

Planning for Distributed Earth Science Data Processing

Petr Votava^{1,2}, Keith Golden², Ramakrishna Nemani²

¹California State University Monterey Bay, Seaside, CA 93955, USA

²NASA Ames Research Center, Moffett Field, CA 94035, USA
votava@ltpmail.gsfc.nasa.gov, {keith.golden, rama.nemani}@nasa.gov

Abstract

An important challenge in Earth science processing is the large volume and distributed nature of the data required by many processing algorithms. Despite the increase in available bandwidth over the last several years, it is still often impractical, or at least very time-consuming, to acquire and locally stage the data prior to processing, because the volumes can run into tens or even hundreds of Gigabytes per day. In this paper, we describe a distributed system for Earth science data processing in which the control and execution of data-processing procedures are decoupled, allowing execution of independent data-processing components to be controlled remotely over the network. We use Web Services to export the interface of these components. Finally, we describe a planner-based agent that operates on the exposed Web Service components to automatically generate and execute data-flow programs to produce the requested products. We can then build entire processing pipelines that include pre-processing, processing and analysis of the results in an automated and efficient way, simplifying the development of distributed Earth science processing systems. We show an application of our system in the Terrestrial Observation and Prediction System (TOPS) whose goal is to provide a daily monitoring and prediction of numerous biospheric variables that are important indicators of the events happening within the Earth system. Finally, we show an application of our system to near-real-time fire monitoring and analysis.

1. Introduction

The management and efficient processing of Earth science data has been gaining an importance over the last decade, due to higher data volumes generated by a large number of satellites and ground stations and to the increase in complexity of Earth science models that use these data sets. This rapid growth is driven by the fact that in order to understand and predict the processes within the Earth, we need a global data set spanning many years and possibly decades. Additionally, in order to take full advantage of the vast amount of data being produced, we want the ability to seamlessly merge data from different sources within the models.

With the launch of NASA's Terra and Aqua missions during the last 3 years, the need for more efficient, scalable, and intelligent data processing systems is even more apparent. In this paper, we address many of the issues facing the Earth science community as we present our distributed intelligent framework, which automates access, transport, translation, distributed processing and analysis of Earth science data. Our system consists of two main components – the Java Distributed Application Framework (JDAF) (Votava, 2002) and a planner-based agent called IMAGEbot (Golden, 2002). JDAF provides the low-level components for data access, processing and analysis, and IMAGEbot provides the intelligence to connect the individual components, synthesizing processing systems that adhere to specific requirements and constraints. JDAF and IMAGEbot interact with each other through a set of interfaces exposed through Java RMI (Pitt, 2001) servers and Web services.

We will describe two applications of our technology – the Terrestrial Observation and Prediction System (TOPS) (Nemani, 2002) and the fire-monitoring project. The main goal of the TOPS system is to provide both nowcast and forecast of a number of biospheric variables. Early warning of potential changes in these variables, such as soil moisture, snow pack, or primary production, could enhance our ability to make better socio-economic decisions relating to natural resource management and food production (Nemani, 2000). The fire-monitoring project provides near-real-time monitoring of reported fires and performs analysis with respect to current conditions and historic trends for that particular location.

The remainder of the paper is organized as follows: in section 2, we introduce some of the challenges faced by the Earth science community and discuss data processing as an AI planning problem; in section 3, we describe both JDAF and IMAGEbot systems and their integration; in section 4, we discuss the applications of our technology. Finally, in section 5 we describe our plans and related work.

2. Background

Earth Science

The latest generation of NASA Earth Observing System (EOS) (King, 1999) satellites has brought a new dimension to continuous monitoring of the living part of the Earth system, the biosphere. EOS data can now provide weekly global measures of vegetation productivity, ocean chlorophyll, and many related biophysical factors such as land cover changes or snowmelt rates. However, information with the highest economic value would be forecasting impending conditions of the biosphere that would allow advanced decision-making to mitigate dangers or exploit positive trends. NASA's strategic plan for the Earth Science Enterprise identifies ecological forecasting as a focus for future research. Ecological forecasting predicts the effects of changes in the physical, chemical and biological environment on ecosystem activity. Imagine if we could accurately predict shortfalls or bumper crops of agricultural production, or West Nile virus epidemics or wildfire danger 3-6 months in advance, allowing improved preparation and logistical efficiency. The climate forecasting accuracy of many coupled Ocean-Atmosphere general circulation models (GCM) (McGuffie, 1997) have steadily improved over the past decade. Given observed anomalies in sea-surface temperatures from satellite data, GCMs are able to forecast general climatic conditions 6-12 months into the future, trends of hotter/colder temperatures and wetter/drier precipitation than normal, with reasonable accuracy. While such

climatic forecasts are useful alone, the advances in ecosystem modeling allow us to explore specifically the impacts of these future climate trends on the ecosystem directly.

