Algorithms and the Grid

Geoffrey C. Fox[1], Mehmet S. Aktas[2], Galip Aydin[3], Harshawardhan Gadgil[4], Shrideep Pallickara[5], Marlon E. Pierce[6], and Ahmet Sayar[7]
Email: [1]gcf@grids.ucs.indiana.edu, [2]maktas@cs.indiana.edu, [3]gaydin@cs.indiana.edu, [4]hgadgil@cs.indiana.edu, [5]spallick@indiana.edu, [6]mpierce@cs.indiana.edu, and [7]asayar@cs.indiana.edu

Community Grids Lab
Indiana University
Bloomington, IN 47404 USA

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 key scientific computing areas like 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 [Fielding2000] when applied to network application programming), and more tightly coupled distributed object systems such as the Object Management Group's CORBA [CORBA], Sun Microsystems' Java RMI [RMI], Microsoft's DCOM [DCOM], and the United States Department of Defense's HLA [HLA]. 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 drive a number of fields, such as high energy physics [Allcock2002], meteorology and weather modeling [Gannon2004], and astronomy [Brunner2001], 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 giving the increase in computer performance.

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 [Dongarra2002]. Grid computing is surveyed in [Foster2003] and [Berman2003]. An earlier review of Grid computing is given in [Foster1998].

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" [Atkins2003]. 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" [Berman2003] [UKeS-C] 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 [GapAnalysis]. As we shall see, this seemingly diverse collection of capabilities can be unified into a single coherent picture, based on Service Oriented Architecture principles [Booth2004].

- 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 sec. 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." [Fox2004B]

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 core concepts that include the following [Booth2004] [WSGrids]:

- Desirable capabilities may be accessed through remote services.
- These services define a contract for interaction using the XML-based Web Service Description Language (WSDL) [Christensen2001].
- Requester agents (i.e. clients) interact with these services, or provider agents.
- Requesters and providers communicate by exchanging messages, encoded in the XML-based Simple Object Access Protocol (SOAP) [Gudgin2003].

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 Sec. 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 Sec. 6. Instead, Grids are geared toward resource management across organizations. Parallel computing applications and hardware are examples of these resources as shown in Figure 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 Sec. 6. Workflow builds on classic distributed programming models such as Linda [Carriero1989] and the data flow pioneered by AVS [Upson1989]. This type of Grid

