Innovative Grid Technologies Applied to Bioinformatics and Hurricane Mitigation

Rosa BADIA [a] Gargi DASGUPTA [b] Onyeka EZENWOYE [c] Liana FONG [d]
Howard HO [e] Sawsan KHURI [f] Yanbin LIU [d] Steve LUIS [c] Anthony PRAINO [d]
Jean-Pierre PROST [g] Ahmed RADWAN [f] Seyed Masoud SADJADI [c]
Shivkumar SHIVAJI [e] Balaji VISWANATHAN [b]
Patrick WELSH [h] Akmal YOUNIS [f]

[a] *Barcelona Supercomputing Center, Barcelona, Spain*
[b] *IBM India Research Laboratory, NewDelhi, India*
[c] *Florida International University, Miami, FL*
[d] *IBM T.J. Watson Research Center, Yorktown Heights, NY*
[e] *IBM Almaden Research Center, San Jose, CA*
[f] *University of Miami, Miami, FL*
[g] *IBM Products and Solutions Support Center, Montpellier, France*
[h] *University of North Florida, Jacksonville, FL*

Abstract. The Latin American Grid (LA Grid) joint research program fosters collaborative research across eleven universities and IBM Research with the objective of developing innovative grid technologies and applying them to solve challenging problems in the application areas of bioinformatics and hurricane mitigation. This paper describes some of these innovative technologies, such as the support for transparent to the application expert grid enablement, meta-scheduling, job flows, data integration, and custom visualization, and shows how these technologies will be leveraged in the LA Grid infrastructure to provide solutions to pharmagenomics problems and hurricane prediction ensemble simulations.

Keywords. Meta-scheduling, job flows, data integration, transparent grid enablement, custom visualization, bioinformatics, hurricane mitigation.

Introduction

Since December 2005, IBM has been engaged with academic partners in Florida, Puerto Rico, Mexico, Argentina, and Spain in the Latin American (LA) Grid initiative to dramatically increase the quantity and quality of Hispanic Technical Professionals entering the Information Technology fields.

At the core of this initiative is the development of a computer grid across multiple universities and businesses that serves as the platform for education and collaborative research in the critical and emerging fields of grid computing, distributed systems and supercomputing. The LA Grid constitutes a living laboratory for advanced research by the universities and IBM Research in application areas such as bioinformatics, hurricane mitigation, and healthcare.

In our ongoing research efforts, we aim to address application area problems with current state-of-the-art grid solutions by employing a top-down approach that provides the right level of abstraction for the domain experts while factoring out the common services that can be reused across domains. Among these common services, our focus has been on providing support for transparent grid enablement, meta-scheduling, job flows, data integration, and custom visualization. In this paper, we first describe the innovative technologies we developed in order to provide such support, and then we illustrate how these technologies can be leveraged towards the resolution of challenging problems in the area of bioinformatics and hurricane mitigation.

The paper is structured as follows. In Section one, we introduce the concepts of grid superscaling and transparent adaptation and show how these concepts can be combined to provide transparent grid enablement. In Section two, we detail our design for meta-scheduling and our on-going prototyping activities in that space. In Section three, we outline our approach to job flows, leveraging WS-BPEL and providing support for fault-tolerant job flows through a wrapping layer. In Section four, we describe the set of services we have been developing and the architecture we designed to provide data integration capabilities in grid environments. In Sections five and six, we show how our innovative technologies are being applied to address challenging scenarios in bioinformatics and hurricane mitigation. Section seven concludes this paper with our future plans towards the creation of a transparent grid environment, which will allow domain experts to express application logic using an appropriate visual interface while making transparent, to the greatest extent possible, the details of the grid hardware and middleware stack.

1. Transparent Grid Enablement

The advent of cluster and grid computing has created a remarkable interest in high performance computing (HPC) both in academia and industry, especially as a solution to complex scientific problems (e.g. bioinformatics and hurricane mitigation applications). To efficiently utilize the underlying HPC facilities using the current programming models and tools, however, scientists are expected to develop complex parallel programs; a skill that they might not necessarily have and is better done by HPC experts.

Current standards for cluster and grid programming such as MPI [1], OGSA [2], and WSRF [3] (and their implementations in offerings like MPICH2 [4], the Globus Toolkit [5], Unicore [6], and Condor [7]; to name just a few) have provided scientists with higher levels of abstraction. Noteworthy, these approaches have been successful in hiding the heterogeneity of the underlying hardware devices, networking protocols, and middleware layers from the scientist developer. However, the scientist is still expected to develop complex parallel algorithms and programs. Moreover, as the code for parallel algorithms typically crosscuts the code for business logic of the application, the resulting code is entangled and is difficult to maintain and evolve.

In this part of our research, we address these problems by enabling a separation of concerns in the development and maintenance of the non-functional aspects (e.g. the performance optimization) and the functional aspects (i.e. the business logic) of scientific applications. We achieve this goal by integrating two existing programming tools, namely, a Grid framework, called *GRID superscalar* [8], and an adaptation-enabling tool, called *TRAP/J* [9]. On one hand, GRID superscalar enables the

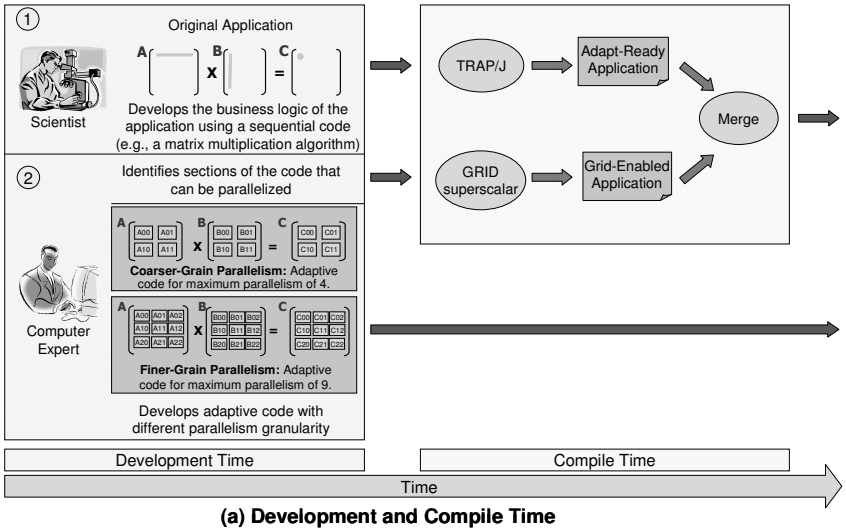
development of applications for a computational grid by hiding details of job deployment, scheduling, and dependencies and enables the exploitation of the concurrency of these applications at runtime. On the other hand, TRAP/J supports automatic weaving of alternative parallel code (including the corresponding calls to GRID superscalar runtime) into the sequential code developed by the scientist to support static and dynamic adaptation to heterogeneous grid computing environments.

1.1. Overview

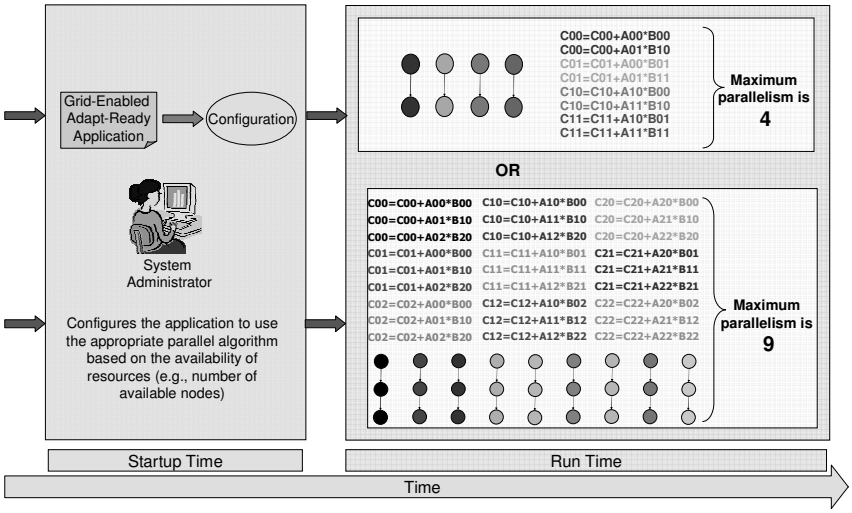
Inspired by the *superscalar* processors, GRID superscalar provides an easy programming paradigm for developing parallel programs [8]. Similar to superscalar processors that provide out-of-order and parallel execution of machine instructions by bookkeeping their dependencies, GRID superscalar provides parallelism to the functions of a program written in a high-level programming language such as Java. Using GRID superscalar, a sequential scientific application developed by a scientist is dynamically parallelized in a computational Grid. GRID superscalar hides the details such as resource mapping, staging input data files, cleaning temporary data files, deploying and scheduling tasks, exploiting instruction-level parallelism, and exploiting data locality. We note that for many of its responsibilities, GRID superscalar depends on other grid computing toolkits such as GT4, Condor, and others.

TRAP/J is a tool that enables static and dynamic adaptation in Java programs at startup and runtime, respectively [9]. It consists of two GUI-based interactive tools as follows: (1) the *Generator*, which generates an adapt-ready version of an existing application by inserting generic hooks into a pre-selected subset of classes in the application, called *adaptable* classes; and (2) the *Composer,* which allows insertion of new code at the generic hooks both at startup or runtime. Adaptable behavior is provided through alternative implementations of adaptable classes. To replace alternative parallel algorithms developed using the GRID superscalar codes, we use the Generator to make the classes with sequential code adaptable, and then we use the Composer to weave in the parallel code.

