2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum

# A Simulation Study on Urban Water Threat Detection in Modern Cyberinfrastructures

Lizhe Wang[1,2], Dan Chen[2], Ze Deng[2], Rajiv Ranjan[3]
1 Center for Earth Observation and Digital Earth, Chinese Academy of Sciences, P. R. China
2 School of Computer Science, China University of Geosciences, P. R. China
3 ICT Centre, CSIRO, Australia

***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.**

**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 middleware 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 automatically includes fault tolerance. The later activity has been conducted on FutureGrid.**

## I. 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 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 considerably large.

Hence, we present a solution that utilizes Grid and Cloud concepts. The Grid portion of this 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. We have enhanced the CoG Kit in regards to workflow management and fault tolerance. 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 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 Grid 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 contributions 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 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 II provides a brief overview of related work. Section III defines a more formal definition of the Contamination Source Characterization problem. Section IV describes briefly the EPANET software that serves the main computational simulation engine for the water distribution system.

Section V presents the optimization model that is based on a GA and Section VI investigates the parallel implementation of the GA while utilizing the Grid workflow framework. Section VII evaluates our implementations on the TeraGrid. IN Section VIII we describe our Hadoop-based solution and in Section IX we summarize our conclusion.

## II. Relationship to Grids, Clouds, and Previous Work

In this section, we introduce briefly the concept of a production Grid computing and Cloud computing.

978-0-7695-4676-6/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPSW.2012.127

### A. Grids

A production Grid 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 the user 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 [3], and gLite[4] are examples for popular Grid Middleware. TeraGrid [5] includes the Globus Toolkit as one of its software and service offerings. For data transfer it uses the Globus Toolkit GridFTP [6] 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.

### B. Cloud computing and MapReduce

A computing Cloud is a set of network enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing infrastructures on demand, which can be accessed in a simple and pervasive way [7], [8], [9]. Conceptually, users acquire computing platforms, or IT infrastructures, from computing Clouds and execute their applications inside. Therefore, computing Clouds render users with services to access hardware, software and data resources, thereafter an integrated computing platform as a service. The MapReduce paradigm and its open-sourced implementation – Hadoop have recognized as representative enabling technique for Cloud computing.

*1) MapReduce paradigm:* The MapReduce [10] programming model is inspired by two main functions commonly used in functional programming: Map and Reduce. The Map function processes key/value pairs to generate a set of intermediate key/value pairs and the Reduce function merges all the same intermediate values. Many real-world applications are expressed using this model.

The most popular implementation of the MapReduce model is the Hadoop framework [11], which allows applications to run on large clusters built from commodity hardware. The Hadoop framework transparently provides both reliability and data transfer.

Other MapReduce implementations are available for various architectures, such as for CUDA [12], in a multicore architecture [13], in FPGA platforms [14], for a multiprocessor architecture [15], in a large-scale shared-memory system [16], in a large-scale cluster [17], in a streaming runtime environment [18], in a Grid environment [19], in an opportunistic environment [20], and in a mobile computing environment [21].

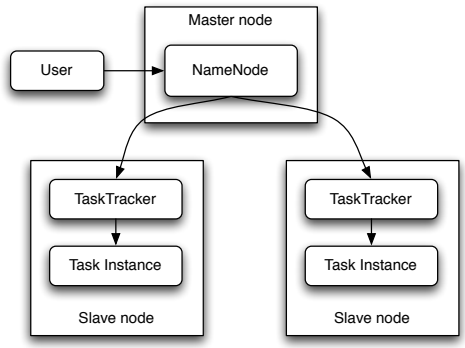


Fig. 1. Hadoop MapReduce

*2) Hadoop:* The MapReduce programming model is designed to process large volumes of data in parallel by dividing the *Job* into a set of independent *Task*s. The *Job* refers here as a full MapReduce program, which is the execution of a *Mapper* or *Reducer* across a set of data. A *Task* is an execution of a *Mapper* or *Reducer* on a slice of data. So the MapReduce *Job* usually splits the input data set into independent chunks, which are processed by the *map* tasks in a completely parallel manner.

The Hadoop MapReduce framework consists of a single Master node that runs a *Jobtracker* instance which accepts *Job* requests from a client node and *Slave node*s running each a *TaskTracker* instance. The *Jobtracker* assumes the responsibility of distributing the software configuration to the Slave nodes, scheduling the job's component tasks on the Tasktrackers, monitoring them and re-assigning tasks to the *TaskTracker*s when they failed. It is also responsible for providing the status and diagnostic information to the client. The *TaskTracker*s execute the tasks as directed by the *JobTracker*. The *TaskTracker* executes tasks in separate java process so that several task instances can be performed in parallel at the same time. Figure 1 depicts the different components of the MapReduce framework.