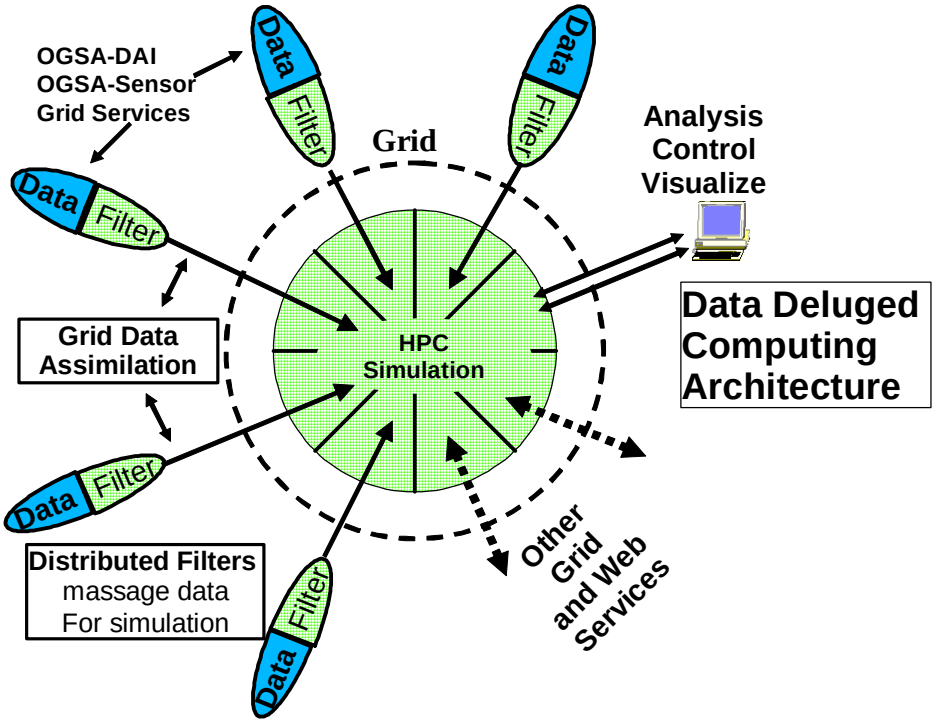

Figure 1: The figure illustrates the hub and spoke architecture for assimilating data using extreme scale computing. Distributed data sources use local clusters for adaptively filtering data, which is then transported using high-performance transport services to one or more centralized high performance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

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 Figure 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 section 6 instead of microsecond messaging speeds in MPI [Fox2004A]. Note the need for latency tolerant algorithms to exploit distributed computing is well known. We don't 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 Sec. 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 Sec. 6 our research on high performance Internet messaging for Grids using the NaradaBrokering 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 [Aktas2004]. See also [QuakeSim].

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 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 [Aydin2005] and more publications and reports are available from [CrisisGrid]. *Web Feature Services* [Vretanos2002] 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 Section 6. *Web Map Services* [Beaujardierre2004] 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* [SensorML] is a family of related specifications for describing sensors. Our efforts here have focused on integrating NaradaBrokering messaging described in Sec. 6 with streaming GPS data.

5.3 Integrating Pattern Informatics and GIS Grid Services: Programming the Grid

The Pattern Informatics (PI) application [Tiampo2002] 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 Sec. 6 is an active field, and overviews are available from [Fox2005A]. 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 is our research effort in this area [HPSearch]. 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 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 [Aydin2005]. 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 Figure 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 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.

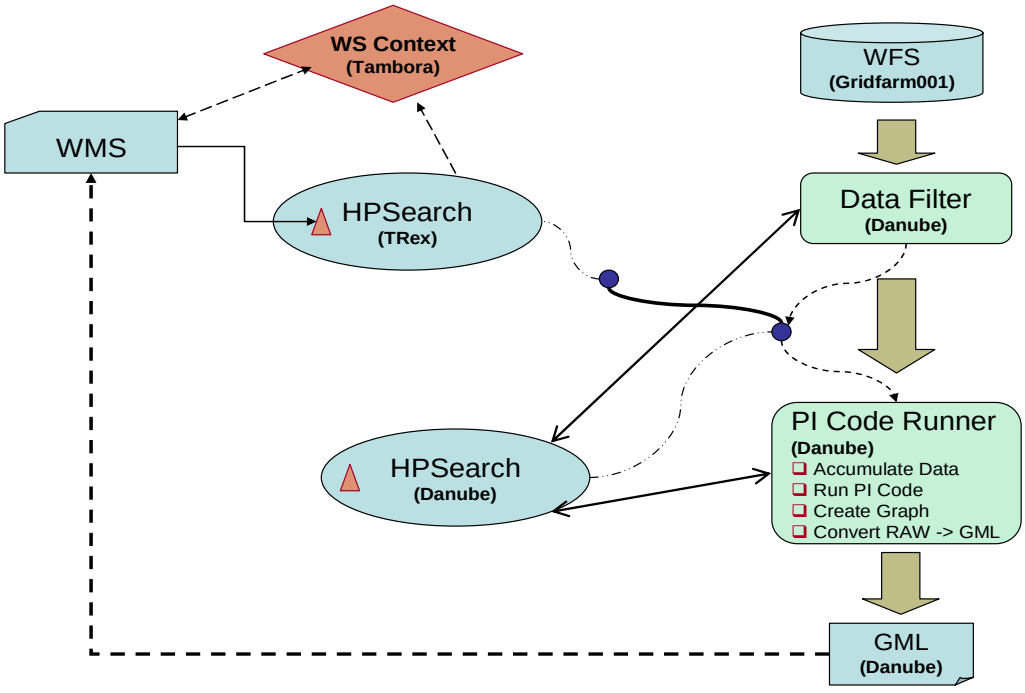

Figure 2 GIS Grid and Execution Grid services may be integrated into meta-applications using workflow orchestration tools such as HPSearch. This represents a "two-level" programming model: applications are

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.

