Abstract

The computation of Contaminant Source Characterization (CSC) is a critical research issue in Water Distribution System (WDS) management. We use a simulation framework to identify optimized locations of sensors that lead to fast detection of contamination sources. The optimization engine is based on a Genetic Algorithm (GA) that interprets trial solutions as individuals. During the optimization process many thousands of these solutions are generated. For a large WDS, the calculation of these solutions are non-trivial and time consuming. Hence, it is a compute intensive application that requires significant compute resources. Furthermore, we strive to generate solutions quickly in order to respond to the urgency of a response.

Grid computing can help in several ways. First, it provides infrastructure that is of sufficient computational power. Second, it allows the introduction of fault tolerant mechanisms, as ample resources could be made available. Third, due to the power of the available systems fast performance can be achieved.

To carry out the calculations we require user-level middleware that can be supporting the workflow of the application and manages the resource assignment in an efficient and fault tolerant fashion. To do so we have prototyped the cyberaide framework that provides a convenient command line and portal layer of steering applications on Grids. Internally, we utilize a sophisticated workflow engine that provides the ability to access elementary fault tolerant mechanisms for job scheduling. This includes the management of job replicas and the reaction on late return of results.

We report the test results of CSC problem solving on a real Grid test bed – the TeraGrid test bed. In addition, we contrast this system architecture with a Hadoop-based implementation that includes automatically fault tolerance. The later activity has been conducted on FutureGrid.

Keywords: Grid computing, Cyberaide workflow, Water Distribution System

1 Introduction

Urban Water Distribution Systems (WDSs) are vulnerable to accidental and intentional contamination incidents that could result in adverse human health and safety impacts [1]. When a contamination event is detected via sensor networks, municipal authorities are faced with obtaining information about the contamination. This includes critical information, such as (a) the location of the contaminant, (b) the time when the contamination started, (c) type of the contamination, (d) concentrations of the contamination and its distribution throughout the WDS. The real-time solution for identifying this information is critical to improve the safety of the overall system and its users. The process to identify this information is referred to as the Contaminant Source Characterization (CSC). CSC presents a significant computational challenges: it requires high-performance computing resources, a modern cyberinfrastructure middleware that includes reliable simulation and optimization software, and spatial-temporal data management methods to deal with the size of the input network that for a million person city is of considerably large.

Hence, we present a solution that utilizes Grid and Cloud concepta. The Grid portion of this work work was enabled through the development of the pioneering Java CoG Kit. The Cog Kit has reached wide acceptance in the Grid community and is distributed today also partially with the Globus toolkit. However, we have enhanced the CoG Kit in regards to workflow management and fault tolletrance, and a new toolkit that we call *cyberaide*. This toolkit strives to provide a number of significant enhancements such as a portal interface, an easier workflow model, and a command shell that enables a more convenient interface to use Grid and Clouds. We use the cyberaide-based middleware for solving the CSC enabling the transparent use of workflow models and parallel calculations in order to achieve fast turnaround. These two concepts are used within the CSC as follows:

- the optimization activities are controlled and implemented with the help of the cyberaide and Java CoG Kit workflow framework, and
- the simulation of the WDS is controlled through the coordinated use of parallelized simulations that are executed via multiple PEPANET [2] servers staged on the Grid resources.

Our contribution in this paper are: (a) we solve the CSC problem by parallelizing the GA with the Grid workflow paradigm, (b) we prototyped a cyberinfrastructure middleware called *cyberaide* to implement the CSC on a production Grid such as the TeraGrid, (c) We developed a HAdoop based implementation to contrast the algorithm design with a more traditional HPC approach.

The rest of this paper is organized as follows. Section 2 provides a brief overview of related work. Section 3 defines a more formal definition of the Contamination Source Characterization problem. Section 4 describes briefly the EPANET software that serves the main computational simulation engine for the water distribution system.

Section 5 presents the optimization model that is based on a GA and Section 6 investigates the parallel implementation of the GA while utilizing the cyberaide workflow framework. Section 7 evaluates our implementations on the TeraGrid. IN Section 8 we describe our Hadoop-based solution and in Section 9 we summarize our conclusion.

2 Relationship to Grids, Clouds, and Previous Work

In this section, we introduce briefly the concept of a production Grid and our lightweight cyberaide toolkit for accessing production Grid resources.

2.1 Grids

A production Grid [3] is a shared computing infrastructure of hardware, software, and knowledge resources that allows the coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations to enable sophisticated international scientific and business oriented collaborations. Most often Grids contain resources and services that allow to conduct high-performance and/or high-end computation. Recently Grids have been expanded to data Grids serving data intensive scientific applications. Hence, elementary characteristics of Grids integrate a large-scale distributed computing infrastructure across multiple administrative domains. A production Grid can be accessed though a shared security infrastructure defined by policies and rules between the institutions.

