Jul 1st, 12:00 AM

Environmental modeling in an event-driven multitasking network environment

G. Smiatek

Environmental modeling in an event-driven multitasking network environment

G. Smiatek [a]

[a]Institute for Meteorology and Climate Research (IMK-IFU),
Forschungszentrum Karlsruhe, 82467 Garmisch-Partenkirchen, Germany

Abstract: In order to provide fast analysis and forecast of a future state of a considered system, environmental modeling requires integration of different tools, data and methods. Using the example of the semi-empirical biogenic volatile organic compound emission model (seBVOC) the integration of numerical weather prediction model (NWP) output, Geographical Information System (GIS) and the seBVOC model with an event-driven multitasking network program is demonstrated. It is based on the Perl Object Environment (POE) and includes HTTP data request and download, Web services exposing GIS functionalities and the model run. The system is run in parallel and in a collaborative event-driven way on several computer platforms connected to a network. Compared to the sequential approach a speed-up factor of up to 2 has been reached in the seBVOC application in the NATAIR project.

Keywords: Multitasking, Perl, POE, Web Services, GIS, BVOC, NATAIR

1 INTRODUCTION

Environmental modeling often requires integration and collaboration of different components. Numerical weather prediction models (NWP), Geographical Informations Systems (GIS), Relational Database Management Systems (RDBMS) and other packages and libraries, like NetCDF (network Common Data Form) or SOAP (Simple Object Access Protocol) are examples of such components. Together with data, often stored in various formats on different hosts, they must be linked in an appropriate way in order to provide fast analysis or forecast a future state of the considered environmental system.

An event-driven multitasking network program application is discussed using the example of the semi empirical biogenic volatile organic compound emission model (seBVOC). It integrates NWP output, GIS and the seBVOC model. The application is based on the Perl Object Environment (POE) and includes HTTP(Hypertext Transfer Protocol) data request and download, SOAP-based Web services exposing GIS functionalities and the model run. It can be run in a distributed mode on various computer platforms connected to a network.

This article describes in broad terms the constraints of the seBVOC application within the EU FP6 (Sixth Framework Programme) NATAIR (Improving and Applying Methods for the Calculation of Natural and Biogenic Emissions and Assessment of Impacts on Air Quality) project (http://natair.ier.uni-stuttgart.de). It also discusses the set up of a POE-based multitasking system and the application of Web services.

2 BVOC MODELING WITHIN NATAIR

Biogenic volatile organic compounds (BVOC) play an important role in the chemistry of the atmosphere. Together with anthropogenic NOx and VOC emissions they contribute to regional and global changes in the HO-radical budget, ozone as well as particle distributions. Therefore, BVOC emission inventories are required for any air quality and climate forcing studies.

The semi-empirical BVOC emission model seBVOC [Smiatek and Steinbrecher, 2006] is based on the algorithm presented by Guenther et al. [1993] and Shao et al. [2001] and is discussed in detail by Stewart et al. [2003]. The BVOC emission is dependending on the emitting land cover represented by foliar biomass and standardized emission rate, air temperature and solar radiation. In the model the light extinction within the canopy is taken into

account with a sunlit/shade model presented by de Pury and Farquhar [1997]. Leaf temperatures are determined from canopy energy budget as proposed by Dai et al. [2004]. seBVOC is written in Fortran and is fully parallelized following the distributed memory parallel (DMP) model approach relying on the MPI (message-passing-interface; Pacheco and Ming [1997]) library. seBVOC can be run with meteorology data (temperature, solar radiation, wind speed and humidity) provided by the MM5 model [Dudhia, 1993] and stored in the MM5V3 version binary data format. In addition, in can employ ASCII files stored in ArcGIS [ESRI, 2006] grid format.

The seBVOC model is integrated in a Geographical Information System (GIS) and a Relational Data Base Management (RDBMS) environment of input data processing, model running, and visualization. It consist of three parts: GIS/RDBMS used for provision of time-invariant data (land cover, emission rates, foliar biomass densities) and visualization, a suite of Perl programs facilitating the data processing and the seBVOC model run.

Within NATAIR the seBVOC model is used to estimate BVOC emissions in hourly temporal resolution and in 10 km x 10 km spatial resolution for the years 1997, 2000 and with future climate data for the year 2010. The meteorological data is provided as daily files and stored in NetCDF data format. Here, the hourly fields have a spatial resolution of 24 km by 24 km. An additional difficulty is that the BVOC inventory has to be provided in Lambert azimuthal projection defined by NATAIR, whilst the meteorological input is in Lambert conformal conic map projection. A change of map projection change and interpolation to the 10 km x 10 km grid has thus to be applied to the data.