One of the key problems in adapting climate forecasts to natural ecosystems is the 'memory' that these systems carry from one season to the next (e.g. soil moisture, plant seed banks, fire fuel accumulation etc.). Simulation models are often the best tools to carry forward the spatiotemporal 'memory' information. The power of models that can describe and predict ecosystem behavior has advanced dramatically over the last two decades, driven by major improvements in process-level understanding, computing technology, and the availability of a wide-range of satellite- and ground-based sensors.

Data Processing

Many Earth science processing systems are driven by large numbers of scripts performing most of the scheduling and processing setups. Despite some advantages of this approach, mainly its rapid development, there are many drawbacks, including difficulties in maintainability, scalability to a larger number of datasets and processes, and flexibility in accommodating new processes and data streams in the existing system. Some of the issues stem from the nature of the scripts and their lack of language-level support to accommodate the translation of the design into the implementation. Other issues relate to the nature of the systems that are developed in this way - they are fast prototypes that often stay around as the only implementation of the system design.

Apart from the lack of flexible and extensible processing framework, one of the main problems in current Earth science systems is a lack of common metadata standards. This makes dealing with large volumes of data very difficult in terms of data fusion and overall system flexibility. Additionally, the data come in many different formats (HDF, HDF-EOS, ASCII, GRIB, binary, and many others), projections, and quality, which can further complicate handling of multiple data streams.

Due to the inflexibility of many current Earth science processing systems, there is often a long time lag between determining a need for a new capability and actually implementing this need in the production. Our Java Distributed Application Framework (JDAF) in combination with the IMAGEbot planner-based agent adds flexibility that will significantly cut the time required to support new capabilities, whether we need to add a new data stream, produce a new data product, add a new model, or create a new application.

Planning

Data processing has traditionally been automated by writing shell scripts. There are some situations when scripts are the best approach: namely, when the same procedure is to be applied repeatedly on different inputs, the environment is stable and there are few choices to be made. However, in many applications none of these assumptions holds. There are many different data products we would like the system to produce, there are many inputs and data-processing operations to choose from in producing those products, and the availability of these inputs can change over time. Additionally, many of these applications lend themselves well to planner-based automation; they have precisely defined inputs, outputs, and operations whose effects can

be precisely characterized. However, there are significant differences between the domain of Earth science data processing and more traditional planning domains, which calls for different techniques. Notable features of data processing domains include large dynamic universes, large plans, incomplete information and uncertainty.

As discussed earlier, there is a large number of data sources that we can choose from, which are applicable under different circumstances. Data sources include several satellites, ground stations, and outputs from other models, forecasts and simulations. In addition to input choices, we also have several choices of models to use with the data. As with the data, the models produce results of various quality, resolution, and geographic extent. Moreover, there may sometimes be significant trade-offs in performance versus precision. An FPAR/LAI algorithm provides a good example of this tradeoff. We can produce an FPAR/LAI pixel using either a lookup table, in $O(1)$ time, or a radiative transfer method, in $O(n \log n)$ time (Knyazikhin, 1999). The radiative transfer method provides better results, but can take substantially more time. Depending on whether time or accuracy is more important, either method may be preferred. Another reason for using different models at different times is their possible regional character. Some models are highly specialized and provide very good and precise results in only certain parts of the world. This is partially due to the fact that the scientists who develop these models have a great deal of knowledge about specific geographic area (Pacific Northwest, the Amazons, etc.). They have collected large amounts of local data over the years, and were able to develop models whose outputs are highly accurate in these regions. We usually don't want to use these models when we are concerned with global monitoring, but they are useful when we have identified an important event occurring at the region where we have a very accurate regional model.