Production Grids use Grid middleware to coordinate its resources and services. The Globus Toolkit [4] that includes a subset of the Java CoG Kit [5, 6], and gLite[7] are examples for popular Grid Middleware. TeraGrid [8, 9] includes the Globus Toolkit as one of its software and service offerings. For data transfer it uses the Globus Toolkit GridFTP [10] services.

Due to the nature of the applications that motivated Grid computing, Grid toolkits employ services that allow uniform job submissions to take place among the diverse set of computational resources, hiding the intrinsic differences between various site-specific batch queues. Hence, Grid users can focus on staging scientific applications while only porting it once to a job execution model rather than implementing a variety of non uniform scripts to address intrinsic differences between the various batch queues deployed at different sites and worrying about getting and managing accounts amongst them.

However, although such services are in place, they often do require a significant investment of time and resources to leverage from site-specific shortcomings to the original goals of Grid computing. Hence it is important to provide lightweight user-level tools and middleware that is attractive even for the non Grid middleware developer to use and to expand its use to less experienced users.

2.2 Clouds

Cloud computing is based on infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) and often allows access to Web 2.0 frameworks. IaaS allows us to abstract the computer infrastructure as part of virtualized resources. Thus in contrast to classical Grids, the virtualization aspect of the resources is an elementary distinction. PaaS enables users of clouds to use solution stacks that simplify the development and deployment of applications to be run in the cloud. It enables us to utilize specific paradigms such as map-reduce within a platform. One such platform is Hadoop that we utilize in our development of a cloud-based solution of our target application. We are using FutureGrid [**?**] as testbed for conducting our cloud experiments with Hadoop. Clouds are a natural evolution from the classical Grid concept. In Fact, Grids with its goal to deliver services to its users has many overlapping features.

2.3 Cyberaide: A Lightweight toolkit for Grid job management

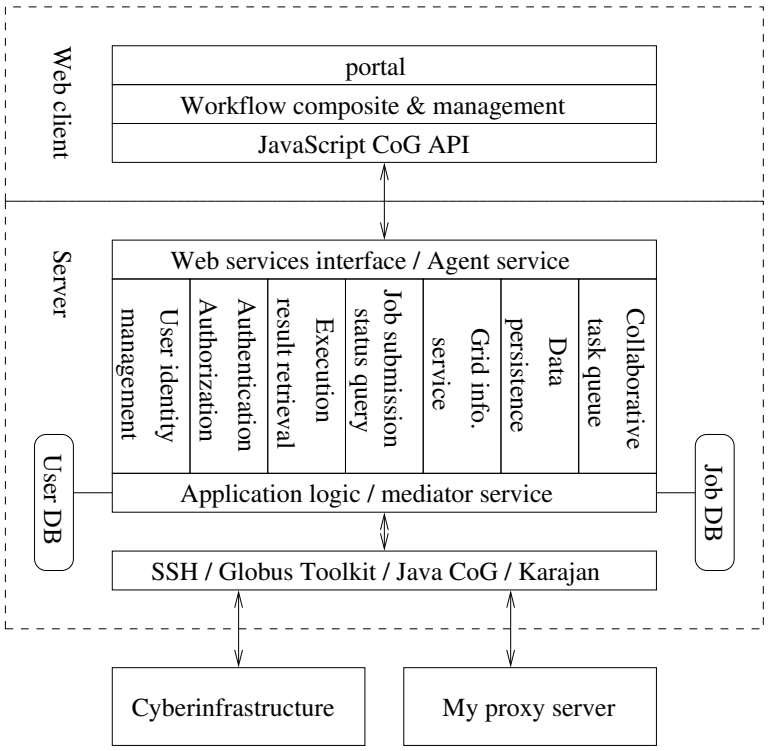

Figure 1: System Architecture of the Cyberaide toolkit

Cyberaide is a lightweight middleware toolkit designed to simplify the interaction with resources that may be hosted in multiple Grids, or clusters. This allows users to formulate their own view of a virtual user-centric cyberinfrastructure. Hence he is not limited to using resources in a single Grid. We have seen that such environments are

needed in case users have access to their own clusters, or multiple Grids that are not integrated into the same security infrastructure. Furthermore, it should be possible to integrate new resources the user has access to without having additional special permissions other than having a user account. The Java CoG Kit provides a subset of the underlying service functionality to enable such a user-centric cyberinfrastructure. Although the Java CoG Kit is now also distributed in part with the Globus Toolkit it does not include the many enhancements that we rely on in cyberaide. These are provided as part of the main Java CoG Kit distribution. Additionally, we have significantly enhanced the logic and services and incorporated them into a toolkit named cyberaide. Cyberaide is available as prototype. The latest enhancements include the following features:

Ease of use. Cyberaide strives to be easier to use for a variety of users while providing a number of access methodologies. This includes not only a Java, but also a Javascript API, a portal interface, and a sophisticated command line and language interface that we are used to from tools such as matlab and mathematica.