Each tool provides us with the necessary features for transparent software adaptation from a sequential code to a grid-enabled one. Figure 1 illustrates the operation of our transparent grid enablement framework in the context of a simple case study, during which a sequential matrix multiplication program (developed in Java) is transparently adapted to run in a grid computing environment. First, we use GRID superscalar to develop alternative hyper-matrix multiplication algorithms by splitting the original matrices into a number of sub-matrices or blocks (Figure 1 (a), development time). Therefore, instead of just one task as in the original approach, using hyper-matrix multiplication and GRID superscalar, up to 4 tasks can be active at the same time. Of course, if we split the matrix into 9 blocks, then up to 9 tasks can be executed at the same time and so on and so forth. Next, using TRAP/J and GRID superscalar code generators, an adapt-ready version of the application is generated (Figure 1 (a), compile time). Next, a system administrator (or an intelligent software agent) configures the application to use the appropriate parallel algorithm based on the availability of resources, for example, the number of available nodes (Figure 1 (b), startup time). Finally, the GRID superscalar code—woven into the application using TRAP/J—will exploit the task-level parallelism by resolving the dependencies of the tasks, each performing multiplication of sub-matrices accumulatively (Figure 1 (b), runtime).



(a) Development and Compile Time

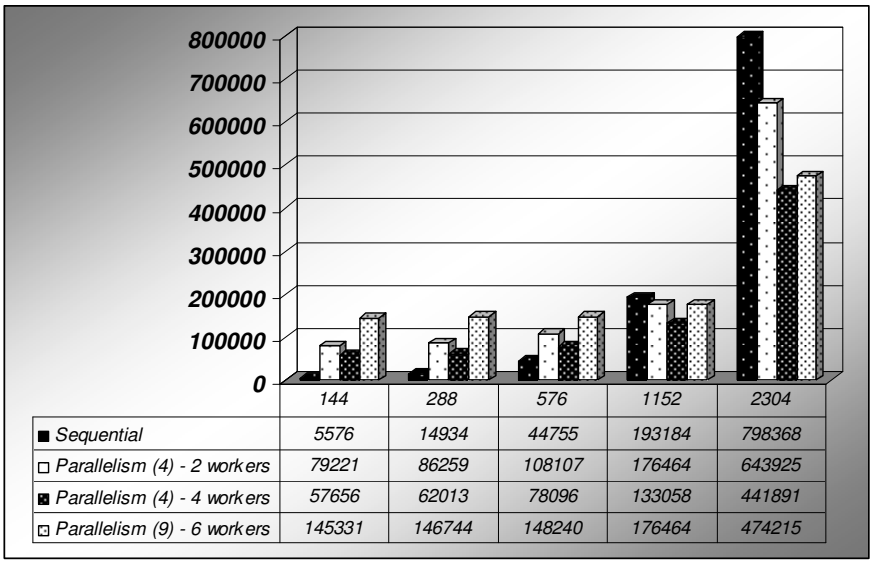


(b) Startup and Run Time

Figure 1. Grid enablement approach for the matrix multiplication case study

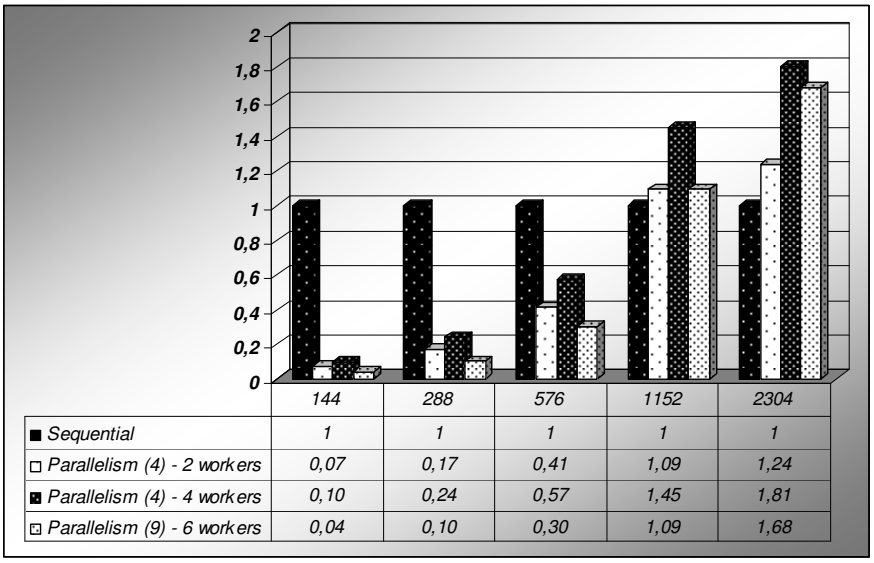
1.2. Experimental Results

To show the validity of our approach, we conducted a set of experiments that measure the speedup gained as a result of the grid enablement of the matrix multiplication application. The results are illustrated in Figure 2.



	144	288	576	1152	2304
Sequential	5576	14934	44755	193184	798368
Parallelism (4) - 2 workers	79221	86259	108107	176464	643925
Parallelism (4) - 4 workers	57656	62013	78096	133058	441891
Parallelism (9) - 6 workers	145331	146744	148240	176464	474215

(a) Execution time



	144	288	576	1152	2304
Sequential	1	1	1	1	1
Parallelism (4) - 2 workers	0,07	0,17	0,41	1,09	1,24
Parallelism (4) - 4 workers	0,10	0,24	0,57	1,45	1,81
Parallelism (9) - 6 workers	0,04	0,10	0,30	1,09	1,68

(b) Speedup

Figure 2. Execution time and speedup of grid-enabled versions of the matrix multiplication application

In Figure 2, we see that as the matrix size increases, the speedup improves and when the size of the matrix reaches 1152 (number of rows = number of columns = 1152), all the algorithms for the grid-enabled application perform better than the original sequential application. We notice that the algorithm, which uses 4 blocks on 4 worker nodes, exhibits the best performance. For a matrix of size 2304, it performs

almost twice as fast as the sequential application. This is because of the even distribution of the load and a one-to-one mapping of the blocks onto the worker nodes, which result in more parallelism and less communication overhead related to the file transfer between the nodes required both at the initialization and finalization stages.

We emphasize that these experiments are part of our ongoing research activities and they are not meant to be representative nor conclusive with respect to providing a quantitative metric for speedup gained because of the grid enablement. The main purpose of these experiments is to show that we were able to use our current transparent grid enablement prototype to transparently adapt an application to run in a grid computing environment.

1.3. Related Work

Satin [10] is a Java based programming model for the Grid which allows for the explicit expression of divide-and-conquer parallelism. HOCS [11] is a component oriented approach based on a master-worker schema. ASSIST [12] is a programming environment that aims at providing parallel programmers with user-friendly, efficient, portable, fast ways of implementing parallel applications. ProActive [13] is a Java grid middleware library for parallel, distributed and multi-threaded computing. Unlike our transparent grid enablement framework, none of the above mentioned approaches provide an explicit separation of concerns identifying separate tasks for scientist developers and HPC expert developers. Our framework can be extended to use these methods instead of or in complement to GRID superscalar and could be used as an enabler for supporting interoperation among the above mentioned approaches.

1.4. Future Work

As we mentioned before, we have been able to achieve *static* adaptation. Our next task will be to extend our framework in support of more autonomic behavior and include adaptation at runtime (dynamic) in response to high level system policies such as the addition of more nodes to the grid, a change in process scheduling or application level policies such as different blocking algorithms or faster algorithms. At present, dynamic adaptation of Java programs with TRAP/J is under progress and is being tested. Furthermore, moving towards building a more autonomic self adapting and self configuring system, we can expand our framework to provide context-aware adaptation, by keeping track of the state of the runtime environment and retrieve information about resource allocation, scheduling, etc.

2. Meta-scheduling

Over the past two decades, computing power has become increasingly available. However, the demand for computing power, driven by many new applications in bioinformatics, healthcare, physical science simulation, supply chain modeling, and business intelligent decision, still surpasses the supply. Grid computing allows harnessing of available computing resources from cooperating organizations or institutes, in the form of virtual organizations (VOs), in order to satisfy user demands and share the cost of ownership of the resources.

The first generation of grid technologies and infrastructures focused on harnessing the computational power of machines. With the evolution and availability of grid infrastructures, today's grid technologies further enhance the collaboration of users by providing easy access to greater varieties of resources, such as data and software services. For examples, the collection of astronomy data at one observatory can be easily made available to scientists around the world, the BioMOBY web services in Taverna [14] are used to support publishing and extracting of biological data.

At the core of grid technology is a resource brokering component, commonly known as meta-scheduler or meta-broker. The meta-scheduler matches user work requests to appropriate resources in the VO to execute the requests. In addition to the challenges of managing VO resources that have dynamic availability attributes, meta-schedulers need to take into account the resource usage policies and security restrictions enforced by the local schedulers, which they interact with.

Currently, there are many studies and systems related to meta-scheduling in the grid community. As discussed by Subramani et al. [15], most meta-schedulers can be classified into three different models:

- Centralized model: one meta-scheduling component has direct information of all resources available at the various institutes of the virtual organization and is responsible for scheduling job execution on all resources; local schedulers at individual institutes will act as job dispatchers. An exemplary system of this model is eNANOS [16].
- Hierarchical model: one meta-scheduling component has no direct access to resources in the virtual organization, but assigns jobs to the local schedulers of the various institutes; local schedulers will match jobs to resources. An exemplary system of this model is the Community Scheduling Framework (CSF) [17].
- Distributed model: multiple local schedulers exist in a VO; each local scheduler has a companion meta-scheduling functional entity; local schedulers can submit jobs to each others through their respective meta-scheduling functional entities. Exemplary systems of this model include IBM Tivoli Workload Scheduler Loadleveler [18] and Gridway [19].