3. System Design

Java Distributed Application Framework (JDAF)

One of the main goals of this framework is flexibility – we wanted to be able to add new components into the system with minimum integration efforts and make them produce results in a reasonable amount of time. One of the problems with many Earth science algorithms is that they comprise tens of thousands of lines of legacy code written in C, Fortran and C++, and it would take substantial efforts to re-write all of these algorithms in a way more suitable for integration. Instead, we have decided to write a set of simple wrappers in Java that would provide an interface between our system and the legacy code. These wrappers are subsequently used for interaction between the Java framework and the processing algorithms using the Java Native Interface (JNI) (Gordon, 1998). However, adding new processing algorithm to the system is only part of the solution; we would also like to integrate new data streams without changing the legacy code. This part is much harder, because the I/O components of the existing algorithms are often almost inseparable from the core science processing. We instead deploy a set of “filters” that preprocess the new data on the fly into a format expected by the processing components. This task is greatly simplified due to the Earth Science Markup Language (ESML) (Ramachandran, 2001) - an XML-based language that provides mechanisms for reading scientific data sets in many formats only by changing their external descriptions stored in XML files, and without the need to modify existing I/O code.

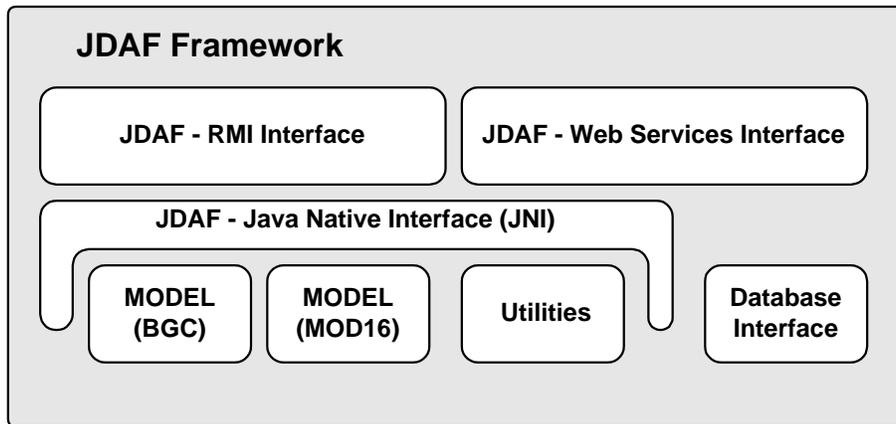


Figure 1: Key components of the JDAF framework

The key components of the JDAF framework are depicted in Figure 1. At the lowest level, there is a set of models and utilities. The number of models in the framework depends on the particular application. Models in our framework refer to complex systems that describe some aspect of the real world in terms of mathematical equations. Low-level utilities consist of functions for data access, extraction, translation and transport. For example, utilities can include data download, composition, or reprojection. Since most of the models and utilities are written in C or C++, we do not access them directly, but rather through wrappers exported into the JNI layer. In order to keep track of the data, models and transformations, we use a three-tier database system that translates the applications' requests into SQL queries and translates the query results into the objects needed by the applications. While our current implementation uses Postgres (Momjian, 2001), the design is flexible enough to accommodate most other implementations by loading the JDBC (Hamilton, 1998) database drivers at run-time. Finally, the models, utility functions and database interfaces are exported in two different ways. First, we use the remote method invocation (RMI) system provided by Java – this has historically been our main implementation strategy. However, with the recent development and improvements of the Web Services technology, we have migrated most of our interfaces in that direction. An important feature of exporting our interfaces through Web Services is ease of distribution and flexibility. We publish our services using WSDL (Chappell, 2002) and so we don't have to distribute source code or binary executables. There are number of tools that convert WSDL descriptions to present the client with a high-level access to our published services. We are using Apache Tomcat (Brittain, 2003) and Apache Axis (Laurie, 2002) as the Web Service engine.

IMAGEbot

IMAGEbot is a planner-based agent that automatically generates and executes data-flow programs in response to user specified goals. The goals may vary with prospective customers of the system, who may be scientists, farmers, fire fighters, or emergency response teams. With

such a variety of customers, there is a strong need for flexible mechanisms for producing the desired data products, taking into account the information needs of the customer, data availability, deadlines, resource usage, and constraints based on context. IMAGEbot provides such a mechanism, accepting goals in the form of descriptions of the desired data products.