Low installation footprint. IN order for the toolkit to be useful, it must be easy to install the client and server services. Complex setup of enhancements to the client side must be avoided. Thus, cyberaide uses a mediator service to which clients communicate via SSL instead of using GSI. The mediator delegates service requests accordingly.

Security. Cyberaide gains access to Grid resources through common security protocols Integration of SSL based and GSI based compute resources and services is possible. Special attention has been put forward in the design of a mediator framework to allow access to Grid resources right from Javascript.

Basic Grid functionality. Cyberaide provides developers access to Grid-based services via abstractions. It may not provide all functionality exposed to by a Grid toolkit but allows the utilization of elementary actions such as filet transfer and job submission. One advantage of this abstraction is that workflow in cyberaide could be viewed as a PaaS in which we use the abstraction, but it is realized and implemented with different backends, such as the powerful Java CoG kit workflow engine or simple a workflow system provided by a queuing system such as Moab.

Advanced functionality. Cyberaide offers advanced user centric functionality that is not provided by regular toolkits such as Globus. An example would be a command line interface that allows users to specifically manage their jobs on the Grid.

Cyberaide is based on a layered architectural design in order to allow its evolution over time and to separate concerns (see Figure 1). The most important component of cyberaide is the *mediator service* that allows easy adaptation and integration of Grid, Cloud, and network of workstation based infrastructure. Access to a mediator service is secured and must be enabled through configuration.

2.4 Cyberaide Workflow Concepts

The Cyberaide workflow concepts include three distinct abilities. First, through the Java CoG Kit Karajan workflow [11, 5] engine we provide the ability to orchestrate workflow like tasks with language constructs such as parallel blocks, loops, direct acyclic graphs, futures, task abstractions, and fault tolerant behavior of tasks. Additionally, we can leverage from the unique feature that the workflow can be manipulated at runtime and can be integrated with the Java COG Kit coaster framework allowing the utilization of a reserved resource to receive multiple tasks as a time without re-authentication and staging of input data. This can save an incredible amount of time, especially if the cost for providing an ab initio data set is high.

The Java CoG Kit Karajan workflow specification language is a common purpose XML-based language, which can express complicated workflows including conditional control flows and loops. A functional programming language formulation of Karajan also exists making it possible to define workflows either through XML or through function definitions. The execution model is based on executing elements representing for example jobs and to react upon events generated during the workflow execution. Of especial interest is the ability to replace the execution engine governing the workflow execution. This allows users and developers to provide their own strategies that may be influenced by dynamic events. Basic checkpointing has been also implemented on the workflow level (the workflow can be checkpointed, when all its elements are in consistent states).

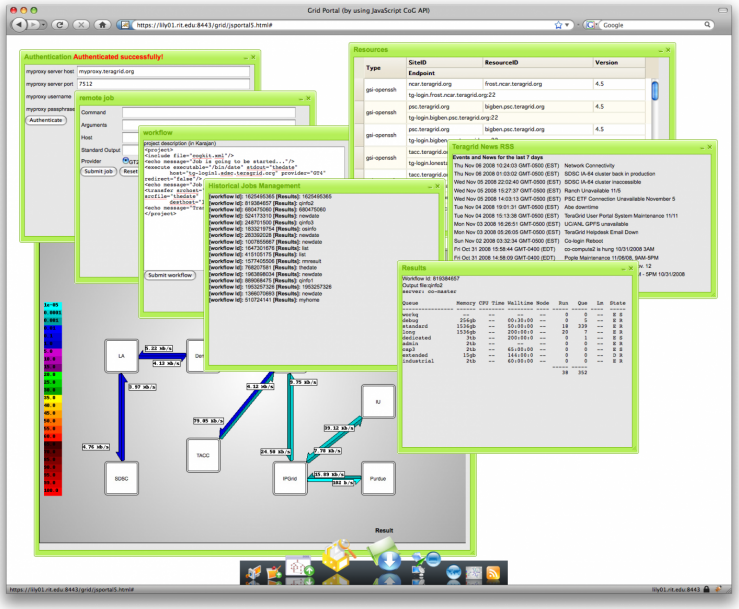

Figure 2: Cyberaide workflow portal

The Cyberaide workflow portal that we prototyped is based on a service oriented architecture that allows users to access Grid infrastructure through JavaScript [12].

Such a framework integrates well with other Web 2.0 technologies since it provides JavaScript toolkit to build web applications. The framework consists of two essential parts. A client Application Programming Interface (API) to access the Grid via JavaScript and a full service stack in server side through which the Grid access is channeled. The framework uses commodity Web service standards and provides extended functionality such as asynchronous task management, file transfer. The availability of this framework simplifies not only the development of new services, but also the development of advanced client side Grid applications that can be accessed through Web browsers. Figure 2 shows an example of Cyberaide workflow portal.