All three models have their respective advantages and disadvantages and are suitable in different deployment environments. The centralized model is relatively simpler than other models. However, the meta-scheduler can become a bottleneck for a VO that has a very large number of resources. The meta-scheduler can also be a potential single point of failure. The hierarchical model is a more scalable scheme than the central model, but the meta-scheduler has less control of the scheduling decisions and can still be a single point of failure. The distributed model is the most complex of the three models and does not present bottleneck and single point of failure exposures.

For LA Grid, our meta-scheduling design is a mix of the hierarchical and distributed models, as shown in Figure 3.

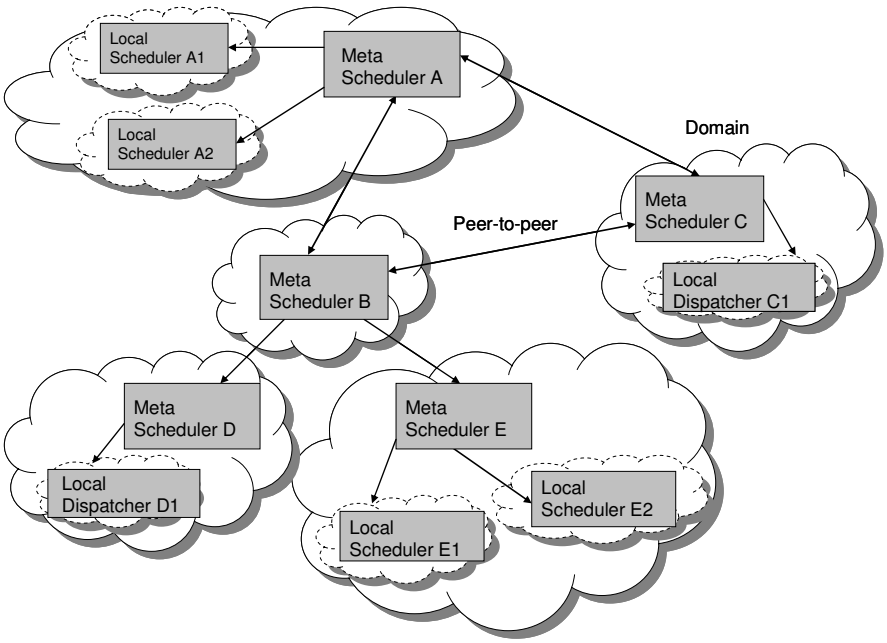


Figure 3. LA Grid meta-scheduling architecture

Our grid model consists of multiple domains. Each domain has its domain meta-scheduler and consists of a collection of local dispatchers, local schedulers or even other meta-schedulers. A domain can be viewed as the meta-scheduling functional entity of an institution. This aspect of our model intends to reflect the reality of many organizations having multiple local schedulers for different lines of business or for various levels of services. The domain meta-scheduler supports the encapsulation of resources and scheduling details within each organization and only its aggregate scheduling capability and capacity would be exposed to the partner organizations of the grid. Then, the grid can be viewed as a sphere with collaborative partners. The peer to peer relationship between domain meta-schedulers is dynamically established upon the agreement between peers. Users of a domain would interact with that specific domain meta-scheduler to access resources of collaborative partners.

The following example illustrates the use of our meta-scheduling model. A bioinformatics data service provider differentiates its services to paying and trial customers. Paying customers are users of a consumer organization that pays for the bioinformatics services. They typically get an expected quality of service as well as access to the full set of databases. Multiple sites are set up to support paying customers with guaranteed service quality and availability. In contrast, trial customers are users without any organization association or users from a non-paying organization. They typically get service on a best effort basis on a single site with access to only sample databases. Using our meta-scheduling model, this bioinformatics provider would set up a meta-scheduling domain with a single local scheduler for trial customers. For the paying customers, the provider would either set up a sub-domain meta-scheduler with multiple local schedulers for each site, or include all the local schedulers in the same domain with a single meta-scheduler as for trial customers. For a consumer

organization, the bioinformatics data service is made available to its users (e.g. by including the data service in the service registry). Depending on the demands of its users and budgetary constraints, the organization would establish either a paying or a non-paying peer-to-peer relationship with the bioinformatics provider. For the users in the consumer organization, their application logic would not be affected by the paying status of the organization.

The Open Grid Forum [20] is leading the effort of defining a standard description language for job submission, called Job Submission Description Language (JSDL) [21]. This language allows specification of job characteristics as well as resources required for the job execution in a grid infrastructure. The adoption of JSDL is a good first step towards achieving collaboration across virtual organizations from a job execution standpoint. However, there is a strong need for a standard interface for expressing meta-scheduler to meta-scheduler interactions and meta-scheduler to local scheduler interactions to realize collaborative job execution. One of our project objectives is to experiment with the necessary interfaces to support interactions between domain meta-schedulers and their associated local schedulers. We categorize these interfaces into three sets:

- Meta-scheduler connection API: used to establish and terminate the connection between domain meta-schedulers, either through a peer-to-peer relationship or an up-stream relationship in a hierarchy. Once the connection is established, heart beats are exchanged to guarantee the healthy state of the connection.
- Resource exchange API: used to exchange the scheduling capability and capacity of the domain controlled by the meta-scheduler; the exchanged information can be a complete or incremental set of data.
- Job management API: used to submit, re-route and monitor job executions across the network of (meta-)schedulers.

A domain meta-scheduler supports these three APIs and implements the necessary functions, as illustrated in Figure 4. It is composed of three functional modules: the resource management module, the scheduling module, and the job management module. The resource management module is responsible for resource discovery, resource monitoring and resource information storage. The resource information storage can be either a persistent storage device or a cache device. The scheduling module is responsible for locating suitable resources or a suitable scheduler for each job request. The job management module manages the lifecycle of the job, including the reception of the job request, its routing or dispatching to the matched resources or scheduler, and the monitoring of the job status.

Our current implementation of the meta-scheduling APIs is using grid web services in order to more easily accommodate existing meta-schedulers and integrate them as collaborative job execution partners using the Globus Toolkit. We will verify the possibility for the same set of APIs to be recursively useful regardless of the relationship between meta-schedulers. Our experimentation platform consists of three collaborative meta-scheduling partners: the first one is based on IBM's Tivoli Dynamic Workload Broker [22]; the second one is based on the Barcelona Supercomputing Center's eNANOS broker, and the third one is based on Gridway or CSF.

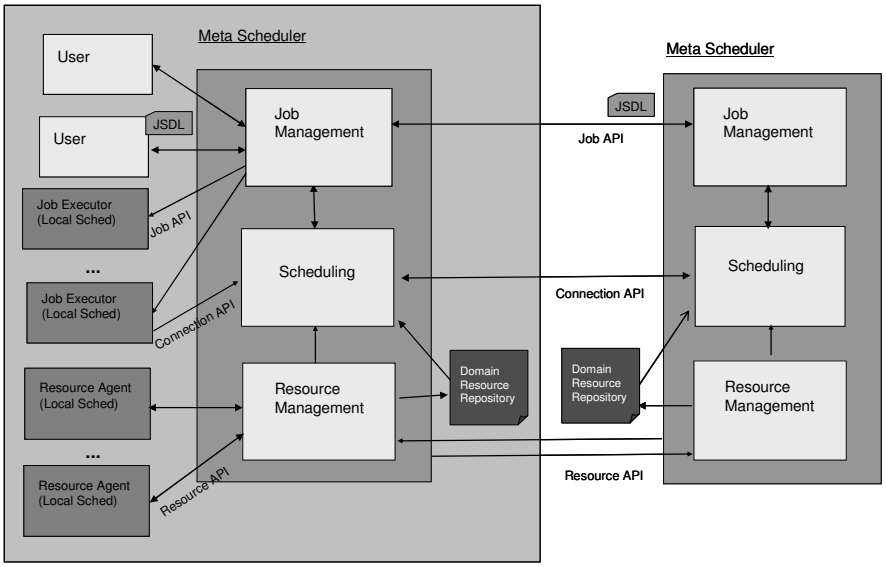


Figure 4. LA Grid domain meta-scheduling

We use the bioinformatics and hurricane mitigation scenarios depicted in Sections five and six as application test cases, as they both exhibit data and compute intensive workloads.

3. Job Flows

For many years, workflow[1] technology has been used in orchestrating multiple tasks in business processes. Only recently, scientific communities became very active in exploring workflow technology for orchestrating the execution of composite jobs that consist of multiple steps. The use of job flow would potentially provide richer expressiveness and flexibility for users to instruct the job management system on how to schedule and execute their jobs. In this part of our research, we explore issues related to job flow in grid environments, including job flow modeling, transparent workflow adaptation, and data dependencies in job flows.

3.1. Background

Job flow management can be achieved through service orchestration or choreography [23]. In service orchestration, job flow management is achieved through a central application. This application (usually an executable workflow) models the interaction between the partner services, so that they collectively accomplish a coarse grain task. The application is aware of the interfaces of the partner services and controls their execution order and message exchanges. In service choreography, job flow management is achieved through a distributed approach, where partner services are

[1] In this document, we use the terms *job flow* and *workflow* interchangeably.

aware of each other and each service knows of its participation in the message exchanges of the interaction. Figure 5 illustrates the difference between orchestration and choreography.