In order to describe the complex data structures, constraints and programs that are the elements of the data processing domain, we have developed the Data Processing Action Description Language (DPADL) (Golden, 2003). DPADL is an expressive, declarative language with Java-like syntax, which allows for arbitrary constraints and embedded Java code. The constraint-based planner subsystem of IMAGEbot uses the DPADL action descriptions to synthesize data-flow programs based on a goal in the form of data description. The constraint solver can handle numeric and symbolic constraints, as well as constraints over strings and even arbitrary Java objects. The latter are evaluated by executing code embedded in the constraint definition, specified in the DPADL input file. Additionally, it can solve a limited class of universally qualified constraints (Golden, 2002).

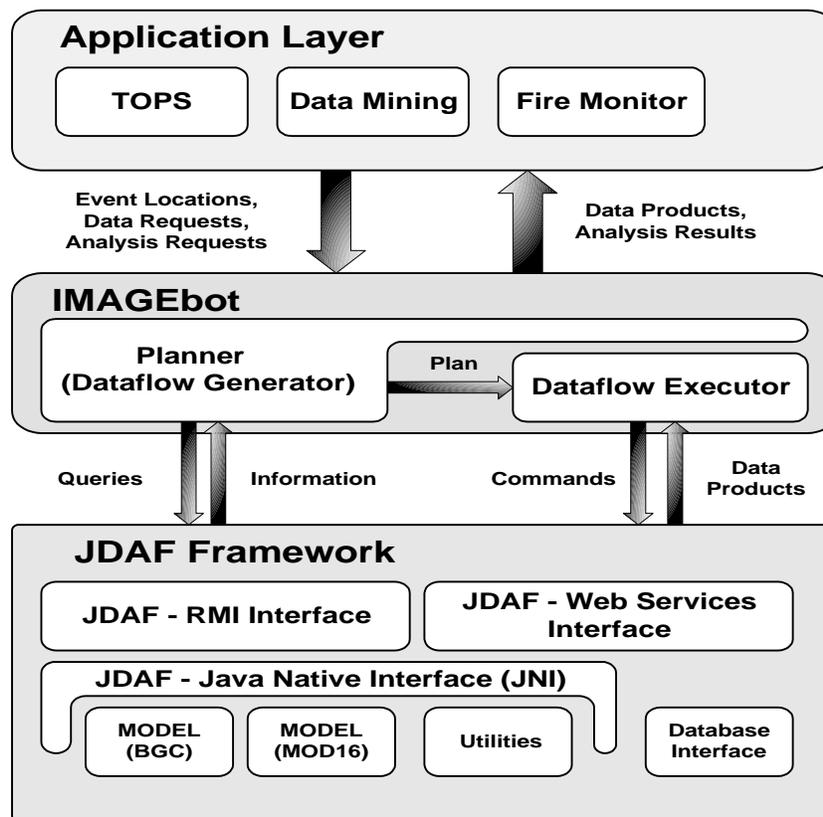


Figure 2: Integrated JDAF, IMAGEbot and applications

Integration

While both JDAF and IMAGEbot are separate components that can be deployed independently of each other to solve variety of different problems, the power of the planner combined with the

flexibility of the framework provides us with a system that is much more robust, flexible and efficient. It can be quickly tailored to vast variety of users and applications, ranging from global long-term monitoring of environmental change to near-real-time analysis of extreme events such as floods, fires, and droughts. From the JDAF point of view, the integration is handled through the Web Services and RMI interfaces exposed to the planner. The planner then needs additional capability – it is not sufficient to only generate plans, but it is necessary to execute them. Thus, there has to be a way to describe how to execute the operations provided by the environment and obtain information about the environment. DPADL provides this capability by permitting embedded Java code in definitions of new constraints and methods for executing actions. Variables used in planning and constraint reasoning subsystems can then reference Java objects as well as primitives such as integers and strings, so fine-grained interaction with the Java runtime environment is possible. Figure 2 depicts the system comprising IMAGEbot, JDAF, and sample applications.

