Comput Visual Sci (2009) 12:115–124
DOI 10.1007/s00791-007-0083-8

REGULAR ARTICLE

# Algorithms and the Grid

**Geoffrey C. Fox · Mehmet S. Aktas · Galip Aydin · Harshawardhan Gadgil · Shrideep Pallickara · Marlon E. Pierce · Ahmet Sayar**

Received: 15 August 2005 / Accepted: 13 January 2006 / Published online: 2 February 2008


**Abstract** We review the impact of Grid Computing and Web Services on scientific computing, stressing the importance of the "data-deluge" that is driven by deployment of new instruments, sensors and satellites. This implies the need to integrate the naturally distributed data sources with large simulation engines offering parallel low latency communication and so to integrate parallel and Grid computing paradigms. We start with an overview of these and the evolving service architectures. We illustrate the identified areas of interest for Algorithms and the Grid with the specific example of SERVOGrid that supports earthquake science research. We comment on the appropriate messaging infrastructure for Grids and data assimilation and contrast it with MPI.

## 1 Introduction: trends in simulation research

Over the last two decades, two very prominent trends may be noticed in many computing fields: the ever-increasing distribution of computing power and the increasing importance of data-centric computing. We consider the following timeline:

- *1990–2000:* The High Performance Computing and Communication (HPCC) initiative in the USA drove the development of parallel computing hardware and parallel algorithms. Major algorithm successes were achieved for partial differential equations and particle dynamics. The use of Message Passing Interface (MPI) implementations dominated parallel computing application development. The importance of data was not emphasized
- *1995-Present:* Distributed and Web computing emerge and grow aggressively. These tend to grow along two major lines: stateless, fault-tolerant, very loosely coupled Web applications based on the HTTP protocol and message exchanges (today referred to as the REST architecture [1] when applied to network application programming), and more tightly coupled distributed object systems such as the Object Management Group's CORBA [2], Sun Microsystems' Java RMI [3], Microsoft's DCOM [4], and the United States Department of Defense's HLA [5]. Grid computing also begins to emerge from academic and government research communities.
- *2000-Present:* Web Service-oriented distributed computing begins to replace distributed object technologies.
- *2000-Present:* Core parallel computing academic research in the United States drops dramatically at the expense of distributed and Grid computing. Parallel computing is continued by government efforts such as the Department of Energy's ASCI program and the Department of Defense's High Performance Computing Modernization Program.
- *2003-Present:* We see the emergence of the "Data Deluge" in many scientific fields. The data storage, management, and processing requirements force a number of fields, such as high energy physics [6], meteorology and weather modeling [7], and astronomy [8,9], to adopt distributed computing approaches. We discuss the impact on geophysics and earthquake science in more detail in this paper. Note that the data deluge is really "just Moore's Law applied to sensors" i.e. the increase in data is largely driven by the same technology driving forces that are driving increase in computer performance.

Communicated by P. Frolkovic.

G. C. Fox (✉) · M. S. Aktas · G. Aydin · H. Gadgil · S. Pallickara · M. E. Pierce · A. Sayar
Community Grids Laboratory, Indiana University, 501 North Morton Street, Suite 224, Bloomington, IN 47404, USA
e-mail: gcf@grids.ucs.indiana.edu

In this sequence of trends (parallel computing to distributed computing to data-deluged computing), each new trend builds upon, rather than replaces, the proceeding trends in layer-cake fashion. For a recent overview of parallel computing, see [10]. Grid computing is surveyed in [11,12]. An earlier review of Grid computing is given in [13].

## 2 Grids and virtual organizations

### 2.1 Introduction

Grid computing has diverged quite far from its initial incarnation as meta-computing, or the assemblage of distributed parallel computers into a single large virtual parallel machine. Such systems ultimately are limited by network communication speeds. Instead, the aim of Grid computing is to build the distributed computing infrastructure to support so-called Virtual Organizations (VOs). In a VO, many different member institutions can contribute resources to a single Grid, which may then be supported by a unified security infrastructure and information and monitoring system. These resources include computers, instruments, sensors, data repositories, networks and people. Useful services, such as secure, uniform remote access to high performance computing resources and secure, cross-institutional, reliable data management tools, can be built on top of this core infrastructure. Today, such systems are often termed "cyberinfrastructure" [14]. The development of cyberinfrastructure has been timely, as academic research funding has favored multi-disciplinary and multi-institutional teams of scientists. These trends in Grid computing to support "e-Science" have been mirrored in the commercial world: distributed computing technology development has been driven by the commercial sector's pressing need to integrate globally distributed enterprises.

### 2.2 Styles of Grids

Grid computing is a catch-all phrase that refers to several different types of distributed computing. In order to clarify this picture, we find it useful to identify major Grid families. As we shall see, this seemingly diverse collection of capabilities can be unified into a single coherent picture, based on Service Oriented Architecture principles [15].

- Computational Grids: these are traditional Grids that are designed to provide support for high performance computing resources. Such Grids are still quite popular and valuable.
- Sensor and Data Grids: these are Grids that provide access to data and related metadata. The data may be archival or real-time, collected in either case from sensors, scientific instruments, etc. Metadata, or "data about data" is also very important in these simulations. Geographic Information Systems (GIS) described in Sect. 5 provide an important sub-family in this group.
- Collaborative Grids: these Grids support communication in all forms, ranging from document and message sharing to instant messaging to audio/video collaboration. Group participation and data sharing are also important to these applications.
- Peer-to-Peer, or Community Grids: these Grids apply principles of peer-to-peer computing (such as decentralized, dynamic organization) to scientific computing resource collections.
- Semantic Grids: these Grids focus on information representation and management. Such information management may be important for both human users as well as machine processing. Semantic information systems may support other styles of Grids: they are potentially an excellent way to manage multi-staged computing tasks ("workflow") that must run in a distributed environment.