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 [Oh2005].

5.4 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 [Granat2002]. 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 Figure 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.

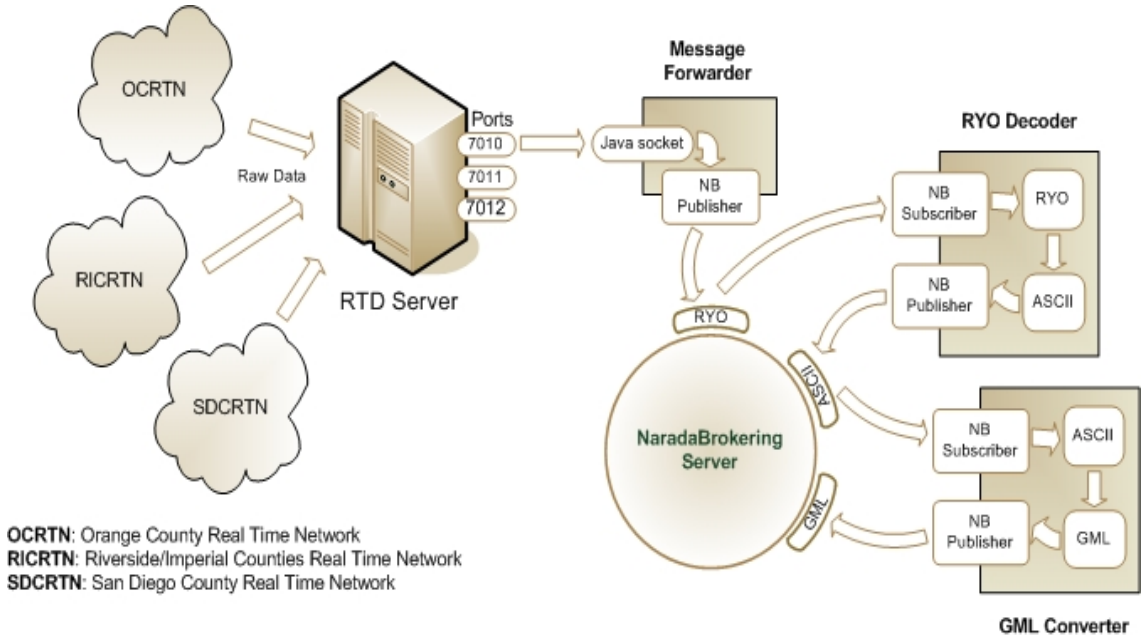

Figure 3 GPS real-time data filtering is accomplished using NaradaBrokering filter chains.

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 Figure 4 for a list of channels. For example, RYO data from the OCRTN network is published to the SOPAC/GPS/Positions/OCRTN/RYO channel. 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 [SensorGrid], which also includes simple demos and downloadable software.

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

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

Returning to Figure 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.

6 Messaging and Problem Solving Environments

6.1 Messaging

MPI and SOAP Messaging both send data from a source to a destination but there are important differences in requirements and functionality [Fox2003B]. In understanding the different approaches to message passing, it is useful to divide multi-processor (distributed memory) and general distributed 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 100-1000 microsecond range typical of simple socket based communication interfaces.
3. Distributed or Grid systems with possibly very high inter-node bandwidth but the latency is typically 100 milliseconds or more as familiar from Internet travel times.

Of course there is really a spectrum of systems with cost-performance increasing by 4 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) – straddling the "Rule of the Millisecond" dividing line [Fox2004A]. MPI especially is aimed at the class (1) with optimized native implementations exploiting particular features of the network hardware. Generic versions of PVM [PVM] and MPI [MPI] using socket based communication on a localized network illustrate (2). One ports MPI to the Grid as in MPICH-G2 [MPICH-G2] and PACX-MPI [PACX] so that one can offer a uniform programming model that supports the low latency MPI on parallel machines. However Grid systems support applications with different requirements than those of parallel computing and hence need messaging with different and greater functionality. Further the larger network latency allows one to add additional software functionality without adding significant overhead.