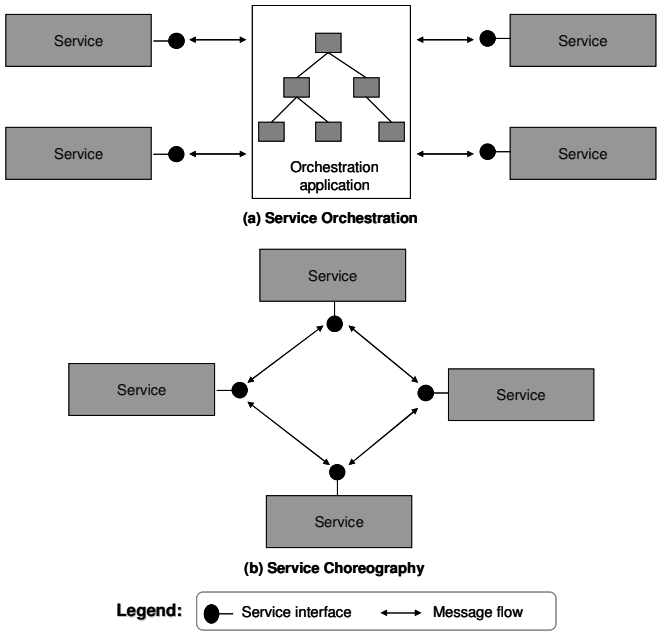


Figure 5. Orchestration and choreography

3.2. Overview

We have adopted a two-level approach to job flow management in grid environments that employs *service orchestration* at the upper level to control coarse-grain job submissions and *service choreography* at the lower level to control the interactions among executing jobs. To create high-level service orchestration for job submissions, we use the Business Process Execution Language (BPEL) [24], which has become the leading standard for web service orchestration. Web services can be integrated, using some XML-grammar, to create a higher-level application (business process). The XML-grammar that defines a BPEL process is interpreted and executed by an orchestration engine which exposes the process itself as a web service. BPEL provides many constructs for the management of process activities, including loops, conditional branching, fault handling and event handling (such as timeout). Additionally, it allows for activities to execute sequentially or in parallel. For the lower level choreography, we use JSDL [20] to describe the requirements of jobs for submission to the grid.

In addition jobs also need one or more input data items for their execution and can produce multiple outputs as well. There may be multiple copies/replicas of these data items in the system. For replica management in grids we use the Replica Location Service (RLS) available in the Globus Toolkit. RLS maintains and provides access to mapping information from logical names of data items to target names. These target names may represent the physical locations of the data items or map to other entries in

the RLS providing a second level of logical naming for the data items. The use of logical to physical name abstraction provides users with the option of specifying logical names of data items in their job descriptions, rather than the actual physical locations. Also, in distributed grids, it is very often desirable to maintain multiple copies of data items so that job executions can be optimized at more than one possible location.

Figure 6 provides an architectural diagram of our job flow management framework and shows how it interacts with meta-schedulers. First, the Grid Job Flow Application Modeling and Tooling captures the job control flow in an abstract BPEL workflow and the data dependencies as embedded JSDL scripts within the BPEL workflow. These documents are then interpreted to extract the job to job, job to data, and job to resource dependencies to form a directed graph. Next, this graph becomes more concrete by mapping the jobs to scheduling domains considering the resources they need, the data they require, and the other jobs they are dependent on. Then, additional steps for data transfer and Replica Location Service (RLS) registration are appended to the BPEL workflow. At this point the abstract BPEL workflow is concretized by binding the jobs to specific resource domains.

Any good job flow management system must adequately address the issue of fault-tolerance on behalf of the job flow. BPEL has constructs for detecting (and generating) fault messages, as well as constructs for specifying event-driven compensation activities. Compensation activities serve to undo some business logic that occurred prior to the event. However, grid environments call for more robust fault handling than is available in BPEL. To make job flows resilient to failure, in this step, we use a previously developed framework, called TRAP/BPEL [25], which adds autonomic behavior into existing BPEL processes automatically and transparently.

Unlike other approaches, TRAP/BPEL does not require any manual modifications to the original code of the BPEL process and there is no need to extend the BPEL language, nor the BPEL engine. Within the TRAP/BPEL framework, a BPEL workflow is made adaptive by first running it through an adaptation generator. The generator generates the adapt-ready BPEL process by inserting “hooks” at specified points in the workflow. These hooks redirect invocations through a proxy that provides adaptive behavior by shielding the workflow from failure and applying recovery mechanisms that are specified in a recovery policy [25].

During adaptation, specific jobs which require monitoring are identified and for each job, an adequate failure handling technique is specified in a recovery policy. The recovery policy is modeled in an XML document that is not part of the job flow definition. Failure handling techniques may include check-pointing, or finding an alternative (substitute) resource or service upon which to submit a job. Invocations for monitored jobs are replaced with invocations to a generic proxy. Messages for those jobs are therefore redirected through this proxy. The proxy using some failure detection mechanism (e.g. polling, event notification) monitors the individual jobs and enforces the recovery policy. The proxy in this case is a job submission and monitoring service and forwards jobs to the schedulers. There is only one generic proxy per job flow engine, although there may be several instances of this proxy performing recovery on behalf of the workflows executing on that engine.

Finally, the adapted concrete BPEL process with embedded JSDL scripts is executed on a BPEL engine to orchestrate the job submission through the use of some job submission web services. As described in Section two, meta-schedulers can engage in peer-to-peer choreography in order to decide on the specifics of how the actual job

execution and data transfers occur (e.g. the exact machine on which the job is to be executed, the exact data transfer protocol that is to be used).

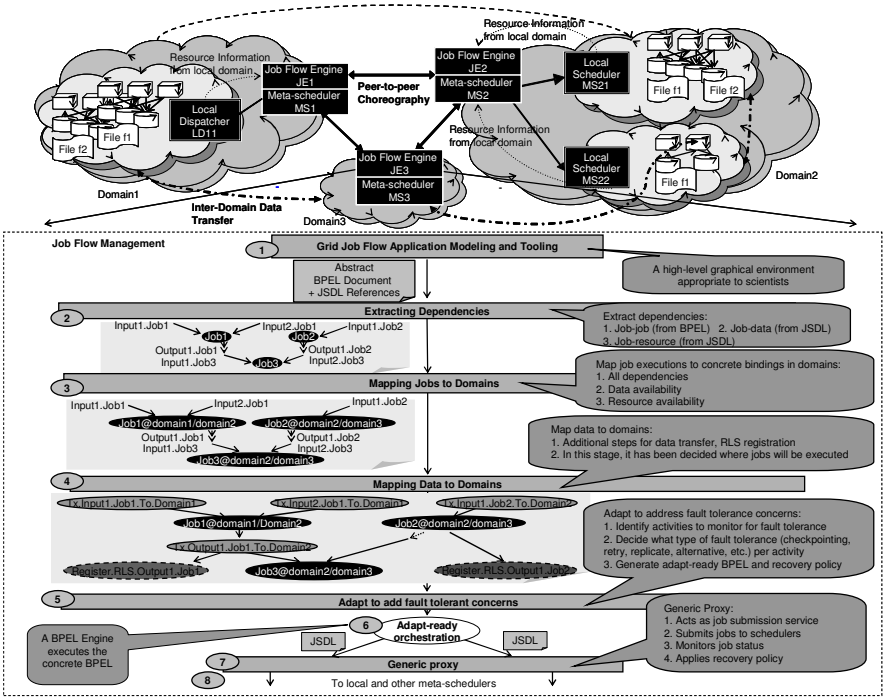


Figure 6. Job flow management framework interaction with meta-scheduling

3.3. Related Work

There are many job flow languages used in the scientific communities including DAGMan of the University of Wisconsin [26], SCUFL of the e-Science group [14], YAWL of Queenland University Technology [27]. In the enterprise domain, the most popular language for business processes is WS-BPEL [24]. WS-BPEL has become the de-facto standard for workflow technology in enterprise systems. To help foster interactions between job flow systems in the grid there is clearly a similar need for standardization. For this reason, in recent times, the scientific community has also started exploring the possibility of using the WS-BPEL standard for job flows. We selected WS-BPEL as our workflow language for our project. Software offers that support WS-BPEL include IBM's WebSphere Process Server [28] and ActiveBPEL [29].

There are many projects and studies of job flows in grid environments. The Pegasus [30] project at the University of Southern California addresses the planning and resource allocation of job flow in grid environments. VDS of GriPhyN [31] supports the virtual data specification language used in conjunction with the DAGMan job flow language. Unlike other studies, our project explores the combined use of WS-BPEL and JSDL along with data mapping.

Our architecture also supports the specification of flexible, user-defined failure handling mechanisms while supporting separation of concerns. By adapting the concrete BPEL workflow, failure handling mechanisms are defined in a manner that does not tangle the code for fault-tolerance with the business logic of the application. Some other Grid workflow systems (Karajan [32], Kepler [33] and Grid-WFS [34]) allow for user-defined failure handling, however, Karajan and Kepler do not support the separation of concerns. Grid-WFS claims to support the separation of concerns but fault handling strategies are specified along with the task in the same workflow definition. This approach does therefore entangle the code for failure handling with that of the business logic. Further, our approach does not require a purpose-built workflow engine, as do the other systems, since we utilize standard BPEL constructs.

For more general information on workflows, the reader is invited to refer to http://www.gridbus.org/reports/GridWorkflowTaxonomy.pdf, which is a good survey paper on workflow taxonomy and workflow projects.