Grid applications in many fields must rely upon services that emerge from many of these families. Military command and control and civilian emergency preparedness and response for crisis management are two such examples. In either case, Grid-based collaboration services must link participants, many of whom will be on unreliable networks. Participants will need to rapidly assess data, so integration of data Grid services with computational processing is required. We refer to these collections of Grid service families as a "Grid of Grids." [16]

## 3 Service oriented architectures

Around the beginning of the current decade, industry software developers began to place emphasis into Web Services at the expense of the distributed object technologies that dominated the 1990's. Similarly, Grid development began to align itself with Web Services shortly thereafter. Motivating this shift was the desire for a loose coupling between distributed components. Distributed objects with an implicit relationship between interface and implementation and the coupling of components via remote procedure calls (RPC) proved hard to scale.

Web Services are founded on the following core concepts [15]:

- Desirable capabilities may be accessed through remote services.
- These services define a contract for interaction using the XML-based Web Service Description Language (WSDL) [17].
- Requester agents (i.e. clients) interact with these services, or provider agents.

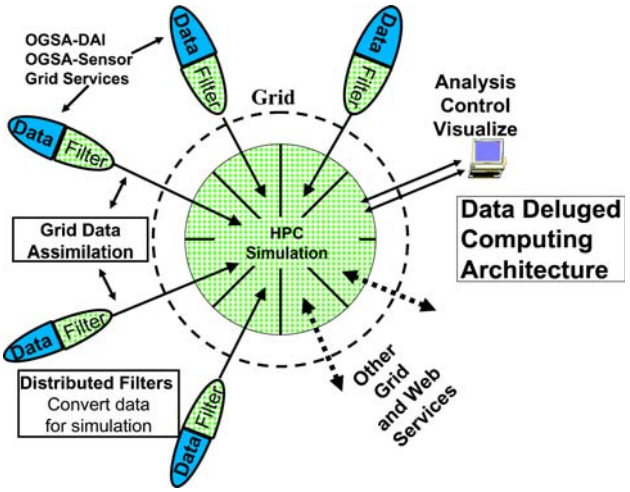


**Fig. 1** The figure illustrates the hub and spoke architecture for assimilating data using extreme scale computing. Distributed data sources use local clusters for adaptively filtering data, which is then transported using high-performance transport services to one or more centralized high performance computers

– Requesters and providers communicate by exchanging messages, encoded in the XML-based Simple Object Access Protocol (SOAP) [18].

A key difference between Web Services and distributed object systems is that Web Services are essentially XML-document (or message) exchange systems. The SOAP-enveloped messages are largely self-contained and self-descriptive. A SOAP message may include, for example, all necessary information for its secure transmission. This allows the message to be decoupled from, and transmitted over, any appropriate transport protocol. It also allows the message to be decoupled from explicit point-to-point connection protocols. SOAP is purposefully constructed to be extensible. Important extensions include reliability, security, and addressing. Later in Sect. 6 we will contrast messaging built SOAP with that familiar in parallel computing through MPI.

## 4 The impact of Grids on algorithms

As discussed above, Grid computing in its general sense does not replace or expand parallel computing in the classic "metacomputing" sense. Thus we find Grid extensions of MPI to be only part of the solution with a richer model discussed in Sect. 6. Instead, Grids are geared toward resource management across organizations. Parallel computing applications and hardware are examples of these resources as shown in Fig. 1. In this simple sense, implementations of parallel algorithms may exist untouched running on resource nodes in a Grid.

The above assumes of course that Grids will be used to manage classic high performance computing applications. While this will remain an important style of Grids, data driven high performance computing applications of data assimilation and data mining are increasingly important. These applications must rely upon external data sources (both real-time data streams and archival data) that typically are remote from the high performance computing resource.

More generally, Grid-based algorithms may be developed to support the interconnections of loosely coupled distributed applications. Such applications may consist of remote data source, a number of linked applications running on separate supercomputers, filtering programs for data format conversations between applications, and so forth. Service oriented Grid computing offers the general architecture for accomplishing these linkages. Management of these loosely integrated applications is known as Grid workflow discussed in more detail in Sect. 6. Workflow builds on classic distributed programming models such as Linda [19] and the data flow concept pioneered by AVS [20]. This type of Grid application includes "code-coupling" which supports multi-disciplinary simulations such as those linking aerodynamics and structures or ocean and atmosphere. These simulation coupling can of course be combined with data assimilation. The latter is a key area where new algorithm work is critical. A typical scenario shown in Fig. 1 can apply to scientific simulations such as weather forecasting with data from satellites and sensors; it can also be seen in financial modeling or military intelligence applications where data could be stock prices or electronic signals. The latter illustrate that new applications are enabled by the data-deluge and these need new algorithms. A good example is complex systems simulations for social science and biology, which up to now have not been major users of large scale computers. The capability of Grids to marshal their data implies the possibility of new algorithms for such simulations as those of biological organisms, the spread of disease and the response of critical infrastructure (transportation, energy, communication) to major disruptions from earthquakes, hurricanes or terrorists.

Such applications must be able to address problems not typically encountered in classic parallel computing.