In fact Grid messaging is more naturally related to message oriented middleware (MOM) [Bernstein1996] [MOM] and software overlay networks [Doval2003] than to MPI or PVM style messaging. The rich MOM infrastructure support messages that are self-describing and as well capture semantic intent. 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. The SOAP messaging used by Web Services illustrates this richer structure with a body containing the "real message" and separate headers corresponding to systems services including security, notification, virtualized addressing, fault tolerance, notification and special operations and relationship to resources. Processing this additional information typically takes a millisecond or so and adds significant functionality at insignificant fractional cost. In designing systems and algorithms for distributed systems it is important to understand this additional functionality and incorporate it into your architecture. Another difference between SOAP and MPI is that MPI only specifies interface and so interoperability between different MPI implementations requires additional work [IMPI]. SOAP however specifies both interfaces and message structure so broad interoperability can be achieved. This is a achieved at a performance cost that we return to in Sec 6.5.

6.2 NaradaBrokering

We have built a MOM style messaging system – NaradaBrokering [Pallickara2003, NaradaBrokering] –- that is designed to support traditional Web Services and SOAP but has an architecture that allows it to support high performance data transport as required for Scientific applications and described briefly in Sec. 6.5 [Fox2005B]. We suggest using this system whenever your application can tolerate 1-100 millisecond 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 [Happner2000]. It also incorporates support for Web Service specifications such as WS-Eventing [Microsoft2004a], WS-ReliableMessaging [Microsoft2004b] and WS-Reliability [OASIS2004]. 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.

6.3 Workflow as the Programming Model for the Grid

The Web Service (Grid) paradigm implicitly assumes a two-level Programming Mode. Here one builds a Service (similar in this respect to a "distributed object" or "computer program" running on a remote computer) using conventional technologies such as C++ Java or Fortran that provides a computation capability or supports data streaming from sensors and satellites or specialized database access such as JDBC. Such services accept and produce data from user files and databases. The Grid is built by coordinating such services assuming we have solved problem of programming the service internally.

The Grid discusses the composition of distributed services with the runtime interfaces to Grid extending a model familiar from UNIX pipes and data streams where one can use UNIX Shell, PERL or Python scripts to produce complex 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 most efforts in the community look at each level (internal and external to services) separately [GrADS].

In programming SOAP and Web Services (the Grid) workflow describes linkage between services [Yu2005]. Since the various services, resources and other components are distributed, the linkage between them must be through the exchange of messages. This linkage is two-way and has both control and data using SOAP as described in Sec 6.1. The BPEL specification from Microsoft-IBM [BPEL4WS] is currently the preferred Web Service XML specification of workflow

6.4 Problem Solving Environments

Problem Solving Environments (PSEs) are environments integrating tools to tackle a particular class of problems with for example SERVOGrid as a PSE for earthquake science and PSEs are similar to portals familiar from Yahoo and other major Internet sites. There is a major effort [Fox2003A] [OGCE] to build Grid portals that are essentially PSEs to a suite of distributed services. We note that traditional PSEs used scripting languages to develop such environments while Grid portals are often implemented using messaging and workflow for the integration. This is an interesting example of the law of the millisecond discussed above. Should one use the sub –microsecond latency of scripting languages (using method call not message based component integration) or the one millisecond (local services) to many millisecond (distributed services) latency of messaging?