4. Applications

Terrestrial Observation and Prediction System (TOPS)

In order to estimate possible future states of the biosphere, we are building a flexible system that integrates ecosystem models with frequent satellite observations and can be forced by weather or climate forecasts and downscaled to resolutions appropriate to resolve surface processes. The Terrestrial Observation and Prediction System (TOPS) is a modeling software system that automatically integrates and pre-processes EOS data fields so that land surface models can be run in near-real-time with minimal intervention, facilitating accurate and timely interpretation of EOS data. Such a system allows us to determine the vulnerabilities of different socio-economic and resource systems to fluctuations within our biosphere, and helps in mitigating potential negative impacts. The goal of TOPS is to monitor and predict changes in key environmental variables. Early warnings of potential changes in these variables, such as soil moisture, snow pack, primary production and stream flow, could enhance our ability to make better socio-economic decisions relating to natural resource management and food production. The accuracy of such warnings depends on how well the past, present, and future conditions of the ecosystem are characterized. The inputs needed by TOPS include:

- Fractional Photosynthetically Active Radiation (FPAR) and Leaf Area Index (LAI)
- Temperatures (minimum, maximum, and daylight average)
- Precipitation
- Solar Radiation
- Humidity (vpd)

We have several potential candidate inputs at the beginning of each model run. The basic properties of the inputs are listed in Table 1.

The first step in TOPS processing is the selection of the inputs. One thing to note is that a criterion in selection may also be availability, because some inputs are not always available. For example, both the Terra and Aqua satellites experienced technical difficulties and data dropouts over periods ranging from few hours to several weeks. Before the selection process can begin, we have to get the data into some common format, so that the dataset comparison is possible. In the case of TOPS, this comparison is done with gridded data, so we have to make sure that we

Source	Variables	Frequency	Resolution	Coverage
Terra-MODIS	FPAR/LAI	1 day	1km, 500m, 250m	Global
Aqua-MODIS	FPAR/LAI	1 day	1km, 500m, 250m	Global
AVHRR	FPAR/LAI	10 day	1km	Global
SeaWIFS	FPAR/LAI	1 day	1km x 4km	Global
DAO	Temp, precip, rad, vpd	8 hours	1.25 deg x 1.0 deg	Global
RUC2	Temp, precip, rad, vpd	1 hour	40km	USA
RUC2	Temp, precip, rad, vpd	1 hour	20km	USA
CPC	Temp, precip	1 day	Point data	USA
Snotel	Temp, precip	1 day	Point data	USA
NEXRAD	Precip	1 day	4 km	USA
GCIP	Radiation	1 day	0.5 deg	Continental

Table 1: TOPS input data choices

convert the point data (CPC and Snotel) to grid data, which by itself is fairly complex and time-consuming process. Next, the data are selected based on the goal, which at the start of the process is just the status of the variables we are interested in (primary productivity, soil water content, . . .) over the continental US. After the data are selected, we must put them into common format, which may involve reprojecting them into a common projection, subset the dataset from its original spatial extent, and populate the input grid used by the model. The data are then ran through the BGC (Hunt, 1992) model, which generates desired outputs. What follows is a new step in many Earth science systems: the data are compared against long term records and statistics, and the system determines whether there is something important happening in the covered area. An example of such event may be a large difference from a long-term normal for one of the output variables. Whatever the “interesting” event is, the system tries to investigate it further, and one way of accomplishing this is by getting a higher resolution information and going through the input selection process again. The goal has now changed, in terms of both detail and geographic extent, because we no longer need to run the model over the entire continent, but only over several selected areas. Furthermore, we would like more detailed information, so we may actually choose to run a more complex model that runs longer but provides us with higher quality information on the ongoing events, together with the prognosis for near future. As we can see, when this feedback loop is added to TOPS, the complexity of the system goes up even further. TOPS provides only a simple illustration of the potential problems, and is by far not as complex as many other models and systems in the Earth sciences, some of which take dozens of different inputs, with sizes reaching into terabytes for each model run.

Fire Monitor

Another application of our system is the Fire Monitor project. The front end to the system is a software agent that collects information about currently burning fires from multiple sources on the Internet. We can query the agent in regular intervals to supply us with the location (in terms of latitude and longitude), description (name of a place), size, and reported date of the fire. Our monitoring system then obtains the latest available data, which in some cases lag only few hours behind real-time. We extract the region information from the data based on the location, execute

a set of models on this subset to acquire fire-relevant variables (soil moisture, gpp, ...) and perform analysis of the results. The analysis looks at the recent history at this particular location in terms of the key variables like temperature, soil moisture, precipitation and so on, and this data is then matched against the historic records over the same area to determine any significant deviations. The comparisons are then plotted against each other in plots like the one in Figure 3.