– Applications must be fault tolerant, as failures become increasingly likely in Grid applications.
– Applications must be able to tolerate millisecond (and preferably longer) communication latencies discussed in Sect. 6 instead of microsecond messaging speeds in MPI. Note the need for latency tolerant algorithms to exploit distributed computing is well known. We do not see any major developments here in new algorithms for traditional core scientific computing problems—say solving partial differential equations or particle dynamics. These

still need low latency i.e. classic parallel systems, for good performance.
– Information management becomes increasingly important, as workflow applications benefit from late binding: decisions on which specific service instances to use are not made until needed.

Generalizing the last point, we discuss in Sect. 6 that Grids are developing the best core technology to build Problem Solving Environments (PSE) as these need exactly the integration and management delivered by Grids and the multi-millisecond latency is typically not important in the initiation of a PSE or toolkit component.

There are examples of important algorithms enabled and required by distributed environments. One example is the distributed hash tables used in scalable peer to peer lookup and less well known are distributed security algorithms to support dynamic communities. Data mining algorithms are already active areas of research and these are needed for the distributed system itself as for example in the analysis of denial of service attacks and other Internet activities.

In the following section, we will draw upon our work in the NASA SERVOGrid project to make proceeding discussion concrete. We then discuss in Sect. 6 our research on high performance Internet messaging for Grids using the Narada-Brokering messaging system.

## 5 SERVOGrid: integrating data and computing Grids

The NASA AIST funded SERVOGrid project is being designed and built to integrate scientific applications with data resources. Users typically interact with the system using computational Web portals. SERVOGrid is described in more detail in [21]. See also [22].

### 5.1 SERVOGrid applications

The following is a partial list of SERVOGrid applications:

– *GeoFEST* is a three-dimensional viscoelastic finite element model for calculating nodal displacements and tractions. It allows for realistic fault geometry and characteristics, material properties, and body forces. GeoFEST requires earthquake fault models as input data.
– *Virtual California* (VC) simulates interactions between vertical strike-slip faults using an elastic layer over a viscoelastic half-space. Virtual California requires earthquake fault and friction models as input data.
– *Pattern Informatics* (PI) calculates regions of enhanced probability for future seismic activity based on the seismic record of the region. Pattern Informatics requires seismic catalogs as input.
– *RDAHMM* (for Regularized Deterministic Annealing Hidden Markov Model) is a time series analysis program based on Hidden Markov Modeling. It produces feature vectors and probabilities for transitioning from one class to another. RDAHMM may be applied to any time series data, such as GPS and seismic data.

As can be seen from the list, these applications have increasingly interesting data requirements: GeoFEST and VC are traditional, parallel high performance computing applications that have external data requirements that may be fulfilled by agencies such as Earthscope or SCEC. PI relies on seismic catalogs that are regularly updated, and RDAHMM relies upon regularly updated GPS catalogs. RDAHMM is also an excellent candidate for real-time data analysis, as we will discuss. Typically in the code development phase of SERVOGrid applications, these data sets are downloaded by the code developers from online archives and stored locally in files. However, when integrating these applications into Web portal environments, or otherwise attempting to automate the code execution phase, we must find an alternative strategy.

This application list is interesting in that it includes classic scientific algorithms solving differential equations mixed with data-mining and analysis codes. These two classes interact with the classic simulations being used to test and motivate the data-mining. In the previous sections, we noted the importance of algorithms that support interactions with data; data-mining typically corresponds to codes that directly analyze data and data assimilation to codes where simulation and data are mixed. We see all these classes growing in importance with data assimilation being developed for SERVOGrid as the amount of real-time data increases.

### 5.2 GIS Grid services for SERVOGrid

Implementations of Geographical Information Systems (GIS) standards provide Data Grid services that can be used to provide programming interfaces (as distinguished from human user interfaces) to data. Our efforts in building GIS Grid services are described in related publications, and we summarize briefly here. We base our service implementations on the Open Geospatial Consortium (OGC)'s specifications. These are described in [23] and more publications and reports are available from [24]. *Web Feature Services* [25] are used to store archived data that may be used to describe abstract map features. The WFS is a useful general purpose GIS archived data service: in SERVOGrid we typically use it to store archived records for GPS stations, seismic activity, and faults. We have implemented Web Service versions of WFS that use both SOAP over HTTP and higher performance streaming data that are described in Sect. 7. *Web Map Services* [26] generate human readable maps from Web

Feature Servers and other Web Map Servers. Our approach to *GIS Information Service* deviates from the OGC specifications, since, by using a Web Service approach to information management, we may adopt more general Web Service information systems. Finally, *Sensor Web Enablement* [27] is a family of related specifications for describing sensors. Our efforts here have focused on integrating NaradaBrokering messaging described in Sect. 6 with streaming GPS data.

## 6 Integrating pattern informatics and GIS Grid services: programming the Grid

The Pattern Informatics (PI) application [28] has been successfully used to forecast regions of increased probability for large seismic activity. PI techniques have been applied regionally (to the Western United States and to Japan) and globally. The PI application uses seismic archive data, such as provided by the Southern California Earthquake Center (SCEC) and the United States Geographical Survey's Advanced National Seismic System (ANSS), as input data and generates a latitude/longitude grid of relative probability changes that may be superimposed on maps.

Automated versions of PI may be integrated with GIS services in three ways:

- Web Feature Services may be used retrieve seismic archives based on query filters such as latitude/longitude bounding boxes, event magnitude ranges, and time periods.
- Web Map Services may be used to build interactive user interfaces and batch-style maps.
- GIS Information services can be used to locate both Web Feature Services and Web Map Services that have the desired capabilities. For example, one may locate a WFS that has the seismic records for the desired bounding box.

These data web services may be coupled to "Execution Grid" style services that can be used to manage running applications. This coupling is sometimes referred to as "service orchestration" or "Grid workflow."