Messaging has several advantages over scripting languages. First, collaboration between the distributed entities is trivial since it simply involves sharing of messages [Qiu2005]. 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. This greater modularity for messaging correspond to an easier (compared to a scripting approach) support model to both maintain an existing PSE and in particular to upgrade particular PSE components. We suggest looking at the system requirements and always linking components with the highest level (slowest) and hence easiest to support approach compatible with these requirements. More often and not PSE components can be linked by messaging and where the inevitable messaging overhead can be tolerated, we recommend using this and building components as services. We call this the “simple service approach” [Fox2004B] where one chooses to build services as small as is possible compatible with communication and bandwidth constraints. Both in parallel and distributed computing, the ratio of communication to computation in a service or parallel component depend on the ratio of surface to volume for the component which decreases as the size (volume) of the component increases [Fox1994].

6.5 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 [Pallickara2005], since the dominant specifications WS-Reliability and WS-ReliableMessaging [Microsoft2004b] [OASIS2004] both mandate the storage of the SOAP message to stable storage prior to forwarding the message.

Here we will leverage the XML and SOAP Infosets [Gudgin2003] [MSBinaryXML]; by separating the SOAP message context from its XML syntax, we can freely move between the binary and classic angle-

bracketed representations of SOAP messages without content loss. We faithfully preserve the SOAP semantics but transport an equivalent binary optimized binary representation. There are several such efficient representations of XML Infosets including SOAP Message Transmission Optimization Mechanism [MTOM] and XML-binary Optimized Packing [XOP]. We are developing schemes which allow two endpoints to first negotiate the best-available transport using the fully interoperable SOAP representation [Oh2005] and then proceed to use it for transfers. To specify this process and to accommodate legacy systems that do not use the XML format, the Data Format Description Language [DFDL] is an XML-based language that describes the structure of binary and character-encoded files and data streams so that their format, structure, and metadata can be exposed. This can also be used in tandem while transferring binary data using SOAP. The NaradaBrokering substrate incorporates support for several transport protocols that can be leveraged to provide appropriate high performance protocols for the transfers of such large binary data. Currently one can choose between TCP, parallel TCP and UDP combined with SOAP level reliable messaging for fault tolerance. Since the substrate allows new transport protocols to be plugged in rather easily, support for newer transport schemes can easily be incorporated as they get developed.

We believe that these strategies will at the cost of upfront overhead at the start of a stream allow systems like NaradaBrokering transport SOAP compatible messages with high bandwidth and a latency that will be insignificant on long streams. Of course sensors or the data from instruments always give such streams of data and so we believe Web services will support them with high performance.

7 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 corresponds to a “two-level” programming model: components may be developed as parallel services implemented by conventional programming models, but the interaction of the services is specified by Grid workflow. Algorithms must be designed that support this rule.

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, and by efficient representations as supported by the NaradaBrokering system. This leads to latencies of a millisecond (close by or within same machine) to 100 millisecond (distant) in communication but potentially very high bandwidths that should be respected by algorithms.

8 Acknowledgements

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).

9 References

[Aktas2004] Mehmet Aktas, Galip Aydin, Andrea Donnellan, Geoffrey Fox, Robert Granat , Lisa Grant, Greg Lyzenga, Dennis McLeod, Shrideep Pallickara, Jay Parker, Marlon Pierce, John Rundle, Ahmet Sayar, and Terry Tullis *iSERVO: Implementing the International Solid Earth Research Virtual*

Observatory by Integrating Computational Grid and Geographical Information Web Services Technical Report December 2004. Accepted for publication in Special Issue of Pure and Applied Geophysics (PAGEOPH) for Beijing ACES Meeting July 2004 http://grids.ucs.indiana.edu/ptliupages/publications/ISERVO_ACES_PAGEOPH.pdf

[Allcock2002] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke "Data Management and Transfer in High Performance Computational Grid Environments." Parallel Computing Journal, Vol. 28 (5), May 2002, pp. 749-771.

[Atkins2003] D. E. Atkins, K. K. Droegemeier, S. I. Feldman, H. Garcia-Molina, M. L. Klein, D. G. Messerschmitt, P. Messina, J. P. Ostriker, and M. H. Wright, "Revolutionizing Science and Engineering Through Cyberinfrastructure." Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure, January 2003. Available from http://www.nsf.gov/cise/sci/reports/atkins.pdf.