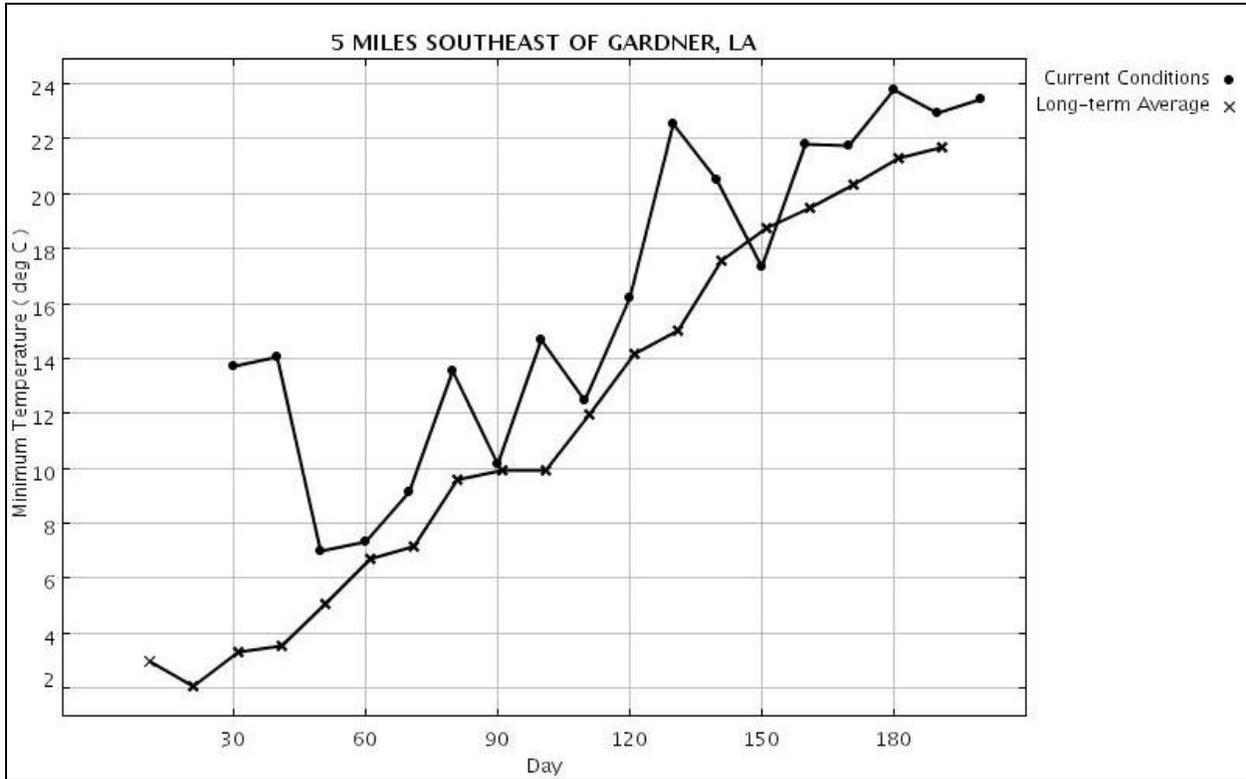


Figure 3: Current and past conditions in a location of reported fire

5. Conclusions and Future Work

We have introduced two main components of our data processing framework – the Java Distributed Application Framework (JDAF) and the planner-based agent IMAGEbot. The integrated system can be used for a rapid development of intelligent Earth science applications.

We have discussed a novel class of planning domains, data processing domains, that pose a number of new challenges for planners. In answer to these challenges, we have developed the DPADL language, which can represent complex, nested data structures, arbitrary constraints, and object creation. One challenge that we have not yet adequately addressed is the multi-criteria optimization problem inherent in the tradeoffs among features such as time, resource consumption and data quality.

The comparison of the system developed using our application framework with the planner to our baseline script-driven system shows improvements in both performance and flexibility. While it can take days and even weeks to integrate a new module into our script-

oriented system, it took less than one hour to integrate a complete MODIS (Justice, 1998) production module using our framework.

Our TOPS and Fire Monitor systems produce data on continuous basis, but there are many improvements that we are currently preparing. The new additions include planner optimization, automatic wrapper generation for and easy drag-and-drop of new modules and data sources into our system, and the addition of a natural language interface (NLI) so that users can query the system by asking questions like “What is the fire danger for Western Montana for tomorrow?” This will make the system more interactive, but it will also introduce additional constraints on data storage and mainly on efficient processing. We have already started parallelization efforts on some of the key models, so that even an execution of a single model can be done in parallel on a compute cluster.