Scientific workflow management described in Sect. 6 is an active field, and overviews are available from [29]. Typically these tools are coupled to programming environments: Grid-enabled scripting languages and XML workflow expression languages are common, and are often coupled to visual programming tools.

HPSearch (www.hpsearch.org) is our research effort in this area. HPSearch collectively includes several constituent parts, including support for JavaScript-based Grid workflow scripting; distributed Web Services for workflow enactment; and Web Service wrappers that can be used to exchange events between Web Services and the HPSearch controlling

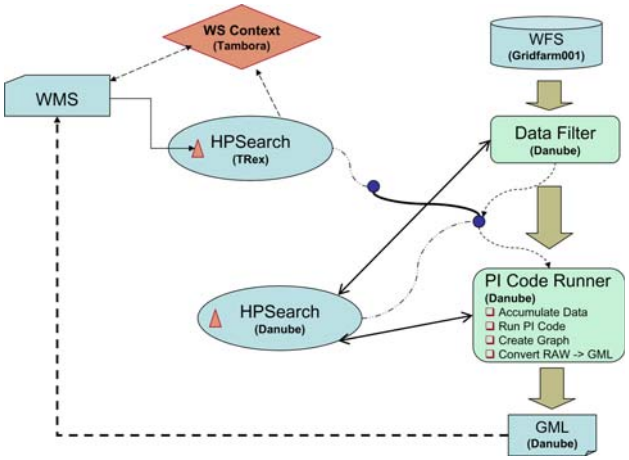


**Fig. 2** GIS Grid and Execution Grid services may be integrated into meta-applications using workflow orchestration tools such as HPSearch. This represents a "two-level" programming model: applications are wrapped as services that are in turn programmed (or scripted) on a Grid. *Dashed arrows* represent actual data (file) flow. *Solid arrows* indicate control messages sent by HPSearch. *Dot-dash arrows* show messages transmitted through NaradaBrokering

engine. HPSearch research focuses on two additional areas not usually accounted for in workflow systems: management of data streams and management of messaging (NaradaBrokering) networks.

Figure 2 below illustrates the integration of HPSearch with GIS services and the PI code. This is described in more detail in [23]. Web Map Server clients initially generate requests for PI code runs by specifying time intervals, latitude and longitude bounding boxes, lower cutoff values for earthquake magnitudes used in the forecast, and the size of the latitude/longitude grid of the bounding box. These request parameters are passed to the HPSearch master node (TRex in the figure), which first chooses an available HPSearch worker node (Danube) to execute the workflow. The worker node is passed a URI name for the JavaScript workflow to be executed. The worker node then manages the workflow chain: the desired seismic data (in GML format) is fetched from an appropriate Web Feature service and filtered into a format understandable by the PI code. The filtered data is then collected and the PI code is launched. PI output data (a grid of seismic "hot spots" forecasts) is then filtered back into GML feature data that may be suitably rendered by the Web Map Server. The WS-Context server is used to keep track of the evolving state of this workflow: several components may register for interest in events, such as the notification of workflow completion.

The individual components in Fig. 2 represent separate Web Services running on distributed hosts. The key element in this model is that the Web Services may be developed and maintained independently of each other. Their combination into this particular application is ad-hoc. HPSearch

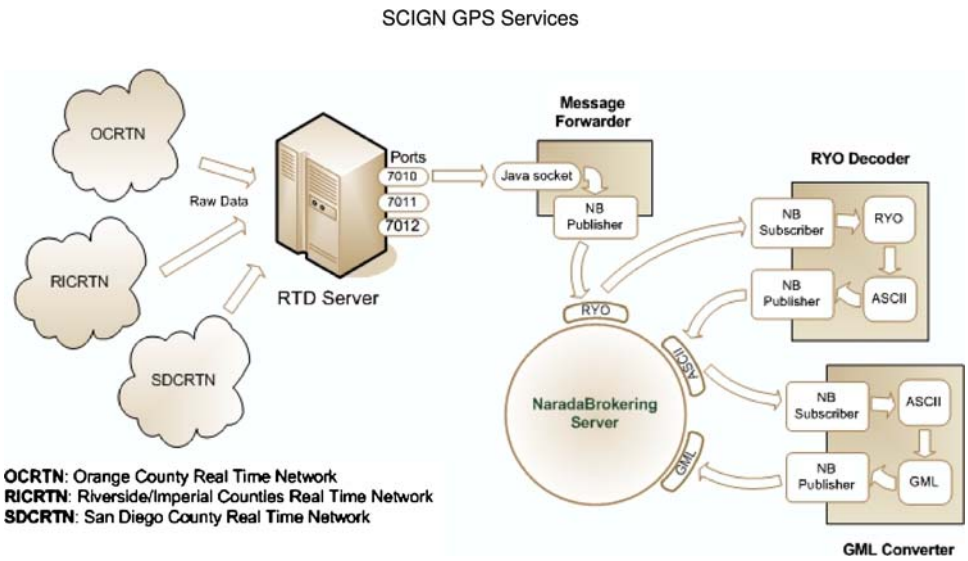


**Fig. 3** GPS real-time data filtering is accomplished using NaradaBrokering filter chains

supplies the scripting control and the "glue" filters for this specific meta-application. The PI application itself is a relatively small code, but more sophisticated parallel applications may be managed in a similar manner.

Our evaluation of this meta-application revealed two not-surprising bottlenecks: HTTP is too inefficient for non-trivial data transport, and ASCII-based XML representation can lead to processing overheads as well as transport overheads. We view both of these problems as opportunities for research. Efficient transport of XML messages is the subject of the next section, and efficient XML processing is an active effort of research [30].