[Aydin2005] Galip Aydin, Mehmet S. Aktas, Geoffrey C. Fox, Harshawardhan Gadgil, Marlon Pierce, Ahmet Sayar *SERVOGrid Complexity Computational Environments (CCE) Integrated Performance Analysis* Technical report June 2005. http://grids.ucs.indiana.edu/ptliupages/publications/gwpap243.pdf .

[Barish1999] Barish, B. C. and Weiss, R., Physics Today, Oct 1999, pp. 44-50; also http://www.ligo.caltech.edu/

[Beaujardierre2004] de La Beaujardiere, Jeff, Web Map Service, OGC project document reference number OGC 04-024.

[Berman2003] Berman, F., Fox, G., and Hey, T., (eds.). Grid Computing: Making the Global Infrastructure a Reality, John Wiley & Sons, Chichester, England, ISBN 0-470-85319-0 (2003). http://www.grid2002.org.

[Bernstein1996] Bernstein, Philip A. "Middleware: A Model for Distributed System Services," Communications of the ACM, February 1996, Vol. 39, No.2, pgs. 86 - 98.

[Booth2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. "Web Service Architecture." W3C Working Group Note, 11 February 2004. Available from http://www.w3c.org/TR/ws-arch.

[BPEL4WS] BPEL4WS: F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, *BPEL4WS, Business Process Execution Language for Web Services*, Version 1.0. Available from http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.

[Brunner2001] R. J. Brunner, S. G. Djorgovski, A. S. Szalay, eds., "Virtual Observatories of the Future," Astronomical Society of the Pacific Conference Series Vol 225, 2001.

[Carriero1989] Nicholas Carriero and David Gelernter *Linda in Context*, Communications of the ACM 32 , 4 (April 1989) pages 444 - 458

[Christensen2001] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001), Web Service Description Language (WSDL) 1.1. W3C Note 15 March 2001.

[CORBA] Common Object Request Broker Architecture Web Site: http://www.corba.org See also the Object Management Group's site, http://www.omg.org.

[CrisisGrid] GIS Research at the Community Grids Laboratory: http://www.crisisgrid.org.

[DCOM] Microsoft Distributed Component Object Model Technologies Web Site: http://www.microsoft.com/com/default.mspx.

[DFDL] The Data Format Description Language (DFDL) working group. https://forge.gridforum.org/projects/dfdl-wg/.

[Dongarra2002] *The Sourcebook of Parallel Computing*, Jack Dongarra, Ian Foster, Geoffrey Fox, William Gropp, Ken Kennedy, Linda Torczon, Andy White, eds. Morgan Kaufmann, 2002. http://www.mkp.com/books_catalog/catalog.asp?ISBN=1-55860-871-0.

[Doval2003] OverLay Networks: A Scalable Alternative for P2P. Diego Doval and Donal O'Mahony. IEEE INTERNET COMPUTING. July August 2003.

[Fielding2000] Roy T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, Dissertation, University of California, Irvine, 2000. Available from http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[Foster1998] Foster, I. and Kesselman, C., (eds.). The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann (1998).

[Foster2004] Foster, I. and Kesselman, C., (eds.) The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann (2004).

[Fox1994] Fox, G. C., Messina, P., Williams, R., "Parallel Computing Works!", Morgan Kaufmann, San Mateo Ca, 1994.

[Fox2003A] Geoffrey Fox, Dennis Gannon and Mary Thomas, *Overview of Grid Computing Environments*, Chapter 20 of [Berman2003].