4. Data Integration

Data integration involves combining data residing at different sources and providing the user with a unified view of such data. Data integration is a traditional problem that exists in numerous applications, for example, integrating the databases for two companies that are being merged, or combining research results from different bioinformatics repositories. Data integration frequently occurs as the volume of data and the need to share it increase. Data integration has been the focus of extensive work and numerous open research problems remain to be solved. In this study, we present a system architecture which aims to define essential components for developing a domain-specific, modular, and decentralized data integration system in a grid environment. The proposed architecture incorporates a series of grid-oriented services (e.g. coordinator, semantic catalog, repository, synchronization) that address the distributed nature and autonomy of the data sources. The architecture also incorporates a series of data-oriented services (e.g. schema mapping generation, query generation, query rewriting) that facilitate the actual integration of data. Nodes in the grid environment can provide data, integration services, or a combination of both. Through the proposed system, users can contribute new data, define relationships among existing data sources and schemas, relate data to domain-specific concepts, or even construct new schemas that can be reused by others.

4.1. Architecture and Services

Our architecture supports distributed storage and manipulation of data and adapts to dynamic addition and removal of nodes. For every session, an application server connects users to nodes and designates a particular node as the “master”. Any node is capable of performing the “master” role. The master node distributes the required tasks among many other nodes in the grid and is also responsible for coordinating, collecting, and merging results from “slave” nodes.

Figure 7 depicts the LA Grid data integration architecture. Each LA Grid node contains seven components/services, namely: the *coordinator service*, the *semantic catalog*, the *repository service*, a *data repository*, the *synchronization service*, the *Data Grid Management System (DGMS)* [35], and the *data services suite*. At this time the

data services suite is completely implemented and other services are at different stages of development. Implementation is done in Java and nodes are currently deployed in a grid architecture based on the Globus Toolkit [5].

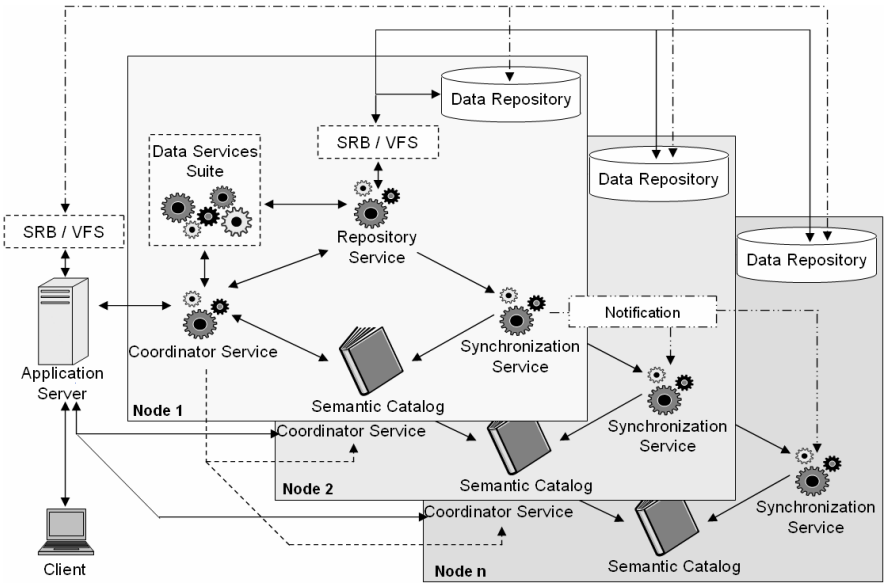


Figure 7. Data integration architecture

Coordinator Service: This service is the front-end interaction point for data-driven operations. It has access to a locally held semantic data catalog. The local coordinator service communicates with remote coordinator services for coordinating operations across the system. Query distribution/data materialization decisions are also handled by this service based on quality of service, load balancing, and optimization requirements. Coordinator services are key elements for automated workflow construction because they are responsible for decisions involving forwarding, splitting, or directly handling a request. Upon receiving a request, the coordinator needs to access the semantic catalog to retrieve information about data needed/involved in the request. Based on metadata and information about registered nodes and deployed services, the coordinator makes a decision about handling the request, i.e., forwarding, splitting, or directly handling it. The coordinator maintains a set of rules to help make such decisions. The efficiency and accuracy in defining such rules is crucial for correct and efficient system performance. While a system could perform its function using a limited set of rules, its performance could be enhanced by adding and tuning rules. For example, a simple rule is to forward a request to the first node that has the required services deployed. However, a better rule is to incorporate the location of data. From our experience, fine tuning rules may result in complex but efficient workflows of services.

Semantic Catalog: The semantic catalog contains the physical locations of data components as well as domain-specific semantic descriptions. For example, the semantic catalog of each schema may store its name, the location of its definition,

semantic mappings from the schema elements to biomedical domain-specific concepts, and pointers to known instances of the schema. Additionally, we may store schema mappings that directly relate pairs of schemas (created by the schema mapping creation service – cf. detailed description below). The semantic mappings between schema elements and domain-specific concepts are currently simple correspondences from schema elements to terms in a conceptual model, similar to an ontology. Users provide ontologies and mappings applicable to their problem domain. This kind of semantic mapping is an active research area [36] that can enhance schema mapping/matching operations (for example, if two or more schema elements can be mapped to the same semantic term in the ontology, a potential match is indicated). In the current implementation, UMLSKS [37] is used as the domain-specific mappings knowledge source. The latest version of the semantic catalog is built as a dynamically evolving OWL (Web Ontology Language) [38] resource. It captures relations between schema definitions, instances, mappings and their domain. Those relations are represented as RDF (Resource Description Framework) [39] statements that can be manually asserted by system users or can be automatically evaluated using a set of utility services. Using this design, the semantic catalog is able to answer queries like: Is there a mapping from schema A to schema B?; Find all mappings that use schema C as a source; Does a mapping have an inverse in the repository?; etc. Pellet [40] - an open source, OWL Description Logic reasoner in Java - is used for reasoning and augmenting information in the catalog while SPARQL [41] is used as the query language.

Repository Service & Data Repository: This service is responsible for storing and extracting all raw data (via the DGMS or VFS - discussed below). It also notifies the synchronization service about new changes in the repository. Currently, this service is implemented on top of the Apache Commons Virtual File System [42]. VFS provides APIs for accessing different file systems and presents a unified view of files from different sources (e.g., local disk, remote ftp, or http servers). In the current system implementation, only a pure XML data repository is supported. Other data representations can be supported only if the data can be exported to XML.

Synchronization Service: The synchronization service keeps the semantic catalog entries synchronized among nodes. When a node is added to the grid, the node has the option of subscribing to various topics. Whenever a change affecting a given topic occurs, the nodes subscribed to that topic receive notifications of the update. This service is implemented on top of the WSRF notification mechanism provided by the Globus Toolkit.

Data Grid Management System: Through multiple abstractions, the DGMS [43] provides a logical namespace that hides the complexity of distributed data and heterogeneous resources. The Storage Resource Broker (SRB) [44] is a tool for managing distributed storage resources. Files in the SRB are referenced by logical file handles that do not require the actual physical locations of the files. A Metadata Catalog, MCAT, maintains maps of logical handles to physical file locations. The proposed semantic catalog and data services can be seen as augmentations on top of DGMS in order to facilitate finer grain semantic integration at the data level. The current system implementation uses the Apache VFS.

Data Services Suite: This component provides a number of web services that allow the creation of schema mappings and operations over those schema mappings. We have selected Clio's [45] schema mapping components, and wrapped them as web services. The suite provides the following data services:

- ***Schema Mapping Creation:*** Given a source schema, a target schema, and a set of "correspondences" between source and target schema elements, this service creates a "mapping" from the source schema to the target schema. This mapping consists of a set of declarative constraints that dictate what the target instance should be, given a source instance. The mapping creation algorithm takes into account the schema constraints (e.g., foreign key constraints, type constraints) as well as correspondences [46].
- ***Query Generation:*** Given a mapping (produced by the Schema Mapping Creation service), this service produces an XQuery, an XSLT, or an SQL/XML query that implements the transformation implied by the mapping [45]. The query and the association between the query and the mapping used to produce the query are stored in the semantic catalog (for future reuse).
- ***Query Execution:*** For convenience, we also have a service that executes the queries generated by the previous service. Given a query script and a set of input XML documents (e.g. instances of the source XML schema), the service executes the query and returns the resulting XML document.
- ***XML Transformation:*** This service allows the direct and scalable execution of the mapping, as opposed to simply executing the query that implements it. Based on the technology detailed in [47], this service takes as input a mapping and the source XML instances and returns the target XML instance that is implied by the mapping. As opposed to the query generation/execution services, this service neither produces nor executes a query; rather, it uses a Java-based engine to optimally execute the mapping.
- ***Query Rewrite:*** An interesting application of mappings is the ability to rewrite target-side queries into queries that work on the source-side. This is useful, for example, if the target-side schemas are virtual and actual data resides on the source side. We use the query rewriting techniques detailed in [48] to implement this service. Given a schema mapping and an XQuery over a target schema instance, this service returns a rewritten XQuery over the source schemas in the mapping.
- ***Schema Integration:*** Given a number of mappings between several schemas, this service attempts to create an "integrated" schema that captures the unified concepts of the schemas that are related by the mapping [49].

