</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>129120</td>
</tr>
<tr>
<td>521</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>188280</td>
</tr>
<tr>
<td>522</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>106290</td>
</tr>
<tr>
<td>523</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>158490</td>
</tr>
<tr>
<td>524</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>130800</td>
</tr>
<tr>
<td>525</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>36930</td>
</tr>
<tr>
<td>526</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>165450</td>
</tr>
<tr>
<td>527</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>183570</td>
</tr>
<tr>
<td>528</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>68190</td>
</tr>
<tr>
<td>529</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>113130</td>
</tr>
<tr>
<td>530</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>172950</td>
</tr>
<tr>
<td>531</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>110730</td>
</tr>
<tr>
<td>532</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>162600</td>
</tr>
<tr>
<td>533</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>118650</td>
</tr>
<tr>
<td>534</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>182610</td>
</tr>
<tr>
<td>535</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>61530</td>
</tr>
<tr>
<td>536</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>67770</td>
</tr>
<tr>
<td>537</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>156630</td>
</tr>
<tr>
<td>538</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>121440</td>
</tr>
<tr>
<td>539</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>137640</td>
</tr>
<tr>
<td>540</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>181650</td>
</tr>
<tr>
<td>541</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>191160</td>
</tr>
<tr>
<td>542</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>157980</td>
</tr>
<tr>
<td>543</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>140730</td>
</tr>
<tr>
<td>544</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>121470</td>
</tr>
<tr>
<td>545</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>177600</td>
</tr>
<tr>
<td>546</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>163680</td>
</tr>
<tr>
<td>547</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>140220</td>
</tr>
<tr>
<td>548</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>143070</td>
</tr>
<tr>
<td>549</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>144300</td>
</tr>
<tr>
<td>550</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>135420</td>
</tr>
<tr>
<td>551</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>160590</td>
</tr>
<tr>
<td>Job ID</td>
<td>Nodes Requested</td>
<td>Processors Requested</td>
<td>User</td>
<td>Organization</td>
<td>Time Requested (sec)</td>
<td>Actual Running Time (sec)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>----------------------</td>
<td>------</td>
<td>--------------</td>
<td>-----------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>552</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>169140</td>
</tr>
<tr>
<td>553</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>165780</td>
</tr>
<tr>
<td>554</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>145560</td>
</tr>
<tr>
<td>555</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>188070</td>
</tr>
<tr>
<td>556</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>146550</td>
</tr>
<tr>
<td>557</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>170160</td>
</tr>
<tr>
<td>558</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>166020</td>
</tr>
<tr>
<td>559</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>147420</td>
</tr>
<tr>
<td>560</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>116580</td>
</tr>
<tr>
<td>561</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>152250</td>
</tr>
<tr>
<td>562</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>130560</td>
</tr>
<tr>
<td>563</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>150210</td>
</tr>
<tr>
<td>564</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>193380</td>
</tr>
<tr>
<td>565</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>194790</td>
</tr>
<tr>
<td>566</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>194610</td>
</tr>
<tr>
<td>567</td>
<td>1</td>
<td>2</td>
<td>user1</td>
<td>org1</td>
<td>1188000</td>
<td>255870</td>
</tr>
<tr>
<td>568</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>155640</td>
</tr>
<tr>
<td>569</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>154830</td>
</tr>
<tr>
<td>570</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>156240</td>
</tr>
<tr>
<td>571</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>156300</td>
</tr>
<tr>
<td>572</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>197010</td>
</tr>
<tr>
<td>573</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>173580</td>
</tr>
<tr>
<td>574</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>159390</td>
</tr>
<tr>
<td>575</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>191040</td>
</tr>
<tr>
<td>576</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>209340</td>
</tr>
<tr>
<td>577</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>163740</td>
</tr>
<tr>
<td>578</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>221100</td>
</tr>
<tr>
<td>579</td>
<td>1</td>
<td>4</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>168960</td>
</tr>
<tr>
<td>580</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>205830</td>
</tr>
<tr>
<td>581</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>186150</td>
</tr>
<tr>
<td>582</td>
<td>1</td>
<td>1</td>
<td>user2</td>
<td>org2</td>
<td>1080000</td>
<td>202980</td>
</tr>
<tr>
<td>583</td>
<td>1</td>
<td>4</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>189600</td>
</tr>
<tr>
<td>584</td>
<td>1</td>
<td>1</td>
<td>user1</td>
<td>org1</td>
<td>1080000</td>
<td>253290</td>
</tr>
<tr>
<td>585</td>
<td>1</td>
<td>2</td>
<td>user2</td>
<td>org2</td>
<td>1188000</td>
<td>288840</td>
</tr>
</tbody>
</table>
E GRAPHML STANDARD