### 6.1 Providing live GPS data for data mining applications

The Pattern Informatics application is an example of file-based workflow, but these techniques may also be applied to real-time applications. In this section we discuss work in progress to integrate GPS data streams with the RDAHMM application [31]. This work is part of a general research area in real-time change detection and data mining.

RDAHMM may be used to analyze time series data to automatically determine underlying physical modes in the series. For example, earthquakes and seismic activity in California may show up in nearby GPS stations maintained by the Southern California Integrated GPS Network (SCIGN). RDAHMM may be trained to automatically detect these in archival GPS records. RDAHMM may also be used to detect more subtle mode shifts in time series signals that are not easily discernable to the human eye. Currently RDAHMM may be used to analyze GPS archival records, and we may apply workflow techniques to this problem that are similar to the PI application discussed above.

However, we also see an opportunity for event change detection in real-time data. RDAHMM service instances may be trained on archival GPS data to identify the station's historical modes. A network of RDAHMM services may then be attached to GPS networks. State changes in individual GPS stations may be detected in real-time, and interested subscribers can receive notification.

A real-time version of RDAHMM is still in development, but we have recently completed an initial version of a real-time GPS filtering system that may be used to deliver data to RDAHMM. This is illustrated in Fig. 3. Current support includes three GPS networks (OCRTN, RICRTN, and SDCRTN) with 28 GPS stations. Raw data is collected by an RTD server maintained by Scripps Orbit and Permanent Array Center (SOPAC). Each network is published on a separate network port in the binary RYO format. Data rates are 1–2 Hz. All stations in a particular network are published on the same port: the 12 GPS stations in OCRTN are all published on the same port, 7010, for example.

Applications such as RDAHMM have two requirements for this data: GPS network streams should be decoupled into individual station streams, and the data streams should be converted into displacement values. We anticipate, with the adoption of OGC Sensor Web Enablement applications, that other applications may expect data in "Observations and Measurements" GML format. We address these issues by building filter chains. NaradaBrokering endpoints (simple Java programs) have been developed to accept data from the streaming ports 7010–7012. This data is then published into a broker network on an appropriate topic (or channel) name. See Fig. 4 for a list of channels. For example, RYO data from the OCRTN network is published to the /SOPAC/GPS/Positions/OCRTN/RYO channel.

GPS Topic Filter Channels

**NaradaBrokering Server:** xsopac.ucsd.edu:3045

| Network | Format | NB Topic |
|---|---|---|
| OCRTN | RYO | SOPAC/GPS/Positions/OCRTN/RYO |
| | ASCII | SOPAC/GPS/Positions/OCRTN/ASCII |
| | GML | SOPAC/GPS/Positions/OCRTN/GML |
| RICRTN | RYO | SOPAC/GPS/Positions/RICRTN/RYO |
| | ASCII | SOPAC/GPS/Positions/RICRTN/ASCII |
| | GML | SOPAC/GPS/Positions/RICRTN/GML |
| SDCRTN | RYO | SOPAC/GPS/Positions/SDCRTN/RYO |
| | ASCII | SOPAC/GPS/Positions/SDCRTN/ASCII |
| | GML | SOPAC/GPS/Positions/SDCRTN/GML |

**Fig. 4** GPS data filters use topics, or channels, to manage data routing using the NaradaBrokering software

NaradaBrokering subscribers to this channel may receive this data directly. One such subscriber is a RYO-to-ASCII filter, which converts data into text format. This filter acts in turn as a publisher to the appropriate topic (such as SOPAC/GPS/Positions/OCRTN/ASCII), where interested subscribers may receive it. We may add any number of additional filters in the filter chain, including GML converters and station decoupling filters. RDAHMM applications may similarly be wrapped to receive data on the appropriate channel, and publish interesting state change events.This work is described in more detail at [32], which also includes simple demos and downloadable software.

Returning to Fig. 1, we see data manipulation is depicted at the edges of the diagram (at the sources) before their integration and assimilation in the "central" simulation. Our GPS filtering is equivalent to the edge filtering whereas Pattern Informatics is a typical central activity. This illustrates the algorithmic opportunity to identify decoupled parts of the analysis that can be placed on cost effective distributed platforms. In fact this decoupling is needed to match the volume of data to the simulation; otherwise the data deluge will overwhelm the simulation engine whose low latency interconnects make it powerful but more expensive per operation than distributed commodity analysis systems. Algorithms that optimally support this split into parallel assimilation and distributed filtering need further study.

## 7 Problem solving environments and messaging

Typical Web Services are used in situations with interaction delays (network transit) of 100s of milliseconds. However, basic message-based interaction architecture only incurs fraction of a millisecond delay. One strategy would thus be the use of Web Services to build all PSE components. Here we note that in such settings interactions between the distributed entities should be based on the use of messages and not remote method/subroutine calls.

Messaging has several advantages over scripting languages. First, collaboration between the distributed entities is trivial since it simply involves sharing of messages. The richness of collaborative interactions is dependent on the information encapsulated within these messages. Second, messaging-based interactions engender greater modularity in the development of systems. For example, one could have separate classes or modules to deal with different types of messages. Finally, Web Services presently have (and we expect this to improve even more) very good support for messaging.

Loosely-coupled applications can also leverage workflow technologies. But perhaps the most important decision in the building of problem solving environments is the determination of the characteristic interaction time (millisecond programs; microseconds MPI and particle) and use the best supported architecture at this level.