4.2. Related Work

A number of data integration systems have been proposed to address the problem of large-scale data sharing (e.g. [50, 51, 52]; and the survey by Halevy [53]). These systems support rich queries over large numbers of autonomous, heterogeneous data sources by making use of semantic relationships between the different source schemas and a mediated schema, which is designed globally. However, the mediated schema becomes a problem since it may be hard to come up with a single mediated schema that everyone agrees on. Moreover, all access (querying) is done against the mediated schema (a single point). Furthermore, this architecture is not robust with respect to the changes in the source schemas. As a result, data integration systems based on mediated schemas are limited in supporting large-scale distributed and autonomous data sharing. Peer Data Management Systems (PDMS), e.g., Piazza [54], have been proposed to address the aforementioned problems and to offer an extensible and decentralized data

sharing system. The study presented in this section can be viewed as an effort to present and discuss the design, components, and services required to realize a PDMS in a grid environment. Our system requirements are, in principle, no different from these peer data management systems. Compared to Piazza's approach, our intended applications imply smaller numbers of data sources. However, the sources have complex schemas and may contain overlapping and potentially conflicting and dynamically changing data. The proposed system emphasizes the use of tools and services that facilitate mappings among schemas and generate the queries that are needed to access and integrate the data.

5. Application to Bioinformatics

A problem facing many bioinformatics researchers today is the aggregation and analysis of vast amounts of data produced by large scale projects such as the Human Genome Project. This is further complicated by the fact that data is distributed among heterogeneous sources. As of September 2006, the Gene Expression Omnibus (GEO) repository at the National Center for Biotechnology Information (NCBI) holds over 3.2 billion individual measurements. Moreover the repository is growing at a rapid rate [55]. The amount of data, the rate of its growth, and the heterogeneity of data sources present real problems, hindering advancements in bioinformatics [56].

In this section, the focus is on data integration problems in bioinformatics. However, a typical bioinformatics research activity involves both computational and data driven aspects. Data driven tasks involve techniques to extract data from multiple sources. Computational tasks involve processing data after extraction for pattern matching, alignment, and clustering.

Pharmacogenomics is a branch of bioinformatics dealing with the influence of genetic variation on drug response in patients. Approaches investigating such influences promise the advent of "personalized medicine", in which drugs and drug combinations are optimized for each individual's unique genetic makeup. To make "personalized medicine" decisions, information from multiple heterogeneous data sources needs to be incorporated; for example OMIM, dbSNP and dbGaP from NCBI [57], Haplotype data from the HapMap project [58], Human Gene Mutation and TRANSFAC databases from BioBase [59] in addition to PHARMKGB [60]. Figure 8 shows a related example that aims to understand rates of gene expressions in different tissues and correlate these expression profiles with active transcription factors and their binding sites. The data required for this study is distributed among various sources (e.g. UCSC Genome Browser [61], GNF SymAtlas [62] and TRANSFAC [63]). The figure shows how Clio mapping technology can be used to provide a high-level definition for mappings between such sources and a target schema. In particular, a graphical user interface allows the identification of correspondences that relate schema elements.

Our sample scenario involves three collaborating groups of scientists. Assume the groups are associated with the three data sources in Figure 8 and located in the USA, Spain, and Mexico, respectively. The USA group is conducting experiments related to identifying known genes and their chromosomal positions. The team from Spain is doing experiments on gene expression levels in different tissues, while the team from Mexico is concerned with identifying transcription factors binding sites for different genes and the associated transcription factors. Furthermore, assume there is a fourth team in the UK that will do the analysis of the collected data. Their role is to collect

and interpret data from the different teams and to discover new knowledge from the experiments conducted in the study.

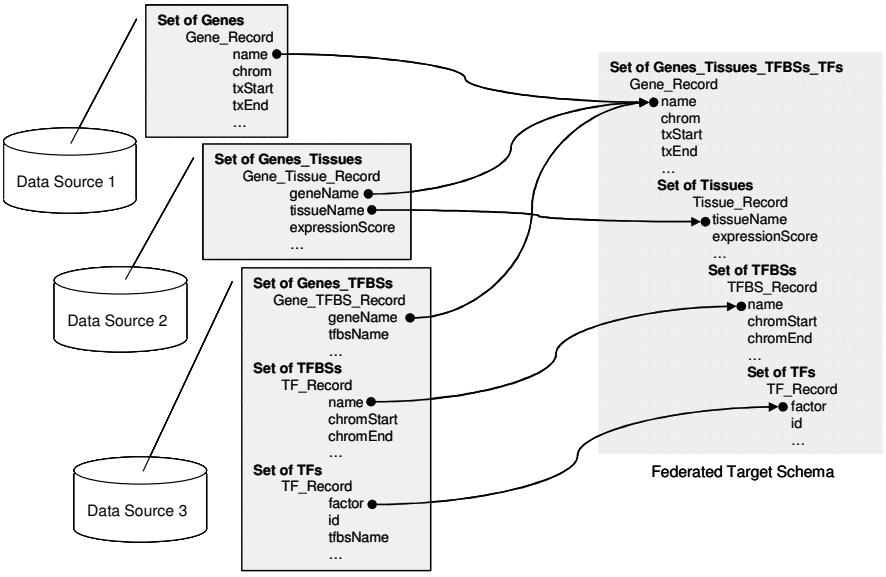


Figure 8. A high-level mapping definition using Clio

Assume that the teams in the USA, Spain and Mexico have uploaded their data to their associated data sources. Now the UK analysis team can start interpretation and analysis of the data. However, they are facing the problem of merging and integrating these three data sources. To efficiently analyze the data, they would like to organize it according to a specific structure. Therefore, the UK team constructs a new schema that captures the required data organization (the target schema in Figure 8) and creates the required domain-specific mappings. Then, they connect to their node and download (via Java Web Start) an application that allows the construction of Clio-based mappings. Source and target schemas are loaded into the tool which shows their structure as a tree of schema elements (very similar to how they are presented in Figure 8). Value mappings are entered by drawing lines from source schema elements to target schema elements. The output from this process is a value mapping, which is sent to the schema mapping service that evaluates the mapping specification and passes it to the repository service for storage. The synchronization service updates the local semantic catalog and notifies remote nodes about the new mapping.

Different query processing scenarios could arise based on the location of data (local vs. remote), and whether the query is against a materialized version of the data or not. If the data is not materialized then either a materialization or a query distribution decision could be made by the coordinator service. Criteria for such decisions can be based on the frequency of the queries against data sources.

For instance, the UK team is trying to answer the following query using the federated target schema:

"Find a list of ***gene names*** *and* ***their chromosomal locations*** *that have an* ***expression level > e*** *in* ***both heart and liver*** *and are* ***regulated by the same set of transcription factors.***"

The above query is written against the target schema. However, it is assumed that the UK node does not have any data associated with this federated schema; all data resides at the other nodes (a *global-local-as-view* –GLAV– scenario [64]). The query is rewritten by the Query Rewrite service into a new query, formulated in terms of the source schemas (at the USA, Spain, and Mexico sites). When executed, the rewritten query retrieves the three source documents and then locally (on the UK node) joins and filters the data, and finally produces an instance of the target schema. Another alternative we are exploring is to further decompose the rewritten query into maximal sub-queries that are sent to the sources. For example we could send a join query to data source 3 (Mexico) and only get the relevant data back. In another scenario, one or more source schemas may also be the output of previous schema mapping operations. In such cases, a nested query rewrite with further decomposition is needed.

6. Application to Hurricane Mitigation

6.1. A Possible Hurricane Mitigation Scenario

A tropical depression in the Caribbean Sea quickly strengthens in the warm waters as it drifts westward into the Gulf of Mexico. It is tracked and modeled by the National Hurricane Center as it becomes a tropical storm and then a category one hurricane by day three. Once in the central Gulf of Mexico the storm intensifies into a category three hurricane by the end of day four. A similar storm is shown in Figure 9.

Hurricane track models begin to indicate that the storm will continue to intensify and re-curve northeast and then east toward the western Florida coast. The NOAA National Hurricane Center model forecasts begin to predict landfall along the central western Florida coast by day six.

Synoptic scale numerical weather prediction (NWP) models capture the general storm circulation and general movement, but do not predict intensity changes well [65, 66]. Results of these large scale models, in turn, are used along with other high resolution data as input to regional and mesoscale models that run ensembles across a computing grid infrastructure of thousands of nodes. These ensemble models do more than determining high resolution hurricane impact; they also provide information about the uncertainty of the hurricanes track and intensity forecast. Between 48 and 72 hours prior to landfall these ensembles allow risk management for the event, as they also include information about the sensitivity of the forecast to both data and physics uncertainty.

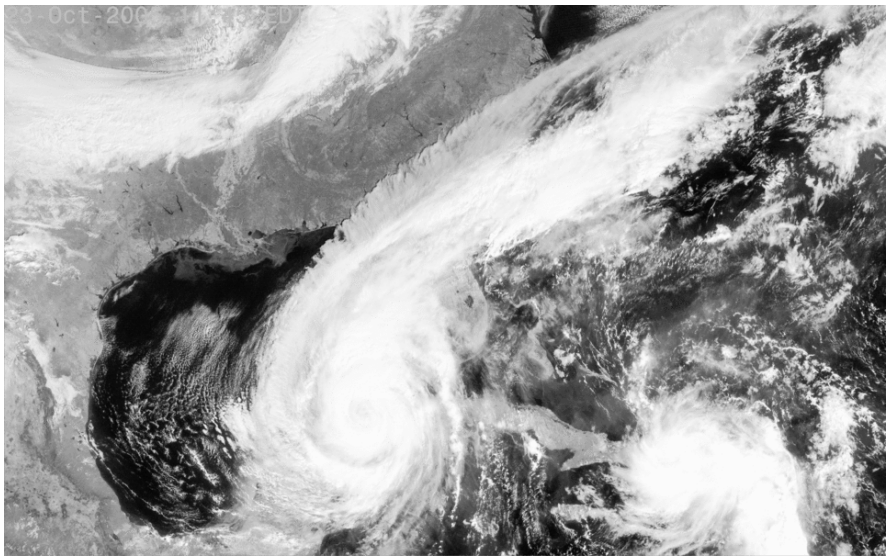


Figure 9. Hurricane Wilma in the Gulf of Mexico - October 23, 2005

6.2. Our approach

The Weather Research and Forecasting (WRF) model is the state-of-the-art mesoscale numerical weather prediction system, serving both operational forecasting and atmospheric research needs [67]. The WRF model Version 2.1.2 software distribution comprises about 360000 lines of source code. This code is highly modular and is greatly optimized to run on several heterogeneous cluster computing facilities using MPI [1] for inter-node communications and OpenMP [68] for intra-node communications among the processors. In this part of our research, we address two problems with the current version of WRF.