3 Contamination source characterization in urban Water Distribution Systems

3.1 Problem definition

As mentioned earlier, CSC [13] in a Water Distribution System (WDS) is to find the contaminant source locations and their temporal mass loading history. The temporal mass history is defined through values such as as THE start time of the contaminant release, duration of release, and the contaminant mass loading during this time. An essential problem is to identify appropriate locations for the sensors to minimize the time needed to detect a contamination. A few nodes are selected as the sensor locations in the Water Distribution System. During a contamination event, concentration readings are obtained at these locations at specified time intervals. Contamination sources are assumed to be present at arbitrary locations in the WDS for a predetermined period of time.

To identify the true contaminant sources, an estimate is generated for the contamination source locations and their release histories i.e. start time of the contaminant, duration and the amount of contaminant present. For every such estimate, the water quality simulation is run to obtain the resulting concentration readings for all the sensor locations for every time step.

The problem of CSC in a WDS is to locate optimized estimated contaminant source locations by minimizing the differences between concentration readings of estimations and those of real measurement from sensors.

The problem of CSC in a WDS is to find a characterization of contaminant sources $S = (X, H(X,t), t_0)$,

$$\min \sum_{i=1}^{I} \sum_{t=t_0}^{t_{\text{end}}} |O(i,t) - C(i,t,S)| \tag{1}$$

where,

- i: i^{th} sensor,

- I is the number of sensors,
- t: time of simulation,
- t_0: estimated start time of a contaminant sources,
- t_{end}: end time of simulation,
- X: contamination source locations, for CSC with multiple contaminant sources, X is a set of contaminant source locations: $X = (x_1, x_2, ..., x_k)^T$, where k is number of contaminant sources
- $H(X,t)$: contaminant mass loading, which is contaminant concentration of sources at time t: $H(X,t) = (h_1(t), h_2(t), ..., h_k(t))^T$
- $O(i,t)$: observed concentration of sensor i at time t,
- S: characterization of contaminant sources with contaminant sources of X, contaminant mass loading of $H(X,t)$, and starting time of t_0, and
- $C(i,t,S)$: calculated concentration of sensor i at time t via a WDS simulation with an estimated characterization of contaminant sources S.

Running such experiments multiple times and obtaining variations on the locations for sensors with the goal to minimize the time for feedback can significantly reduce the cost of the sensor placement.

3.2 Simulation and optimization framework

To solve aforementioned problem an optimization framework is introduced that uses simulations to obtain the result (see Figure 3) [14]. An optimization model is coupled with the simulation model by providing various input trial variables and retrieving simulated results. The optimization model generates optimized trials for the characterization of the contaminant sources $S = (X, H(X,t), t_0)$ as part of the CSC problem. These optimized trials are than send to the simulation model. Together with other input data such as the network model of a WDS, the simulation model computes the output determining the contaminant concentrations at the locations of sensors. The simulation outputs are compared with the observational sensor data to be integrated into a self-correction. If however the error, between the simulation and the observed values is bellow a threshold the optimization and simulation approach is terminated.

4 PEPANET as a simulation engine for CSC

The EPANET software [15] models a water distribution piping systems and performs the simulation of the hydraulic and water quality behavior within a pressurized network of water pipes. The EPANET computer program [16], developed by the U.S.