Figure 1 shows the activity and responsibility diagram of the modeling system which includes the host running the entire system, GIS, data download and the seBVOC model. The model can be run on a single processor or in parallel mode on a cluster computer. The land cover data, emission rates as well as the foliar biomass densities are time invariant and are calculated at the beginning of the program. The data download, preprocessing by the GIS and model run are performed in a loop of daily time steps.

In order to speed up the seBVOC application within NATAIR the entire system is run in a distributed event driven multitasking approach.

3 EVENT DRIVEN MULTITASKING SYSTEM

3.1 POE

Perl object environment (POE) is a free and open powerful multitasking framework. It allows applications to be distributed across a network of machines and perform several tasks at once [Caputo, 2003]. POE provides three levels of abstraction enabling developers to rapidly create and deploy applications tailored to their needs. The POE environment consists of states, the kernel and sessions. States are routines that are executed when some specified events occur. The kernel runs the entire system. A POE session handles the state, sends POE messages, creates child session, etc. POE comes with a steadily growing number of drivers, filters, wheels and components [Taylor and Goff, 2001] and is available from CPAN (Comprehensive Perl Archive Network) (www.cpan.org).

3.2 System setup

The distributed seBVOC model run within NATAIR is performed with a task system. For simplicity only three tasks are considered here: *http*, *gis* and *model* (see Figure 2). The task *http* performs the download of the meteorology files in netCDF format from the foreign NATAIR host. Task *gis* is responsible for the unloading of data arrays from the NetCDF file, import into an ArcGIS GRID file, interpolation, projection to Lambert Conformal Conic and exports to ASCII grid. The GIS-based part is run on a foreign host hosting the ArcGIS software. Finally, the task *model* performs the run of the seBVOC either on a single foreign host or a computer cluster. In the parallel approach *http* will download the input for the day d, while task *gis* handles the data of day $d-1$ and task *model* runs the model for the day $d-2$. All write operations are preformed to the same file system mounted over NFS (Network File System), thus, an additional data exchange over a network is not required.

Figure 3 shows the set-up of the main POE session. It needs only a few lines of code and defines the events _start, next_task, task_info, task_done, task_error and _stop and starts the event processing. The events _start and _stop are native POE states. _start is always called when a POE Session is created, while _stop will terminate the session. The session will run in an infinite loop until the event _stop occurs putting the session into the _stop state. The events itself are handled by calling the &start_task, &handle_done, &handle_info and