A major difficulty in these frameworks is not their construction but their support. Here we suggest that it might be best to minimize life-cycle support risks. While building very specialized problems solving environments it must be noted that sometimes these applications tend to be very specialized to support much. Designers must thus expect to use *different* technologies at each level even though it might be possible to do everything with one technology. Designers might also have to make decisions regarding trade offs involving support versus performance/customization

### 7.1 Requirements for MPI messaging

MPI and SOAP Messaging both send data from a source to a destination. We now proceed to briefly enumerate some of the characteristics pertinent to MPI. MPI includes support for multicast (broadcast) communication, and specifies both the destination and a context (in the comm parameter). MPI specifies the data to be sent and has a tag to allow flexibility in processing in the source processor. Furthermore, MPI has calls to understand context (number of processors etc). MPI requires very low latency and high bandwidth so that $t_{comm}/t_{calc}$ is at most 10 (here, $t_{calc}$ corresponds to the time spent processing a message while $t_{comm}$ corresponds to the communication delays for the message in transit). For example, BlueGene/L has bandwidth between 0.25 and 3 Gigabytes/sec/node and latency of about 5 ms.

Web Services have much of the same requirements as MPI with two differences, where MPI is more stringent than SOAP. First, latencies are inevitably 1 (local) to 100 milliseconds, which is 200 to 20,000 times that of BlueGene/L. To put these numbers in perspective, typically it takes 0.000001 ms for a CPU to perform a calculation; MPI latency tends to be in the order of 0.001–0.01 ms; it takes around 1–10 ms to wake

up a thread or perform a context switch between processes; finally, Internet delay tends to be between 10–1,000 ms. However, unlike MPI, SOAP has far greater flexibility in areas like security, fault-tolerance, "virtualizing addressing" because one can run a lot of software in 100 ms. Typically it takes 1–3 ms to process a modest message in Java and "add value"

SOAP defines a very obvious message structure with a header and a body just like email. The header contains information used by the "Internet operating system." This includes the intended destination, source, routing information, and context. The SOAP message body is partly further information used by the operating system and partly information for the application when it is not looked at by "operating system" except to encrypt, compress it etc. Here we note that WS-Security supports separate encryption for different parts of a document. Much discussion in the field revolves around what is referenced in header. This structure makes it possible to define very sophisticated messaging.

One key difference between SOAP and MPI is that MPI only specifies interface and so interoperability between different MPI implementations requires additional work [33]. Pervasive networks can support high bandwidth (Terabits/sec soon) but the latency issue is not resolvable in general way. One can possibly combine MPI interfaces with SOAP messaging but this has perhaps not yet been done. Just as walking, cars, planes, phones coexist with different properties; so SOAP and MPI are both good and should be used where appropriate.

In messaging systems distributed entities communicate through the exchange of messages. These messages are self-describing and contain the semantic intent of the message. Depending on the application these messages can be made to encapsulate system conditions, method invocations, resource sharing, data interchange among others. Messages may also describe their correlation, dependency and causal relationships to other messages. In understanding the requirements of message passing, it is useful to divide multi-processor (distributed memory) systems into three classes.

1. Classic massively parallel processor systems (MPP) with low-latency, high-bandwidth specialized networks. Here one aims to have message latencies in the order of one to a few microseconds and scalable bisection bandwidth. Ignoring latency, the time to communicate a word between two nodes should be a modest multiple (perhaps 20) of time taken to calculate a floating point result. This communication performance should be independent of number of nodes in system.
2. Commodity clusters with high performance but non-optimized communication networks. Latencies can be in the 100–1,000 ms range typical of simple socket based communication interfaces.
3. Distributed or Grid systems with possibly very high internode bandwidth but the latency is typically 100 ms or more as familiar from Internet travel times.

Of course there is really a spectrum of systems with cost-performance increasing by four orders of magnitude or so as one goes from (1) to (3). Here we will focus on the endpoints (1) and (3)—MPP's and the Grid—and not worry about intermediate cases like 2). MPI especially is aimed at the class (1) with optimized *native* implementations exploiting particular features of the network hardware. Generic versions of PVM [34] and MPI [35] using socket based communication on a localized network illustrate (2) as do many other specialized programming environments (such as agent-based approaches). The latter typically cannot afford the development effort to optimize communication. Furthermore, Grid messaging requires substantially more functionality than MPI and PVM. Grid systems are very diverse and there is little understanding at this stage as to critical performance and architecture (topology) characteristics. There is thus a need for a rich messaging environment very different from MPI and PVM. Note this does not mean that we should not port systems like MPI to the Grid as in MPICH-G2 [36] and PACX-MPI [37]; there is clearly a need to run all messaging environments on all classes of machine.

### 7.2 NaradaBrokering

We have built a messaging system—NaradaBrokering [38]—that is designed to support traditional Web Services but has an architecture that allows it to support high performance data transport as required for Scientific applications. We suggest using this system whenever your application can tolerate 1–10 ms latency in linking components. For applications that require lower latencies we recommend the use of MPI.

The NaradaBrokering messaging infrastructure incorporates support for enterprise messaging specifications such as the Java Message Service [39]. It also incorporates support for Web Service specifications such as WS-Eventing [40], WS-ReliableMessaging [41] and WS-Reliability [42]. The NaradaBrokering messaging infrastructure includes support for services such as reliable and ordered delivery. Services available within the messaging substrate also include replay support, recording, synchronized Network Time Protocol based timestamps, buffering and discovery of nodes.

### 7.3 Fast Web service communications

Internet Messaging systems allow one to optimize message streams at the cost of *startup time*. Web Services can deliver the fastest possible interconnections with or without reliable messaging. The costs involved in reliable messaging for Web Services tend to be a little higher, since the dominant

specifications WS-Reliability and WS-ReliableMessaging both mandate the storage of the SOAP message to stable storage prior to forwarding the message.