The GraphML standard defines a format for modeling graphs in an XML format. While this appendix presents a simplified document with an example, the standard (Brandes et al., 2002) and associated documents (Brandes et al., 2005) should be referred to for further information.

E.1 Tags

The following tags are used in a GraphML document:

- <xml ?>
  - Like all XML documents, this definition tag exists at the head of the file, and is used to define the version of the XML standard the document conforms to, and the text encoding.
  - An extremely common example: <xml version="1.0" encoding="UTF-8"/>

- <graphml>
  - This tag is used to enclose a graph or group of graphs, and resides just after the opening <xml ?> tag, and at the end of the file.
  - The opening tag may also include xmlns namespace information, though this is not required. An example of this is seen in Listing E.1.

- <graph>
  - This tag opens and closes each individual graph, and defines the graph’s identity, and the default edge directionality (undirected vs. directed).
  - Example tag pair: <graph id="graphidentity" edgedefault="directed"> ... </graph>

- <node>
This tag is used to create a graph node, with its associated identity. If no extra data is necessary, a single-tag type is used, but if extra data (using the data tag) is used, there will be separate opening and closing tags.

- Example single tag: `<node id="A"/>`
- Example tag pair: `<node id="B">... </node>`

- `<edge>`

  - This tag is used to create a relationship between nodes, including both source and target nodes, and optionally edge identifying information, and directionality information.
  - If the directionality (directed vs. undirected) is not specified, the edge defaults to the directionality of the enclosing graph.
  - If the directionality of an edge is undirected, the terms “source” and “target” are still used, but are considered interchangeable.
  - If the edge has associated data, a pair of opening and closing tags are used; otherwise, a single tag is used.
  - Example single tag: `<edge id="e1" source="A" target="B"/>`
  - Example tag pair: `<edge id="e2" source="B" target="C">... </edge>`

- `<data>`

  - This tag is utilized inside either a pair of node tags, or a pair of edge tags, to associate a name-value pair with that node or edge.
  - Example: `<node id="C"><data key="name">value</data></node>`

### E.2 Example

Figure E.1 and Listing E.1 show the relationship between the in-memory design and the GraphML representation of a graph.

The code segment below demonstrates the GraphML representation of the graph shown in Figure E.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns"
>`
<http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<graph id="organization_name" edgedefault="directed">
  <node id="A">
    <data key="attr">val2</data>
    <data key="attr2">val10</data>
  </node>
  <node id="B">
    <data key="attr">val3</data>
  </node>
  <node id="C" />
  <node id="D">
    <data key="attr">val1</data>
  </node>
  <node id="E">
    <data key="attr">val4</data>
  </node>
  <node id="F" />
  <node id="G" />
  <node id="H" />
  <node id="I" />
  <edge source="A" target="B" />
  <edge source="A" target="C" />
  <edge source="B" target="F" />
  <edge source="C" target="F" />
  <edge source="D" target="A" />
  <edge source="D" target="E" />
  <edge source="E" target="F" />
  <edge source="E" target="G" />
  <edge source="F" target="D" />
  <edge source="G" target="H" />
  <edge source="I" target="G" />
</graph>
</graphml>
Figure E.1: GraphML Example Graph