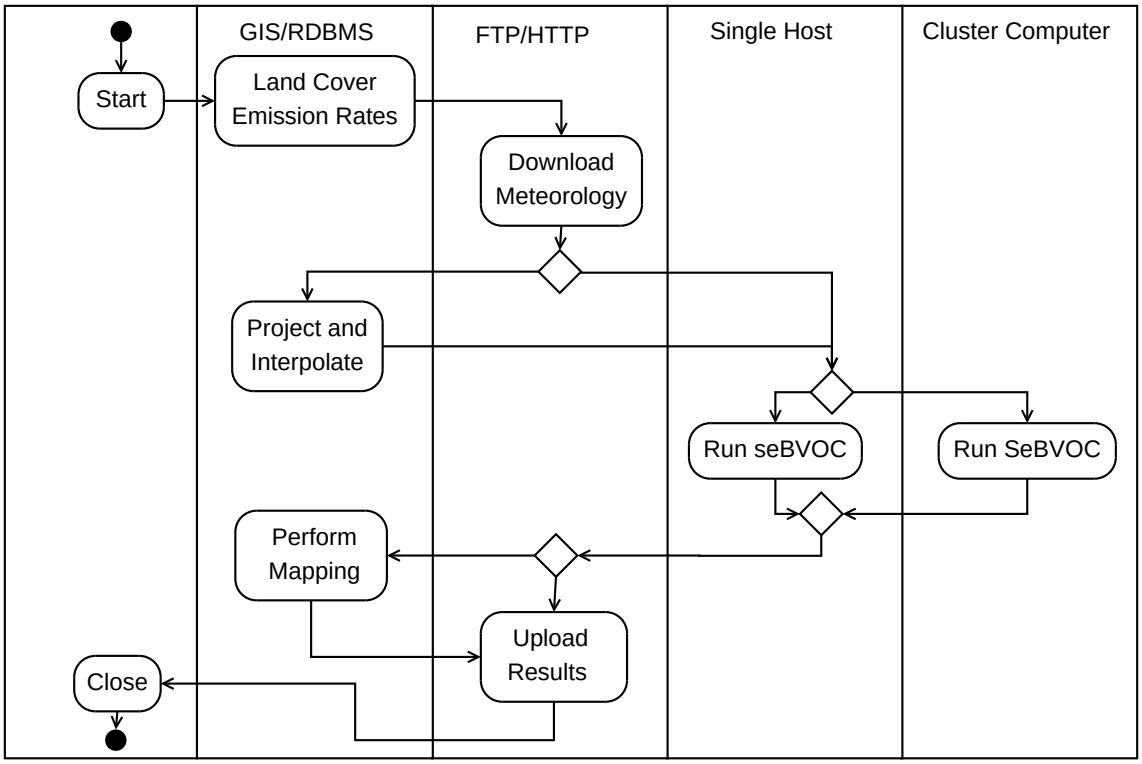

Figure 1: Activity/Responsibility diagram in seBVOC application

&handle_error functions.

The function &start_task is as simple as the session. It keeps track of the dates to calculate and reads the content of an array whose elements are the task names to be performed. It loops over those elements, submits the task and removes the task name from the array. Thus, when the system starts there will be only one task *http* in the array. Once this task is finished two tasks will be put into the array: *http* handling the day d and *gis* handling the day $d - 1$. After this loop the array will consist of all three items until the last day is reached. First the *http* task will be skipped, and then the *gis* task. Finally, only the task running the model for the last day is started. The event task_done of that task identified by a *task_id* is used to invoke the _stop event ending the session.

In POE a hash called the *heap*, available in every state of the session, is used to store session specific data. The task id and the task name are stored in the heap variable.

The key question is how the tasks are started in a non-blocking way? Here the POE:Wheel part of POE provides appropriate functionality. Wheels are event handlers (states) that encapsulate some program code to perform common tasks. Each wheel must be created by a session, and each belongs to its parent session. Again several states are defined (see Figure 4). The state *Program* is the function that performs the real work. It takes as argument the current date stored in the $DATE array. Additional states are: *StdoutEvent* used to transport information when the task ends, *StderrEvent* to catch any error and *CloseEvent* issued when the task finished successfully. I should be noted that the states issue events *task_info*, *task_error* and *task_done* are defined in the parent session.

Once the program function is started, the wheel object will immediately return the control to the POE session and, thus, the next task can be started without being blocked by a previous task.

Also in the wheel, a POE::Filter is used to transport some from the tasks. Filters are useful for converting data from one format to another. Another practical Perl module is Date::Calc, available from CPAN. It provides a large range of functions for date calculations based on the Gregorian calendar.