It is possible to have very fast communications within Web Services. Here we outline our scheme which works for streams, which are essentially sets of related messages. For individual messages in the streams the SOAP header is constant except for headers pertaining to sequence number (Message ID), time-stamp. Next, one needs two types of new Web Service Specifications viz. WS-StreamNegotiation and WS-FlexibleRepresentation. WS-StreamNegotiation defines how one can use WS-Policy to send messages at the start of a stream to define the methodology for treating remaining messages in the stream. WS-FlexibleRepresentation is used to define new encodings for messages.

We then use WS-StreamNegotiation to negotiate stream in Tortoise SOAP—ASCII XML over HTTP and TCP. We deposit the basic SOAP header through this connection; this is part of the context for the stream (linking of two services). This exchange is also used to agree on firewall penetration, reliability mechanism, binary representation and fast transport protocols. Next we use WS-FlexibleRepresentation to define encoding of a Fast transport (On a different port) with messages just having FlexibleRepresentationContextToken, Sequence Number, Time stamp if needed. RTP packets have essentially this structure. Here we could add stream termination status. Using this strategy one can monitor and control with the original negotiation stream. Furthermore, one can also generate different streams optimized for different endpoints.

### 7.4 The role of workflow

The Web Service (Grid) paradigm implicitly assumes a two-level Programming Mode. Here one makes a Service (same as a "distributed object" or "computer program" running on a remote computer) using conventional technologies such as C++ Java or Fortran Monte Carlo module, data streaming from a sensor or Satellite and specialized (JDBC) database access. Such services accept and produce data from users files and databases. The Grid is built by coordinating such services assuming we have solved problem of programming the service

The Grid is discussing the composition of distributed services with the runtime interfaces to Grid as opposed to UNIX pipes/data streams. One can use UNIX Shell, PERL or Python scripts to produce real applications from core programs. Such interpretative environments are the single processor analog of Grid Programming. Some projects like GrADS from Rice University are looking at integration between service and composition levels but dominant effort looks at each level separately.

In programming SOAP and Web Services (the Grid) workflow describes linkage between services. Since the various services, resources and other components are distributed, the linkage between them must be through the exchange of messages. This linkage is twoway and has both control and data. The BPEL specification from Microsoft-IBM is currently the preferred Web Service XML specification of workflow.

## 8 Summary and conclusion

We have reviewed the impact of the Grid computing and data-intensive computing on future algorithm development. We must build upon parallel computing applications and high performance computing to integrate geographically distributed data sources. The driving constraint that separates Grid computing from parallel computing is the so-called "rule of the millisecond." Grid application components must be tolerant of millisecond (or longer) communication latencies, but this constraint does not apply to the interior workings of specific components, which may be parallel applications. This implies a "two-level" programming model: components may be developed as parallel services, but they are programmed by Grid workflow technologies.

Grid components are developed as Web Services that follow the conventions of Service Oriented Architecture design. In this model, Grid components are self-contained, have a well-defined programming interface defined in WSDL, and communicate using SOAP messaging. The efficiency of SOAP messaging may be dramatically improved by adopting faster transport mechanisms, such as the NaradaBrokering system, and by efficient representations.

**Acknowledgments** This paper describes work that is supported by the Advanced Information Systems Technology Program of NASA's Earth-Sun System Technology Office and the UK e-Science program's Open Middleware Infrastructure Institute (OMII).

## References

1. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Dissertation, University of California, Irvine (2000)
2. The Object Management Group: common object request broker architecture: Core Specification. See: http://www.omg.org/docs/formal/04-03-12.pdf
3. Sun Microsystems: Java Remote Method Invocation (Java RMI) See: http://java.sun.com/products/jdk/rmi/
4. Horstmann, M., Kirtland, M., DCOM Architecture
5. Standard for Modeling and Simulation High Level Architecture (HLA)—Framework and Rules. Electronics Engineers (IEEE) Standard 1516 (2000)
6. Allcock, B., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnal, D., Tuecke, S.: Data management and transfer in high performance computational grid environments. Parallel Comput. J. **28**(5), 749–771 (2002)