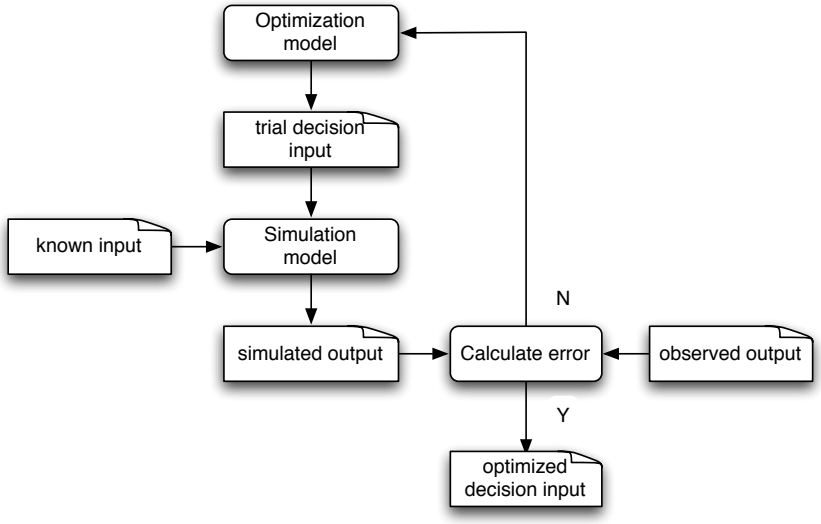

Figure 3: The simulation and optimization approach

EPA (Environmental Protection Agency), solves the nonlinear energy equations and linear mass equations for pressures at nodes and flowrates in pipes. The EPANET reads the data file that defines the characteristics of the pipes, the nodes (connection points of the pipe), and the control components (such as pumps and valves) in the pipe network. The calculation of flowrates involves several iterations because the mass and energy equations are nonlinear. The number of iterations depends on the system of network equations and the user-specified accuracy. A satisfactory solution of the flowrates must meet the specified accuracy, the law of conservation of mass and energy in the Water Distribution System, and any other requirements imposed by the user. The calculation of HGL requires no iteration because the network equations are linear. Once the flowrate analysis is complete, the water quality computations are then performed.

A parallelized version of EPANET called PEPANET was developed by our partners at NCSU. This MPI based software is available to simulate the water flow in a WDS and a Parallel Genetic Algorithm is used as the optimization model.

5 Genetic algorithm as an optimization engine for CSC

The genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution to generate useful solutions to optimization and search problems. We uses a PGA in the optimization model to find optimized Contamination Sources S in the CSC problem. Algorithm 1 shows the GA skeleton for the CSC in a WDS. In the GA, a trial Contamination Source S is encoded into an individual of populations. The simulation time is discretized into multiple time steps. In the each time step, a GA is applied with multiple generations to reach an optimized S. The evaluation process of

individuals includes the following steps:

1. Calculated the simulated output via the simulation model implemented by the PEPANET software

2. Calculated the fitness, which is the prediction error as follows:

$$f = \sum_{i}^{I} |O(i,t) - C(i,t,S)| \tag{2}$$

The GA optimization is done in all time steps until the simulation time reaches the end time t_{end}.

Algorithm 1 WTM application skeleton

```
1  t := t_0
2  g := 0
3  individuals := assign at random;
4  generate a set of N sub-populations;
5  while t = t_end do
6  t := t+1
7      while the predefined termination criteria is not reached do
8          g := g+1
9          foreach sub-population do
10           apply GA operators: selection, replication, mutation;
11         end
12         calculate the fitness foreach individual
13         calculate the distances between all sub-populations;
14     end
15 end
```

6 Parallel contamination source characterization with Cyberaide workflow

Instead of using the PEPANET MPI based code, we have transformed the code to be executed as part of a workflow. This allows us to decouple the computation of CSC by distributing tasks and orchestrating them with cyberaide workflow framework. This includes the interplay between a multi-population GA and the PEPANET simulations.

MGA: MGA is a framework for multi-population based niched co-evolutionary approach based evolutionary strategy for generating multiple alternative solutions for the Water Distribution System contaminant source identification problem. The MGA code is the optimization engine of the simulation – optimization framework. It calls the simulation component: a persistent wrapper residing