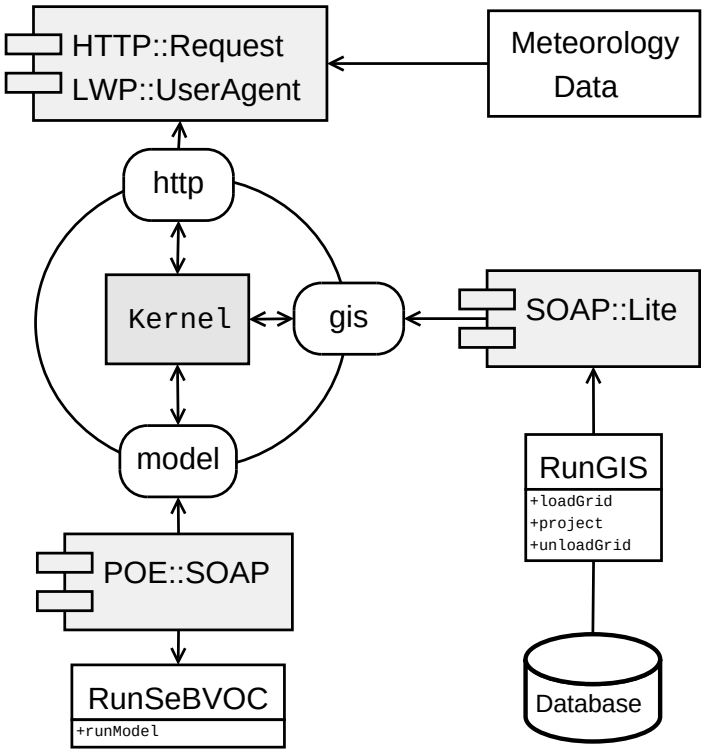

Figure 2: POE kernel with wheel-based tasks and components used to download data, set up and access Web services

```
1 use POE;
2 POE::Session->create
3 (inline_states =>
4  {_start     => \&start_task,
5   next_task  => \&start_task,
6   task_result=> \&handle_result,
7   task_info  => \&handle_info,
8   task_error => \&handle_error,
9   _stop      => sub{print "Shutdown"},
10  }
11 )
12  $poe_kernel->run();
13  exit;
```

Figure 3: Definition of a POE session

```
1$task = POE::Wheel::Run->new
2 ( Program      => sub
3     { do_gis($DATES[$dd]) },
4   StdoutFilter =>
5     POE::Filter::Reference->new(),
6   StdoutEvent  => "task_info",
7   StderrEvent  => "task_error",
8   CloseEvent   => "task_done",
9 );
```

Figure 4: Definition of a POE Wheel

3.3 Input data download

The download of the NetCDF meteorological input files from the foreign host is facilitated with a client based on the HTTP::Request and LWP::UserAgent modules from the libwww-perl library available from CPAN. This library is a set of Perl modules which provide classes and functions forming a simple programming interface to the World-Wide Web. Figure 5 shows that the file download is performed using a few lines of Perl code. $ua is a LWP::UserAgent object. LWP::UserAgent configures it with values for timeouts, proxies, name, and others. Then, an instance of HTTP::Request for the request of the file specified in the $urlFile file variable is created. The object $response contains the NetCDF file and many other information, that can be used to handle any communication errors. The NetCDF file is saved to the disk.

There is also a POE::Component::Client::HTTP component available that might be used for the same purpose. This is useful in case the client has to communicate with several servers.

3.4 Web services

The access to some functions of the GIS and to the seBVOC model run are implemented as Web services based on the Simple Object Access Protocol. SOAP is a protocol specification for invoking methods on servers, services, components and objects using XML (Extensible Markup Language) and HTTP as the method invocation mechanism. Its specification was published in 2000 [Box et al., 2000].

In the present approach, the *gis* task will employ a Web service exposed on a foreign host. Both server and client functions use the SOAP::Lite Perl library. The word "Lite" in the name refers to the simple interface it provides, not to its capabilities. A short SOAP::Lite tutorial is given by Kulchenko [2001].

Within the *gis* data arrays of a given time step and meteorological variables are extracted from the binary NetCDF file fist. Here, the Perl PDL-NetCDF library available from CPAN provides an appropriate access function to the various data types defined by NetCDF. These files are transformed into ASCII ArcGIS GRID format.

The server will dispatch all requests to a Perl class called *RunGIS* which provides methods for data import, map projection change, interpolation and export. The server set up and application of the SOAP::Lite library in a GIS environment is discussed in detail by Smiatek [2005]. Figure 6 shows the simple client definition. For specified $SOAP_URL and $SOAP_PROXY, which contains the address of the host and the communication port, the client will invoke the method *projectGrid* on the foreign host which exposes the Web service.

The implementation of the seBVOC model run is provided by the *RunSEBVOC* Perl class exposed as a Web Service through POE::Component::Server::SOAP component. This component publishes POE event handlers via SOAP over HTTP. Figure 7 illustrates the client definition. It is even simpler than the native SOPE::Lite client. If provided with the URL and the port, the client will invoke the method *runModel* on the foreign host. Again it will communicate any errors via *task_error* state.

4 RESULTS

The system outlined in paragraph 3 creates the frame in which the distributed run of the seBVOC is performed. The application has been run for a

```
1 use HTTP::Request;
2 use LWP::UserAgent;
3 $request  = HTTP::Request->new(
4             GET => $urlFile);
5 $ua       = LWP::UserAgent->new;
6 $response = $ua->request($request);
```

Figure 5: WWW client definition

```
1 use SOAP::Lite;
2 $client = SOAP::Lite->uri($SOAP_URL);
3 $client->proxy($SOAP_PROXY);
4 $arc = $client
5       ->call(new =>)
6       ->result;
7 $resp = $client->projectGrid($arc,
8         $inGrid,$outGridGis);
```