Figure 2 illustrates the high-level pipeline of the Hadoop

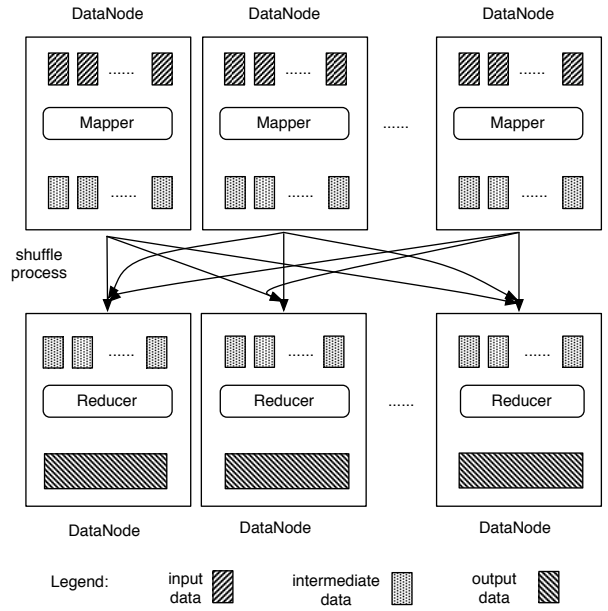


Fig. 2. Hadoop high level data flow

MapReduce. The MapReduce input data typically come from the input files loaded into the HDFS. These files are evenly distributed across all the nodes in the cluster. In Hadoop, computer nodes and data nodes are all the same meaning that the MapReduce and HDFS run on the same set of nodes. At the mapping phase, the input file is divided into independent InputSplits and each split of these Splits describes a unit of work that comprises a single map task in the MapReduce job. The map tasks are then assigned to the nodes in the system based on the physically residence of the input file splits. Several map tasks can be assigned to an individual node, which attempts to perform tasks as many in parallel as it can. When the mapping phase has completed, the intermediate outputs of the map tasks are exchanged between all nodes; and they are also the input of the reduction tasks. This process of exchanging the map intermediate outputs is known as the shuffling. The reduce tasks are spread across the same nodes in the cluster as the mappers. The output of the reduce tasks are stored locally on the slave node.

## III. Contamination source characterization in urban Water Distribution Systems

### A. Problem definition

As mentioned earlier, CSC [22] 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,
- $t_{\text{end}}$: 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.

### B. Simulation and optimization framework

To solve the aforementioned problem, an optimization framework is introduced that uses simulations to obtain the result (see Figure 3) [23]. 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 then sent 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 below a threshold, the optimization and simulation approach is terminated.

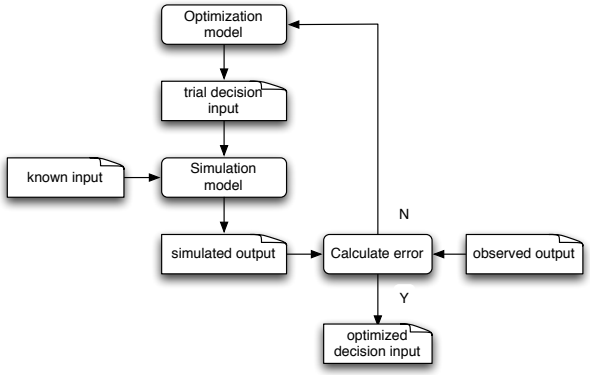


Fig. 3. The simulation and optimization approach

## IV. PEPANET AS A SIMULATION ENGINE FOR CSC

The EPANET software [24] models 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 [25], developed by the U.S. 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.