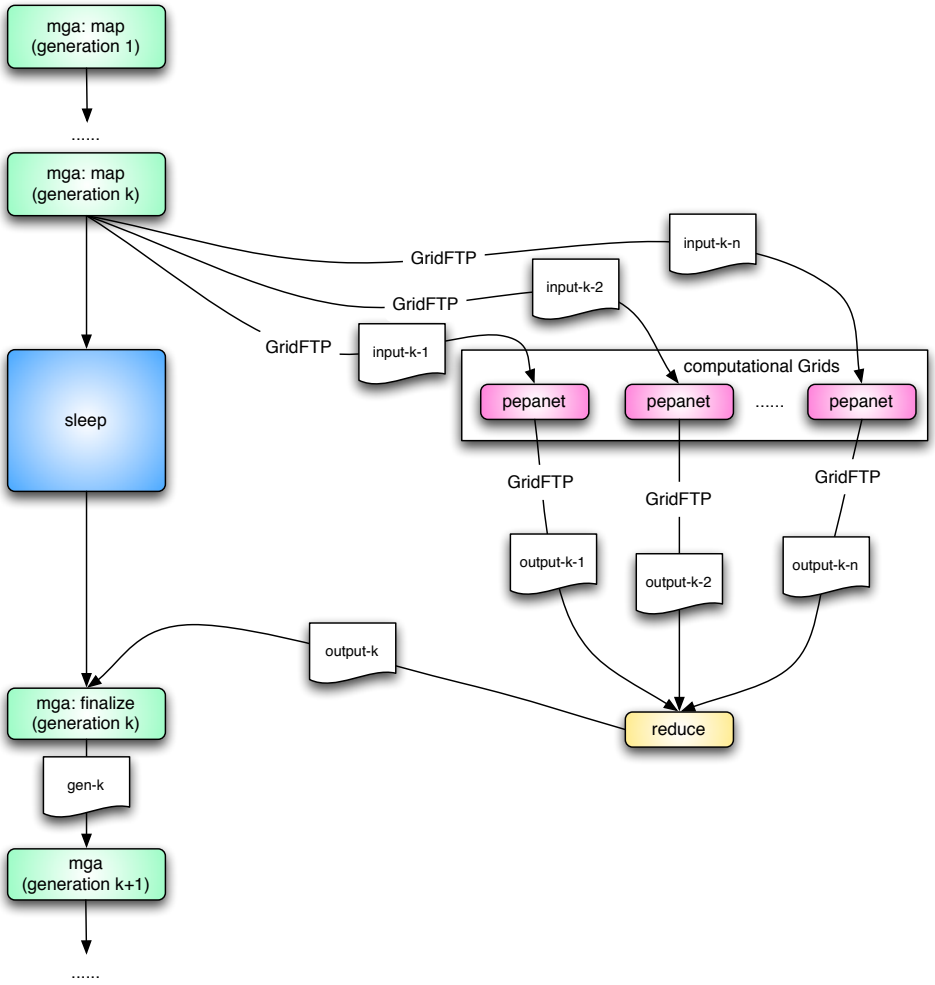

Figure 4: Parallel contamination source characterization with Cyberaide workflow

in the directory "PEPANET" to carry out EPANET simulations in parallel. For the water distribution problem, solutions are represented as a location (node number), start time with an integer and contaminant loading profile as a list.

PEPANET: The PEPANET is the parallel version of EPANET simulator [13]. It receives a number of contamination source parameters from an input file and divides them into multiple file chunks to different EPANET servers to compute. The communication between EPANET servers is done using MPI.

REDUCE: The REDUCE code collects and merges computing results from multiple EPANET servers, and sends them to MGA for individual evaluation.

The Cyberaide workflow fort the CSC is shown in Figure 4. The MGA code is executed iteratively on multiple generations at the client. At each generation, it sends input data for PEPANET computation via GridFTP. The PEPANET servers are launched

Table 1: Testbed setup in TeraGrid

Compute site	Resource description
Abe (NCSA)	Dell blade system: 1200 PowerEdge 1955 nodes Each node: dual socket, quad core compute blades InfiniBand interconnect 100 TB of storage in a Lustre filesystem
Big Red (IU)	IBM e1350: 768 IBM JS21 compute nodes Each node: two dual-core 2.5 GHz PowerPC 970MP CPUs 8 GB memory 72 GB of local scratch disk
Queen Bee (LONI)	Dell PowerEdge 1950 cluster: 668 nodes. Each node: two Quad Core Intel Xeon 2.33GHz 64-bit processors 8 GB of memory 10 Gb/sec Infniband 192TB (raw) of storage in Lustre a file system .

on remote Grid resources and simulate the urban Water Distribution System. After the simulation finishes, the results are sent back to the REDUCE code via GridFTP. The MGA code resumes when all results are due, it then evaluates all individuals and proceeds to the next GA generation.

7 Performance evaluation and discussion

Our experiments are conducted on a number of TeraGrid compute resources [17]. TeraGrid includes 11 supercomputing centers across USA. We test the performance of the CSC application with our implementation while using three of them: Abe at National Center for Supercomputing Application (NCSA), Big Red at Indian University (IU), and Queen Bee at the Louisiana Optical Network Initiative (LONI). The reason why we chose these three resources is based on the fact that the application could be easily ported to them. Other resources on TeraGrid provided not the right libraries or software stacks to run the application successfully. The test bed is described in Table 1. We execute the CSC simulation with Cyberaide workflow on the above resources of the TeraGrid. At each resource 16 processors are allocated for the CSC problem calculation.

7.1 Performance evaluation and discussion

Figure 5 shows the CSC task execution time on Big Red, Abe and Queen Bee with different GA generation numbers. It shows that on each compute resource (Big Red, Abe

and Queen Bee) task execution time is proportional to the number of GA generation. This can be explained as follows:

- The execution time of simulating the WDS is determined largely by the input data size and compute resource capacity. As we calculate the same WDS and the input data is the same for all tests, the time to execute the simulations is proportional to the compute resource capacity.
- The data transferred between clients and TeraGrid resources (Big Red, Abe and Queen Bee) includes the WSD system information and input trial individuals. The WDS system information contains the WDS parameters and layouts, whose size is around several tens of megabytes in total for our test problem. These files can be uploaded to TeraGrid resources before the test and remain unchanged during the execution. For a simulation of each input individual, the input data is around several kilobytes. The data transfer time of input data can be ignored considering the high-speed network of the TeraGrid (typically 1 Gigabit/S or 10 Gigabit/S).

Such performance characteristic can be easily included in better utilization of the distributed resources in the TeraGrid for this application.