Related Work

There has been little work in planner-based automation of data processing. Two notable exceptions are Collage (Lansky, 1993) and MVP (Chien, 1997). Both of these planners were designed to provide assistance with data analysis tasks, in which human was in the loop, directing the planner. In contrast, the data processing in systems like TOPS must be entirely automated; there is simply too much data for human interaction to be practical.

References

- Brittain, J., and Darwin, I.F. 2003. *Tomcat: The Definitive Guide*. O’Reilly.
- Chappell, D., and Jewell, T. 2002. *Java Web Services*. O’Reilly.
- Chien, S., Fisher, F., Lo, E., Mortensen, H., and Greeley, R. 1997. *Using artificial intelligence planning to automate science data analysis for large image database*. In Proceedings of 1997 Conference on Knowledge Discovery and Data Mining.
- Golden, K., and Frank, J. 2002. *Universal quantification in a constraint-based planner*. In Proceedings of 6th International Conference on AI Planning Systems.
- Golden, K. 2003. *A domain description language data processing*. In ICAPS 2003 Workshop on the Future of PDDL.
- Gordon, R. 1998. *Essential JNI: Java Native Interface*, Prentice Hall.
- Hamilton, G., Cattell, R., Fisher, M. 1998. *JDBC Database Access with Java*. Addison Wesley.
- Hunt, E.R. Jr., and Running, S.W. 1992. *Effects of climate and life form on dry matter yield from simulations using BIOME-BGC*. International Geosciences and Remote Sensing Symposium IGARSS 92: 1631-1633.
- Justice, C., et al. 1998. *The Moderate Resolution Imaging Spectroradiometer (MODIS): Land remote sensing for global change research*. IEEE Transactions on Geoscience and Remote Sensing, 36(4), 1228-1249, 1998.
- King, M., and Greenstone, R. 1999. *EOS Reference Handbook, A Guide to NASA’s Earth Science Enterprise and the Earth Observing System*. http://eosps0.gsfc.nasa.gov/ftp_docs/handbook99.pdf, 2004/02/28.

Knyazikhin, Y., et al. 1999. *MODIS Leaf Area Index (LAI) and Fraction of Photosynthetically Active Radiation Absorbed by Vegetation (FPAR) Product (MOD15) Algorithm Theoretical Basis Document*. http://modis.gsfc.nasa.gov/data/atbd/atbd_mod15.pdf, 2004/02/28.

Lansky, A.L., and Philpot, A.G. 1993. *AI-based planning for data analysis tasks*. In Proceedings of Ninth IEEE Conference on Artificial Intelligence for Applications (CAIA-93).

Laurie, B., and Laurie, P. 2002. *Apache: The Definitive Guide, 3rd Edition*. O'Reilly.

McGuffie, K., and Henderson-Sellers, R. 1997. *A Climate Modeling Primer*. John Wiley & Sons.

Momjian, B. 2001. *PostgreSQL: Introduction and Concepts*. Addison Wesley Professional.

Nemani, R., White, M., Votava, P., Glassy, J., Roads, J., and Running, S. 2000. *Biospheric forecast system for natural resource management*. In Proceedings of GIS/EM -4.

Nemani, R., Votava, P., Roads, J., White, M., Thornton, P. and Coughlan, J. 2002. *Terrestrial observation and prediction system: Integration of satellite and surface weather observations with ecosystem models*. Proceedings of the 2002 International Geoscience and Remote Sensing Symposium (IGARSS), Toronto, Canada.

Pitt, E., and McNiff, K. 2001. *java.rmi: The Remote Method Invocation Guide*, Addison Wesley.

Ramachandran, R., et al. 2001. *Earth Science Markup Language: A Solution for Generic Access to Heterogeneous Data Sets*. In Proceedings of Earth Science Technology Conference.

Votava, P., Nemani, R., Bowker, C., Michaelis, A., Neuschwander, A., Coughlan, J. 2002. *Distributed Application Framework for Earth Science Data Processing*. Proceedings of the 2002 International Geoscience and Remote Sensing Symposium (IGARSS), Toronto, Canada.