First, to mitigate the impact of hurricane landfalls, we need to provide even more accurate and timely information to enable effective planning [69, 70, 71, 72]. Pushing the limits of WRF today, there is a increasing need for fine-grain simulation in smaller regions (e.g. zip-code level hurricane simulation). Considering the limited resources available—that each interested organization may have—would leave us with no choice than to scale out from single-managed cluster computing to grid computing that can span over many organizations with different needs and administrative domains. Such grid-enabled WRF code can employ the resources available in several organizations to contribute toward solving fine-grain hurricane simulations. As the WRF code does not provide any inherent support for grid computing and as we do not want to stray away from the mainstream revisions of the WRF code, we are employing our Transparent Grid Enablement approach (mentioned in Section one) to enable execution of WRF on grid computing environments in a transparent manner to the original WRF code (no manual modification to the WRF code).

Second, to contribute to future WRF model development, meteorologists are required to develop new dynamics and physics model packages in languages such as FORTRAN and C as well as to understand the architecture of the underlying

computing platforms to optimize code – very challenging skills. In this part of our research, we address this problem by investigating on a high-level modeling platform, a web-based interface called the Grid WRF portal. The portal allows meteorologists to use WRF and to develop new meteorology model packages adaptable to different grid computing environments. Our preliminary work shows that a specialized visual modeling interface supported by a workflow language (e.g. BPEL) may be a good choice. We continue to address challenging questions such as the expressive power of workflow languages [73, 74, 75, 76], and the efficiency of the generated WRF code [77]. This high-level graphical user interface is backed up with the results of our other research mentioned in Sections two and three, where we have designed and partially developed a job flow management system and a meta-scheduler that can adapt to the dynamic changes of a grid computing environment.

Beyond grid enablement of hurricane modeling across multiple geographic and temporal scales for implementation of real time ensemble simulations, there is the potential for creating a suite of products both distinct and integrated which couple multiple types and scales of model simulations (atmospheric, ocean wave, storm surge, hydrological, socio-economic models, etc.). This capability coupled with user- and application-centric visualizations gives rise to a new class of data analytics for decision support in a proactive sense that has heretofore been unavailable.

Ocean wave and storm surge models are also run on the grid and are used to better predict impacts on coastal infrastructure, shipping and oil drilling interests in the eastern Gulf of Mexico and along the shoreline as the hurricane continues its eastward track. Mean and ensemble members of the atmospheric, ocean wave, storm surge and hydrological models are coupled to create predictions on storm movement, intensity, near shore wave heights, storm surge, and flooding forecasts that include both optimistic and pessimistic bounds on the likely outcomes. The various model forecasts are concurrently produced and visualized with grid resources to create customized decision support guidance for emergency management, utility, transportation, debris removal and other interests as the storm makes landfall along the western Florida coast.

National authorities do not currently have the required computational power nor the bandwidth to produce such high resolution (cloudscale) ensembles and deliver them to emergency management, businesses, and the public. Within the grid environment, smaller domain storm surge results and visualization of complex rainfall, wind fields and waves can be generated for distinct localities in high detail. Mesoscale and cloudscale model ensemble forecasts continue to run in real time utilizing grid resources to enable timely generation of decision support products. Information, data and analytics are produced in multiple forms, which are then integrated and delivered to support planning, response and remediation as the storm moves eastward across Florida and exits along the northeastern coast. These analytics allow improved estimates of damage to electrical grids and transportation infrastructure while supporting efforts to plan and mobilize storm recovery.

Grid enablement for the generation of new and unique products, guidance and analytics in support of mission critical decision making in a transparent manner while utilizing a dynamic, heterogeneous and geographically disparate set of computing resources is one of the strategic goals of the LA Grid hurricane mitigation project. It is a necessary and evolutionary step in the convergence of science and technology for societal benefit.

7. Future Plans

Our initial work demonstrated in previous sections is part of a novel approach to grid application development we call Transparent Grid Environment (TGE), whose goal is to allow domain experts to effectively express the logic and software artifacts of domain applications while hiding the details of the grid architecture, software, and hardware stack. This TGE paradigm will serve as the foundation for the study of application development methodologies, platforms, and tools that will significantly ease grid-enabled application development (hence broadening grid utilization) and make applications more portable and adaptable to future changes of grid technologies. We believe that grids utilizing the TGE paradigm will be agile, flexible, and capable of serving a broad set of scientific and business communities.

Our approach is characterized as application-driven (hence "top-down") by basing and focusing our investigation on (1) supporting grid-enablement for a few carefully chosen critical application domains, e.g. hurricane mitigation and bioinformatics, and (2) developing common methodologies, services and tools for deploying grid-enabled applications in these domains. In our approach, we factor out common services that can be reused across domains. This will ensure that our tools have broad significance and utility to a range of applications, thus avoiding the tendency for tools to be too generic to be effective. Our future plans entail investigation of the following key challenges:

1. *High-level Visual Interactive Development Environment (IDE)*: What is the appropriate IDE for domain experts to easily specify the logic of their applications? Do IDEs targeted for different domains have many common properties? Is it possible to develop a common IDE that supports multiple domains? Are the workbenches being developed and used by domain experts today the right solution?
2. *Automated Code Generation and Software/Hardware Reuse*: How can we automate the generation of executable code from a high-level specification provided by a domain expert? How can we reuse the existing software and hardware components and map abstract specifications to concrete resources in order to execute the application?
3. *Hiding the Heterogeneity of Grid Architectures*: How do we hide the details of heterogeneous grid architectures and resources and provide a virtualized interface for application development while addressing efficient resource utilization?

Under the LA Grid initiative, we have established a globally-integrated research and education program to respond to the above challenges and to realize a TGE. We have taken a divide-and-conquer approach that allows us to tackle the above challenges in parallel. To conduct our investigations we have identified nationally-important domains of *Grid Applications,* as discussed in Sections five and six. We have established a number of *Grid Integration* projects that enable aggregation and discovery of data, visualization of data, and weaving of high-level grid enablement services into the logic of domain-specific applications, as described in Section four. Finally, we have established several *Grid Enablement* projects that provide a layer of abstraction on top of heterogeneous Grid architectures by offering high-level services, as presented in Sections one, two, and three.

As our research progresses, the technologies and their associated tools developed in these projects will form grid application enabling platforms at different levels of abstraction. First, the projects in the Grid Applications Layer will identify the

requirements and provide methodologies, frameworks, and modeling tools that enable domain experts to model, design, and code their applications with minimal attachment to the underlying grid. In designing and developing these "application interfaces," common high-level, application-oriented functions, components, and tools are extracted and packaged into the Grid Integration Layer, which provides services that can be reused by other applications in the targeted domains. The projects in the Grid Enablement Layer will provide system-level services and tools to support efficient and transparent management and utilization of heterogeneous grid resources through uniform virtualized interfaces. Such services will make applications even more adaptable to changes of the underlying infrastructure.

The LA Grid initiative aims to simplify the manner by which scientific and business domain experts develop, use, and maintain software applications over distributed computing resources. Using our TGE approach we will create innovative tools which promote the reuse of commonalities across domains resulting in flexible and cost effective grid implementations which allow experts in diverse domains to easily code their application logic using an integrated development process.

References

[1] http://www-unix.mcs.anl.gov/mpi/.
[2] http://www.globus.org/ogsa/.
[3] http://www.globus.org/wsrf/.
[4] http://www-unix.mcs.anl.gov/mpi/mpich2/.
[5] http://www.globus.org/toolkit/.
[6] http://www.unicore.org/.
[7] http://www.cs.wisc.edu/condor/.
[8] R.M. Badia, R. Sirvent, J. Labarta, J.M. Perez, Programming the GRID: An Imperative Language Based Approach, Book chapter in *Engineering the Grid*, Section 4, Chapter 12 , January 2006.
[9] S.M. Sadjadi, P.K. McKinley, B.H.C. Cheng, and R.E.K. Stirewalt, TRAP/J: Transparent generation of adaptable Java programs, *Proc. International Symposium on Distributed Objects and Applications (DOA'04)*, Agia Napa, Cyprus, October 2004.
[10] R.V. van Nieuwpoort, J. Maassen, T. Kielmann, and H.E. Bal, Satin: Simple and efficient Java-based grid programming, *Scalable Computing: Practice and Experience* **6(3)**, 19-32, September 2005.
[11] S. Gorlatch and J. Dunnweber, From Grid Middleware to Grid Applications: Bridging the Gap with HOCs, in *Future Generation Grids*, Springer Verlag, 2005.
[12] M. Aldinucci, M. Coppola, M. Danelutto, M. Vanneschi, and C. Zoccolo, Assist as a research framework for high-performance grid programming environments, in *Grid Computing: Software Environments and Tools*, J.C. Cunha and O.F. Rana, Eds., Springer Verlag, 2004.
[13] F. Baude, L. Baduel, D. Caromel, A. Contes, F. Huet, M. Morel, and R. Quilici, Programming, Composing, Deploying for the Grid, in *Grid Computing: Software Environments and Tools*, J.C. Cunha and O.F. Rana, Eds., Springer Verlag, January 2006.
[14] http://taverna.sourceforge.net/.