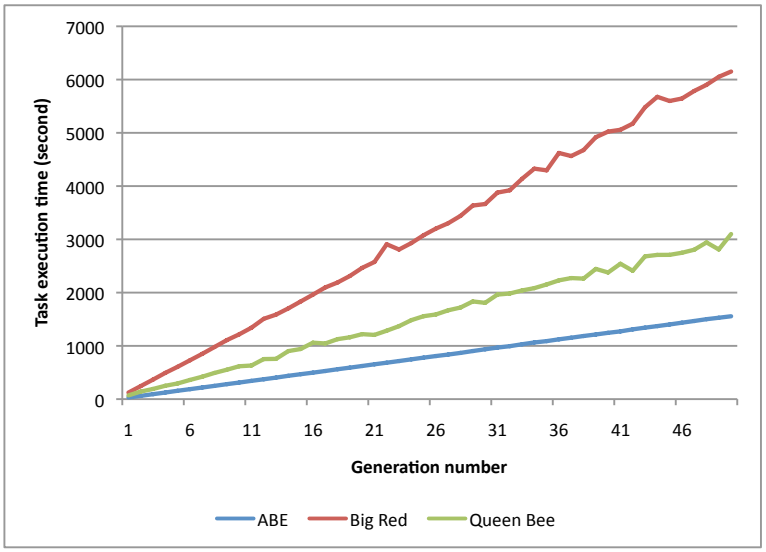

Figure 5: Task execution time vs. generation number

8 Hadoop Implementation on FutureGrid

We implemented the application while using the Hadoop methodology. To evaluate the Hadoop implementation performance, we also modified the original program and implemented a WTM in a master/slave paradigm. To achieve this modified algorithm,

we created in each generation of the Genetic Algorithm that is inherent in the application, a master that divides multiple individuals to sub-tasks that are then sent to slaves for individual evaluation. To allow the use of distributed resources the communication between master and slaves is protected via SSL allowing the use of tools such as ssh and scp to coordinate execution and copying of results. We refer to this implementation as the ?SSH? implementation. We used the india cluster from the FutureGrid project to conduct performance experiments. Figures 6 and 7 show the performance of the hadoop vs the ssh implementation while comparing it with the number of nodes used. We observer that the ssh based solution is more efficient than the haddop based solution.

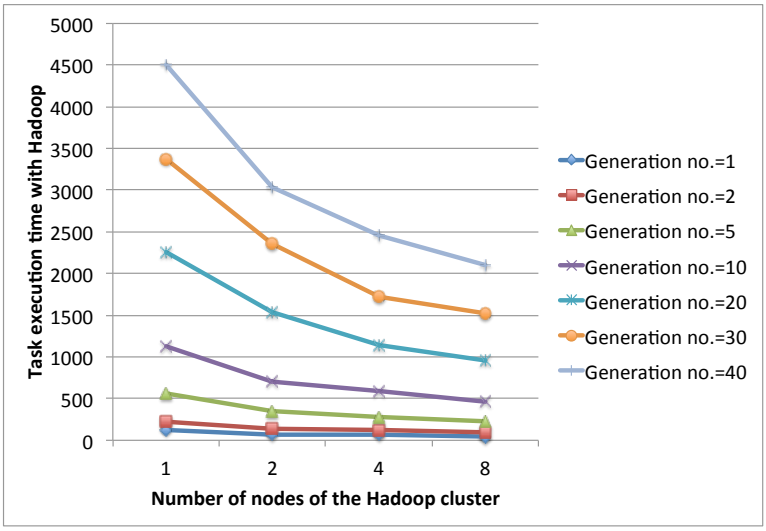

Figure 6: Task execution time in hadoop vs. number of nodes

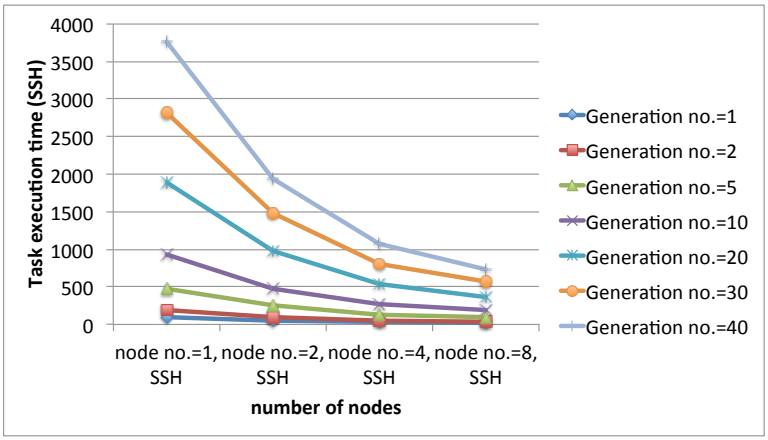

Figure 7: Task execution time using ssh vs. number of nodes

9 Conclusion

The Contaminant Source Characterization (CSC) in a Water Distribution System (WDS) is a critical research issue due the importance of Urban Water Distribution Systems. The CSC calculation is a compute-intensive application that typically requires high performance computing resources. In this paper, we solve the CSC problem on the supercomputing resources of the TeraGrid project with the Cyberaide workflow. We developed the Cyberaide workflow engine, portal and wrapper service for executing the optimization – simulation framework of the CSC application. Test results of CSC problem on the TeraGrid resources justify our design and implementation of the Cyberaide workflow system.