[Fox2003B] Geoffrey Fox Messaging Systems: *Parallel Computing, the Internet and the Grid* Proceedings of EuroPVM/MPI 2003, pages 1-9 of Recent Advances in Parallel Virtual Machine and Message Passing Interfaces, edited by Dongarra, Laforenza, Orlando, Springer LNCS 2840, September 30 2003, http://grids.ucs.indiana.edu/ptliupages/publications/gridmp_fox.pdf

[Fox2004A] Geoffrey Fox, "Software Development Around a Millisecond" Computers in Science and Engineering March/April 2004 p93-96 http://grids.ucs.indiana.edu/ptliupages/publications/cisejano4.pdf

[Fox2004B] Geoffrey Fox, "Grids of Grids of Simple Services" Computers in Science and Engineering July/August 2004, p84-87 http://grids.ucs.indiana.edu/ptliupages/publications/Cisegridofgrids.pdf

[Fox2005A] Geoffrey Fox and Dennis Gannon (eds). *Concurrency and Computation: Practice and Experience*. Upcoming special issue on Grid workflow based on GGF10 meeting at Berlin http://www.extreme.indiana.edu/groc/ggf10-ww/. Editorial is *Workflow in Grid Systems* http://grids.ucs.indiana.edu/ptliupages/publications/Workflow-overview.pdf.

[Fox2005B] Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, Harshawardhan Gadgil, *Building Messaging Substrates for Web and Grid Applications* to be published in special Issue on Scientific Applications of Grid Computing in Philosophical Transactions of the Royal Society of London 2005 http://grids.ucs.indiana.edu/ptliupages/publications/RS-CGL-ColorOnlineSubmission-Dec2004.pdf

[Gannon2004] D. Gannon, S. Krishnan, L. Fang, G. Kandaswamy, Y. Simmhan, A. Slominski, On Building Parallel and Grid Applications: Component Technology and Distributed Services, Proceedings, Challenges of Large Applications in Distributed Environments (CLADE) In conjunction with the 13th International Symposium on High Performance Distributed Computing (HPDC-13), pp. 44-51, June, 2004.

[GapAnalysis] Geoffrey Fox, David Walker, *e-Science Gap Analysis*, June 30 2003. Report UKeS-2003-01, http://www.nesc.ac.uk/technical_papers/UKeS-2003-01/index.html.

[GrADS] GrADS (Grid Application Development Software Project http://www.hipersoft.rice.edu/grads/

[Granat2002]Granat, R., and A. Donnellan, A hidden Markov model tool for geophysical data exploration, *Pure and Appl. Geophys*, 2271–2284, 2002.

[Gudgin2003] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., and Nielsen, H. (2003), SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation 24 June 2003. Available from http://www.w3c.org/TR/soap12-part1/.

[Happner2000] Mark Happner, Rich Burridge and Rahul Sharma. Sun Microsystems. Java Message Service Specification. 2000. http://java.sun.com/products/jms

[HLA] High Level Architecture (HLA), Defense Modeling and Simulation Office, United States Department of Defense Web Site: https://www.dmso.mil/public/transition/hla/.

[HPSearch] Grid Service workflow and management http://www.hpsearch.org

[IMPI] Interoperable MPI http://impi.nist.gov/IMPI/.

[Microsoft2004a] Web Services Eventing. Microsoft, IBM & BEA. http://ftpna2.bea.com/pub/downloads/WS-Eventing.pdf

[Microsoft2004b] Web Services Reliable Messaging Protocol (WS-ReliableMessaging) ftp://www6.software.ibm.com/software/developer/library/ws-reliablemessaging200403.pdf

[MOM] http://dsonline.computer.org/middleware/intro_MOM.html

[MPI] The Message Passing Interface Standard http://www-unix.mcs.anl.gov/mpi/standard.html

[MPICH-G2] MPICCH-G2: Grid-enabled implementation of the MPI v1.1 standard. http://www.nsf-middleware.org/NMIR3/components/mpichg2.asp

[MSBinaryXML] Adam Bosworth, Don Box, Martin Gudgin, Mark Nottingham, David Orchard, Jeffrey Schlimmer, Microsoft and BEA, *XML, SOAP, and Binary Data* http://msdn.microsoft.com/webservices/webservices/understanding/specs/default.aspx?pull=/library/en-us/dnwebsrv/html/infoset_whitepaper.asp

[MTOM] SOAP Message Transmission Optimization Mechanism. Microsoft, IBM and BEA. http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/

[NaradaBrokering] The NaradaBrokering Web Site: http://www.naradabrokering.org.