[15] V. Subramani, R. Kettimuthu, S. Srinivasan, and P. Sadayappan, Distributed Job Scheduling on Computational Grids Using Multiple Simultaneous Requests, *Proc. 11th IEEE International Symposium on High Performance Distributed Computing*, Edinburg, Scotland, July 24-26, 2002.
[16] I. Rodero, et al., Looking for an Evolution of Grid Scheduling: Meta-brokering, submitted to ICS 2007.
[17] http://www.globus.org/grid_software/computation/csf.php.
[18] http://www-306.ibm.com/software/tivoli/products/scheduler-loadleveler/.
[19] http://www.gridway.org/.
[20] http://www.ogf.org/.
[21] A. Anjomshoaa, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, Job Submission Description Language (JSDL) Specification, Version 1.0, November 2005, Copyright © Global Grid Forum (2003-2005).
[22] http://www-306.ibm.com/software/tivoli/products/dynamic-workload-broker/index.html.
[23] C. Peltz, Web services orchestration and choreography, *IEEE Computer*, **36(10)**, 44–52, 2003.
[24] S. Weerawarana, F. Curbera, Business process with BPEL4WS: Understanding, *http://www-128.ibm.com/developerworks/library/ws-bpelcol1/*, 2002.
[25] O. Ezenwoye and S.M. Sadjadi. TRAP/BPEL: A framework for dynamic adaptation of composite services, *Proc. International Conference on Web Information Systems and Technologies (WEBIST 2007)*, Barcelona, Spain, March 2007.
[26] http://www.cs.wisc.edu/condor/dagman.
[27] http://yawlfoundation.org/index.php.
[28] http://www-306.ibm.com/software/integration/wps/.
[29] http://www.activebpel.org/.
[30] E. Deelman, G. Singh, M.H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, G. K. Vahi, G.B. Berriman, and J. Good, Pegasus: A framework for mapping complex scientific workflows onto distributed systems, *Scientific Programming* **13** (2005), 219–237.
[31] http://vdt.cs.wisc.edu/components/vds.html.
[32] G. von Laszewski and M. Hategan, Java CoG Kit Karajan/GridAnt Workflow Guide, *Technical Report, Argonne National Laboratory*, Argonne, IL, USA, 2005.
[33] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, J. Tao, and Y. Zhao, Scientific Workflow Management and the KEPLER System, *Concurrency and Computation: Practice & Experience*, Special Issue on Scientific Workflows, 2005.
[34] S. Hwang and C. Kesselman, GridWorkflow: A Flexible Failure Handling Framework for the Grid, *Proc. 12th IEEE International Symposium on High Performance Distributed Computing*, Seattle, WA, June 2004.
[35] Data Grid Management System, http://www.sdsc.edu/srb/index.php/DGMS.
[36] Y. An, A. Borgida, and J. Mylopoulos, Constructing Complex Semantic Mappings between XML Data and Ontologies, *Proc. Fourth International Semantic Web Conference* (2005), 6–20.
[37] http://umlsks.nlm.nih.gov/.
[38] http://www.w3.org/TR/owl-features/.
[39] http://www.w3.org/RDF/.
[40] http://pellet.owldl.com/.

[41] http://www.w3.org/TR/rdf-sparql-query/.
[42] http://jakarta.apache.org/commons/vfs/.
[43] A. Jagatheesan, and A. Rajasekar, An introduction to data grid management systems, *Proc. SIGMOD* (2003), 683.
[44] http://www.sdsc.edu/srb/index.php/.
[45] L.M. Haas, M.A. Hernandez, H. Ho, L. Popa, and M. Roth, Clio Grows Up: From Research Prototype to Industrial Tool, *Proc. SIGMOD* (2005), 805–810.
[46] L. Popa, Y. Velegrakis, R.J. Miller, M.A. Hernandez, and R. Fagin, Translating Web Data, *Proc. VLDB* (2002), 598–609.
[47] H. Jiang, H. Ho, L. Popa, and W.S. Han, Mapping-Driven XML Transformation, *Proc. 16th International World Wide Web Conference* (2007).
[48] C. Yu and L. Popa, Constraint-Based XML Query Rewriting for Data Integration, *Proc. SIGMOD* (2004), 371–382.
[49] L. Chiticariu, P.G. Kolaitis, and L. Popa, Semi-Automatic Generation and Exploration of Schema Integration Alternatives, Manuscript under preparation (2007).
[50] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J.D. Ullman, V. Vassalos, and J. Widom, The TSIMMIS Approach to Mediation: Data Models and Languages, *Journal of Intelligent Information Systems*, **8(2)** (1997), 117–132.
[51] O.M. Duschka and M.R. Genesereth, Answering recursive queries using views, *ODS* (1997), 109–116.
[52] I. Manolescu, D. Florescu, and D. Kossmann, Answering XML Queries on Heterogeneous Data Sources, *VLDB* (2001), 241–250.
[53] A.Y. Halevy, Answering Queries Using Views: A Survey, *VLDB* (2001) 270–294.
[54] A.Y. Halevy, Z.G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov, The Piazza Peer Data Management System, *IEEE Trans. Knowl. Data Eng.*, **16(7)** (2004), 787–798.
[55] T. Barrett, D. Troup, S. Wilhite, P. Ledoux, D. Rudnev, C. Evangelista, I. Kim, A. Soboleva, M. Tomashevsky, and R. Edgar, NCBI GEO: mining tens of millions of expression profiles database and tools update, *Nucleic Acids Research*, **35** (2006), D760–D765.
[56] L. Stein, Creating a Bioinformatics Nation, *Nature*, **417(6885)** (2002), 119–120.
[57] http://www.ncbi.nlm.nih.gov/.
[58] http://www.hapmap.org/.
[59] http://www.biobase-international.com/pages/.
[60] https://www.pharmgkb.org/.
[61] W. Kent, C. Sugnet, T. Furey, K. Roskin, T. Pringle, A. Zahler, and D. Haussler, The Human Genome Browser at UCSC, *Genome Research*, **12(6)** (2002), 996–1006.
[62] A. Su, et al., Large-scale analysis of the human and mouse transcriptomes, National Academy of Sciences of the United States of America (2002).
[63] V. Matys, et al., TRANSFAC: transcriptional regulation, from patterns to profiles, *Nucleic Acids Research*, **31** (2003), 374–378.
[64] M. Friedman, A. Levy, and T. Millstein, Navigational plans for data integration, Proc. 16th National Conference on Artificial Intelligence (AAAI), 1999.
[65] R.L. Elsberry, T.D.B. Lambert, and M. Boothe, Accuracy of atlantic and eastern north pacific tropical cyclone intensity forecast guidance. Submitted to *Weather and Forecasting* (2006).

[66] F.D. Marks, and L.K. Shay, Landfalling tropical cyclones: Forecast problems and associated research opportunities, *Bull. Amer. Met. Soc.,* **79** (1998), 305–323.
[67] http://www.wrf-model.org/index.php.
[68] http://www.openmp.org/drupal/.
[69] H.E. Willoughby. Improvements in observations, models and forecasts. *HURRICANE! Coping with Disaster*, R.H. Simpson, Ed., AGU (2002), 205-216.
[70] P. Singh, N. Zhao, S.-C. Chen, and K. Zhang, Tree Animation for a 3D Interactive Visualization System for Hurricane Impacts, *Proc. IEEE Intl. Conf. on Multimedia* (2005), 598-601.
[71] S.-C. Chen, S. Hamid, S. Gulati, N. Zhao, M. Chen, C. Zhang, and P. Gupta, A Reliable Web-based System for Hurricane Analysis and Simulation, *Proc. IEEE International Conference on Systems, Man and Cybernetics* (2004), 5215–5220.
[72] S.-C. Chen, M.-L. Shyu, C. Zhang, W.Z. Tang, and K. Zhang, Damage Pattern Mining in Hurricane Image Databases, *Proc. IEEE Intl. Conf. on Info. Reuse and Int.* (2003), 227–234.
[73] O. Ezenwoye and S.M. Sadjadi, Composing aggregate web services in BPEL, *Proc. 44th ACM Southeast Conference* (ACMSE), March 2006, Melbourne, FL.
[74] O. Ezenwoye and S.M. Sadjadi, TRAP/BPEL: A framework for dynamic adaptation of composite services, *Proc. International Conference on Web Information Systems and Technologies* (WEBIST), March 2007, Barcelona, Spain.
[75] O. Ezenwoye and S.M. Sadjadi. RobustBPEL2: Transparent autonomization in business processes through dynamic proxies, *Proc. 8th International Symposium on Autonomous Decentralized Systems* (ISADS), March 2007, Sedona, AZ.
[76] O. Ezenwoye and S.M. Sadjadi, Enabling robustness in existing BPEL processes., *Proc. 8th International Conference on Enterprise Information Systems*, May 2006, Paphos, Cyprus.
[77] S.M. Sadjadi, J. Martinez, T. Soldo, L. Atencio, R.M. Badia, and J. Ejarque, Improving separation of concerns in the development of scientific applications. *Technical Report FIU-SCIS-2007-02-01*, School of Computing and Information Sciences, Florida International University, Miami, FL, February 2007.