Although we have shown that one could design a master slave framework including fault tolerance with the Java CoG Kit Karajan workflow engine (as we have done), Hadoop provides also an implicit framework for executing map reduce like tasks in parallel. The advantage of using a framework such as Karajan is the ability to develop sophisticated fault tolerant services with custom behaviors. The advantage of using a framework such as Hadoop is that fault tolerant mechanisms are provided by default with little effort. Effort to use frameworks such as map reduce map/reduce is non trivial and poses the need for the engineers to be familiar not only with map reduce but also with the target application. However the cost for such an implementation is punished by significant performance overhead. In addition we observed that the use of FutureGrid provided us with a complete new way of implementing a solution to the problem that was not accessible to us before.

Acknowledgment

This work, conducted by Gregor von Laszewski and Lizhe Wang, is supported (in part) by NSF CMMI 0540076, NSF SDCI NMI 0721656 and 2010 HP Annual Innovation Research Awards. The original versions of MGA and PEPANET software with MPI were implemented by Dr. Jitendra Kumar and Mr. Sarat Sreepathi from North Carolina State University.

References

[1] S. Sreepathi, K. Mahinthakumar, E. Zechman, R. Ranjithan, D. Brill, X. Ma, and G. von Laszewski, "Cyberinfrastructure for Contamination Source Characterization in Water Distribution Systems," in *Proceedings of the INternational Conference on Computational Science, ICCS 2007*, ser. Lecture Notes in Computer Science, vol. 4487. Springer, 2007, p. 1058.

[2] G. von Laszewski, K. Mahinthakumar, R. Ranjithan, D. Brill, J. Uber, K. Harrison, S. Sreepathi, and E. Zechman, "An Adaptive Cyberinfrastructure for Threat

Management in Urban Water Distribution Systems," in *Proceedings of the International Conference on Computational Science, ICCS 2006*, vol. 3993, 2006, pp. 401–408.

[3] G. von Laszewski, "The Grid-Idea and Its Evolution," *Journal of Information Technology*, vol. 47, no. 6, pp. 319–329, Jun. 2005.

[4] "The Globus Toolkit." [Online]. Available: http://www.globus.org

[5] G. von Laszewski, "Java CoG Kit Workflow Concepts," *Journal of Grid Computing*, Jan. 2006, http://dx.doi.org/10.1007/s10723-005-9013-5.

[6] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8-9, pp. 643–662, 2001.

[7] "gLite." [Online]. Available: http://glite.web.cern.ch/glite/

[8] P. Beckman, "Building the TeraGrid," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 363, no. 1833, pp. 1715–1728, 2005.

[9] C. Catlett, "The Philosophy of TeraGrid: Building an Open, Extensible, Distributed TeraScale Facility," in *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, 2002, pp. 8–8.

[10] "The GridFTP Protocol and Software." [Online]. Available: http://www.globus.org/datagrid/gridftp.html

[11] M. Hategan, G. von Laszewski, and K. Amin, "Karajan: A Grid Orchestration Framework," Supercomputing 2004, Pittsburgh, 6-12 Nov. 2004, (Refereed Poster). [Online]. Available: http://www.sc-conference.org/sc2004

[12] G. von Laszewski, F. Wang, A. Younge, Z. Guo, and M. Pierce, "JavaScript Grid Abstractions," in *Proceedings of the Grid Computing Environments 2007 at SC07*, Reno, NV, Nov. 2007. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/07-javascript/vonLaszewski-07-javascript.pdf

[13] S. Sreepathi, "Cyberinfrastructure for contamination source characteraization in water distribution systems," Master's thesis, North Carolina State University, Raleigh, North Carolina, USA, 2006.

[14] B. Y. Mirghania, K. G. Mahinthakumara, M. E. Trybya, R. S. Ranjithana, and E. M. Zechman, "A parallel evolutionary strategy based simulation-optimization approach for solving groundwater source identification problems," *Advances in Water Resources*, vol. 32, no. 9, pp. 1373–1385, 2009.

[15] "Epanet," [Online], http://www.epa.gov/nrmrl/wswrd/dw/epanet.html.

[16] L. Rossman, “EPANET 2 users manual,” US Environmental Protection Agency, Cincinnati, Ohio, Tech. Rep., 2000.

[17] “TeraGrid,” 2001. [Online]. Available: http://www.teragrid.org/