Figure 6: Definition of a client accessing a Web service which implements changing the geographical map projection.

```
1   SOAP::Lite
2 -> uri('http://localhost:32080/')
3 -> proxy('http://localhost:32080/
5                 ?session=MyServer')
6 -> run_seBVOC($date)
7 -> result
8 ;
```

Figure 7: POE SOAP client definition

10 day episode in June 2000. The hourly BVOC emissions have been calculated in 10 km x 10 km resolution for the entire NATAIR area consisting of 519 columns and 544 rows. In a sequential run the download of a single NetCDF File (one day) required about 300 s. The GIS run was more expensive with approximately 2000 s. While the calculation of the BVOC emission with seBVOC took 1200 s. Thus, an event-driven sequential 10 day run will require almost 10 hours. In the distributed multitasking approach this time has been reduced to less then 6 hours. It is clear that the GIS part is the limiting resource of the system. Running the GIS application on a faster computer host could increase the speed of the system. On the other hand in the present approach it was sufficient to run the seBVOC model on a single processor instead of a cluster computer.

The POE-based system has been run several times on various hosts and under different network conditions showing high stability and excellent performance.

5 CONCLUSIONS

The Perl POE toolkit provides an excellent means to create and deploy powerful cooperative multitasking applications in Perl. POE comes with a large number of contributed components, tutorials and example applications making it extremely easy to use in any environment. There is of course, more programming needed to implement the model, run of the GIS functions, especially the interpolation routines or error exception handling. But these parts are the genuine domain of the environmental modelers. Thanks to POE, programming the Web communication, and Web services, the multitasking environment can be reduced to a minimum.

ACKNOWLEDGMENTS

This article has been prepared in part as a contribution to the NATAIR, an FP6 activity funded by the European Commission, contract no. 513669

REFERENCES

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H., Thatte, S., Winer, D., 2000. Simple Object Access Protocol (SOAP) 1.1., www.w3.org/TR/SOAP/.

Caputo, R. (2003): POE: The Perl object environment. http.//www.perl.org/poedown/poe-whitepaper-a4.pdf

Dai, Y., Dickinson, R.E., Wang, Y.-P., 2004. A two-big-leaf model for canopy temperature, photosynthesis, and stomatal conductance. Journal of Climate, 17, (12), pp. 2281–2299.

de Pury, D.G.G., Farquhar, G.D., 1997. Simple scaling of photosynthesis from leaves to canopies without the errors of big-leaf models. Plant, Cell and Environment, 20, 537–557.

Dudhia, J.(1993): A nonhydrostatic version of the Penn State-NCAR mesoscale model: Validation tests and simulation of an atlantic cyclone and cold front, *Monthly Weather Review, 121*

ESRI(2006): What Is ArcGIS?. http://www.esri.com/software/arcgis/about/overview.html

Guenther, A., P. Zimmerman, P. Harley, R. Monson, and R. Fall, 1993. Isoprene and monoterpene emission rate variability: Model evaluation and sensitivity analysis, *J. Geophys. Res., 98*, 12609–12617.

Kulchenko, P., 2001. SOAP::Lite for Perl, www.soaplite.com.

Pacheco, P. S., and W. Ch. Ming, 1997. Introduction to message passing programming. MPI userś guide in Fortran. 55 pp.

Shao, M., K. V. Czapiewski, A. C. Heiden, K. Kobel, M. Komenda, R. Koppmann, and J. Wildt, 2001. Volatile organic compound emissions from Scots pine: Mechanisms and description by algorithms, *J. Geophys. Res., 106*(D17), 20483-20492, doi:10.1029/2000JD000248.

Smiatek, G., 2005. SOAP-based Web Services in GIS/RDBMS environment. Environmental Modelling & Software, doi:10.1016/j.envsoft.2004.04.008

Smiatek, G. and R. Steinbrecher (2006): Temporal and spatial variation of forest VOC emissions in Germany in the decade 1994 - 2003. Atmospheric environment (in print)

Stewart E. H., Hewitt, C.N., Bunce, R.G.H, Steinbrecher, R., Smiatek G., Schoenemeyer, T., 2003. A highly spatially and temporally resolved inventory for biogenic isoprene and monoterpene emissions - model description and application to Great Britain. Journal Geophysical Research 108(D20), 4644, doi: 10.1029/2002JD002694.

Taylor D. and J. Goff (2001): A Beginner's Introduction to POE.