7. Gannon, D., Krishnan, S. Fang, L., Kandaswamy, G., Simmhan, Y., Slominski, A.: On building parallel and grid applications: component technology and distributed services. In: Proceedings of 2nd international workshop on challenges of large applications in distributed environments (CLADE 2004), 7 June 2004, pp. 44–51 Honolulu (2004)
8. Barish, B.C., Weiss, R.: Physics Today, October 1999, pp. 44–50; See: http://www.ligo.caltech.edu/
9. Brunner, R.J., Djorgovski, S.G., Szalay, A.S. (eds.): Virtual observatories of the future, astronomical society of the pacific conference series, Vol. **225** (2001)
10. Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., White, A. (eds.): The Sourcebook of Parallel Computing, Morgan Kaufmann, San Francisco (2002)
11. Foster, I., Kesselman, C. (eds.): The Grid 2: Blueprint for a New Computing Infrastructure, 2nd edn. Morgan Kaufmann, San Francisco (2004)
12. Berman, F., Fox, G., Hey, T. (eds.): Grid Computing: Making the Global Infrastructure a Reality. Wiley, Chichester, http://www.grid2002.org, (2003)
13. Foster, I., Kesselman, C. (eds.).: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1998)
14. Atkins, D.E., Droegemeier, K.K., Feldman, S.I., Garcia-Molina, H., Klein, M.L., Messerschmitt, D.G., Messina, P., Ostriker, J.P., Wright, M.H.: Revolutionizing Science and Engineering Through Cyberinfrastructure. Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure, Available from http://www.nsf.gov/cise/sci/reports/atkins.pdf, (2003)
15. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web Service Architecture. W3C Working Group Note, 11 February 2004, Available from http://www.w3c.org/TR/ws-arch, (2004)
16. Fox, G.: Grids of Grids of simple services. Comput. Sci. Eng. **6**, 84–87 (2004)
17. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Service Description Language (WSDL) 1.1. W3C Note (2001)
18. Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H.: SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation 24 June 2003. Available from http://www.w3c.org/TR/soap12-part1/, (2003)
19. Carriero, N., Gelernter, D.: Linda in context. Commun. ACM **32**(4), 444–458 (1989)
20. Upson, C., Faulhaber, T. Jr., Laidlaw, D., Schlegel, D., Vroom, J., Gurwitz, R., Dam, A.: The application visualization system: a computational environment for scientific visualization. IEEE Computer Society Press, IEEE Comput. Graph. Appl., Vol. **9**, no. 4, 30–42 (1989)
21. Aktas, M., Aydin, G., Donnellan, A., Fox, G., Granat, R., Grant, L., Lyzenga, G., McLeod, D., Pallickara, S., Parker, J., Pierce, M., Rundle J., Sayar, A., Tulli,s T.: iSERVO: Implementing the International Solid Earth Research Virtual Observatory by Integrating Computational Grid and Geographical Information Web Services. Accepted for publication in Special Issue of Pure and Applied Geophysics (PAGEOPH) for ACES Workshop, Beijing, China (2004)
22. Donnellan, A., Rundle, J., Fox, G., McLeod, D., Grant, L., Tullis, T., Pierce, M., Parker, J., Lyzenga, G., Granat, R., Glasscoe, M.: QuakeSim and the Solid Earth Research Virtual Observatory, Accepted for Publication in Special Issue of Pure and Applied Geophysics (PAGEOPH) for ACES Meeting, Beijing, China (2004)
23. Aydin G., Aktas M.S., Fox G.C., Gadgil H., Pierce M., Sayar A.: SERVOGrid Complexity Computational Environments (CCE) Integrated Performance Analysis. Grid Computing, 2005. The 6th IEEE/ACM International Workshop on 13–14 November 2005, pp. 256–261 (2005)
24. GIS Research at the Community Grids Laboratory: http://www.crisisgrid.org
25. Vretanos, P.: Web Feature Service Implementation Specification, OpenGIS project document: OGC 02-058, version 1.0.0. (2002)
26. de La Beaujardiere, J.: Web Map Service, OGC project document reference number OGC 04-024 (2004)
27. Botts, M.: Sensor Model Language (SensorML) for In situ and Remote Sensors, OGC document reference number: 04-019r2
28. Tiampo, K.F., Rundle, J.B., McGinnis, S.A., Klein, W.: Pattern dynamics and forecast methods in seismically active regions. Pure Appl. Geophys. **159**, 2429–2467 (2002)
29. Fox, G., Gannon, D. (eds.).: Concurrency and Computation: Practice and Experience. Upcoming Special Issue on Grid Workflow. See http://www.extreme.indiana.edu/groc/ggf10-ww/
30. Oh, S., Bulut, H., Uyar, A., Wu, W., Fox, G.: Optimized Communication using the SOAP Infoset For Mobile Multimedia Collaboration Applications Proceedings of the International Symposium on Collaborative Technologies and Systems CTS05, St Louis Missouri (2005)
31. Granat, R., Donnellan, A.: A hidden Markov model tool for geophysical data exploration. Pure Appl. Geophys. 2271–2284 (2002)
32. Sensor Grid Research at the Community Grids Laboratory: http://www.crisisgrid.org/html/sensorgrid.html
33. George, W., Hagedorn, J., Devaney, J.: IMPI: Making MPI interoperable. J. Res. Nat. Inst. Standards Technol. **105**(3) (2000)
34. Sunderam, V.: PVM: a framework for parallel distributed computing, Concurrency. Prac. Exp. **2**(4), 315–339 (1990)
35. The Message Passing Interface Standard, http://www-unix.mcs.anl.gov/mpi/standard.html
36. Karonis, N., Toonen, B., Foster, I.: MPICH-G2: a grid-enabled implementation of the message passing interface. J. Parallel Distrib. Comput. **63**(5), 551–563 (2003)
37. Beisel, T., Gabriel, E., Resch, M.: An Extension to MPI for Distributed Computing on MPPs. In: Bubak, M., Dongarra, J., Wasniewski, J. (eds.) Recent Advances in Parallel Virtual Machine and Message Passing Interface, Lecture Notes in Computer Science, Springer, Heidelberg, pp. 75–83 (1997)
38. Pallickara, S., Fox, G.: NaradaBrokering: a distributed middleware framework and architecture for enabling durable peer-to-peer grids. In: Endler and M., Schmidt, D. (ed.): Proceedings of Middleware 2003, Rio de Janeiro, Brazil, June 16–20, pp. 41–61 (2003). See http://www.naradabrokering.org
39. Happner, M., Burridge, R., Sharma, R.: Sun Microsystems. Java Message Service Specification. See, http://java.sun.com/products/jms, (2000)
40. Box, D., et al.: Web Services Eventing (WS-Eventing) Specification., Microsoft, IBM & BEA (2004)
41. Bilorusets, R., et al.: Web Services Reliable Messaging Protocol (WS-ReliableMessaging) Specification (2004)
42. Iwasa, K. (ed.): Web Services Reliable Messaging TC WS-Reliability 1.1. Oasis Standard (2004)