[OASIS2004] Web Services Reliable Messaging TC WS-Reliability. http://www.oasis-open.org/committees/download.php/5155/WS-Reliability-2004-01-26.pdf

[OGCE] Open Grid Computing Environment OGCE Portal Collaboration http://www.collab-ogce.org/nmi/index.jsp

[Oh2005] Sangyoon Oh, Hasan Bulut, Ahmet Uyar, Wenjun Wu, Geoffrey Fox *Optimized Communication using the SOAP Infoset For Mobile Multimedia Collaboration Applications* Proceedings of the International Symposium on Collaborative Technologies and Systems CTS05 May 2005, St. Louis Missouri, USA. http://grids.ucs.indiana.edu/ptliupages/publications/OptSOAP_CTS05.pdf

[PACX-MPI] Extending MPI for Computational Grids. Project Webpage http://www.hlrs.de/organization/pds/projects/pacx-mpi/.

[Pallickara2003] Shrideep Pallickara and Geoffrey Fox *NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids*. Proceedings of ACM/IFIP/USENIX International Middleware Conference Middleware-2003. pp 41-61. Lecture Notes in Computer Science 2672 Springer 2003, ISBN 3-540-40317-5 http://grids.ucs.indiana.edu/ptliupages/publications/NB-Framework.pdf

[Pallickara2005] Shrideep Pallickara, Geoffrey Fox, Beytullah Yildiz, Sangmi Lee Pallickara, Sima Patel and Damodar Yemme An Analysis of the Costs for Reliable Messaging in Web/Grid Service Environments Technical report June 2005 http://grids.ucs.indiana.edu/ptliupages/publications/Wsrm-Performance.pdf

[PVM] The Parallel Virtual Machine. http://www.netlib.org/pvm3/

[Qiu2005] Xiaohong Qiu *Message-based MVC Architecture for Distributed and Desktop Applications* Syracuse University PhD March 2 2005 http://grids.ucs.indiana.edu/ptliupages/publications/qiuPhDthesis.pdf.

[QuakeSim] The QuakeSim Project Web Site: http://quakesim.jpl.nasa.gov/

[RMI] Java Remote Method Invocation (Java RMI) Web Site: http://java.sun.com/products/jdk/rmi/.

[SensorGrid] Sensor Grid Research at the Community Grids Laboratory: http://www.crisisgrid.org/html/sensorgrid.html.

[SensorML] Sensor Model Language (SensorML) Project Web Site: http://vast.nsstc.uah.edu/SensorML/.

[Tiampo2002] Tiampo, K. F., Rundle, J. B., McGinnis, S. A., & Klein, W. Pattern dynamics and forecast methods in seismically active regions. *Pure Ap. Geophys.* 159, 2429-2467 (2002).

[UKeS-C] NeSC: National e-Science centre for UK program at Edinburgh, http://www.nesc.ac.uk/

[Upson1989] Craig Upson, Thomas Faulhaber, Jr., David H. Laidlaw, David Schlegel, Jefrey Vroom, Robert Gurwitz, Andries van Dam, The Application Visualization System: A Computational Environment for Scientific Visualization, IEEE Comput. Graph. Appl., vol. 9, no. 4, 1989, pp 30-42. IEEE Computer Society Press.

[Vretanos2002] Vretanos, P (ed.) (2002), Web Feature Service Implementation Specification, OpenGIS project document: OGC 02-058, version 1.0.0.

[WSGrids] M. Atkinson et al., 'Web Service Grids: An evolutionary approach', Concurrency and Computation: Practice and Experience 17, 377-389, 2005; http://www.nesc.ac.uk/technical_papers/UKeS-2004-05.pdf (defines WS-I+)

[XOP] XML-binary Optimized Packing. Microsoft, IBM and BEA. http://www.w3.org/TR/2005/REC-xop10-20050125/.

[Yu2005] Jia Yu and Rajkumar Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, Technical Report, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005. http://www.gridbus.org/reports/GridWorkflowTaxonomy.pdf