## V. 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 use 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 as individuals within a population. The simulation time is discretized into multiple time steps. In 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:

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

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

**Algorithm 1** WTM application skeleton

1) $t \leftarrow t_0$
2) $g \leftarrow 0$
3) Randomly encodes individuals and generate a set of $N$ sub-populations
4) **REPEAT**
5) $t \leftarrow t + 1$
6) **REPEAT**
7) $g \leftarrow g + 1$
8) In each sub-population, apply GA operations such as **selection**, **replication** and **mutation**.
9) calculate the fitness of each individual and distances between sub-populations
10) **UNTILL** the predefined criteria is reached.
11) **UNTILL** $t = t_{\text{end}}$

Algorithm 1 illustrates the GA variant customized for CSC. In this particular algorithm, a trial of contamination source $S$ is encoded into an individual of a population (line 3). We assume the simulation progresses in a sequence of discrete time steps. For each time step, the GA evolves for multiple generations (line 6 – 10) to obtain an optimized $S$ within these generations. The GA-based optimization runs through all time steps until the simulation end time $t_{\text{end}}$ (line 11). The individuals (trials of contamination source) are evaluated in following steps:

1) Calculate the simulated output via the simulation model implemented by the EPANET software,
2) Calculate the fitness, which is prediction error as follows:

$$f = \sum_{i}^{I} |O(i,t) - C(i,t,S)| \quad (3)$$

## VI. PARALLEL CONTAMINATION SOURCE CHARACTERIZATION WITH GRID 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 Grid workflow framework. This includes the interplay between a multi-population GA and the PEPANET simulations.

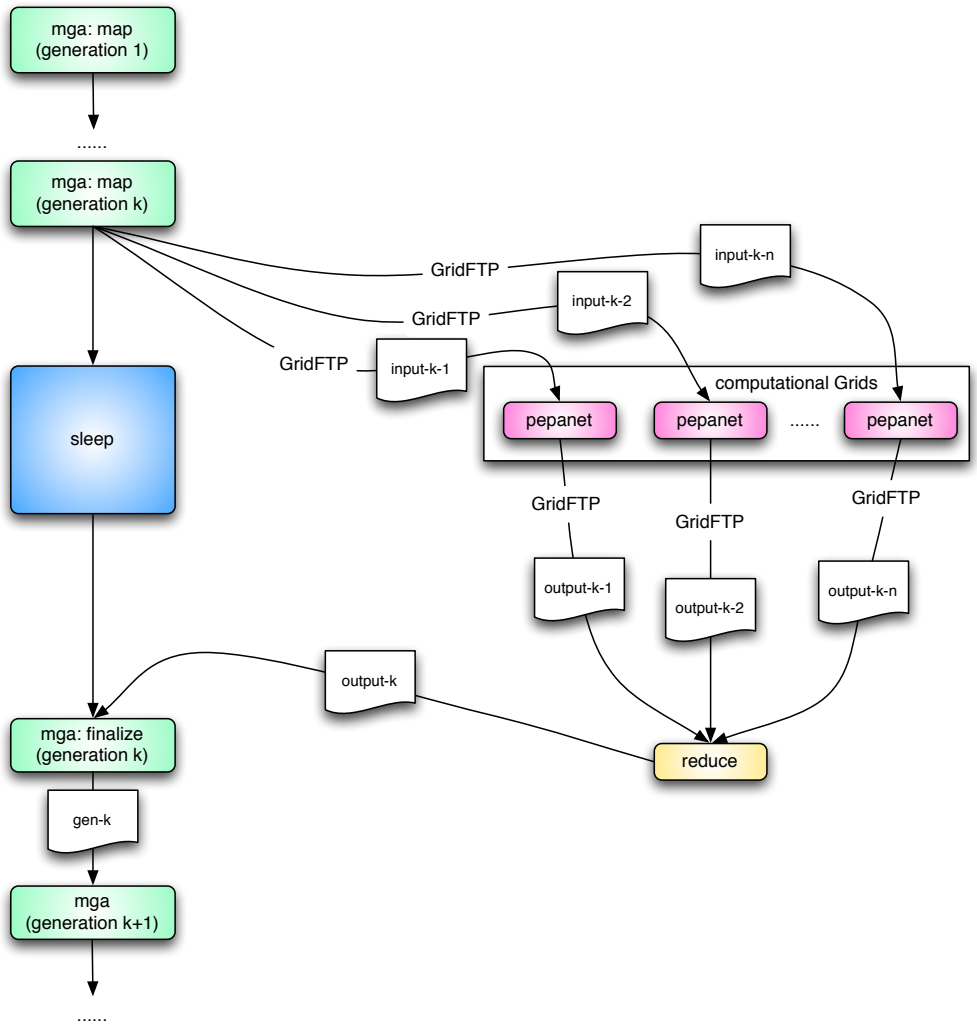


Fig. 4. Parallel contamination source characterization with Grid workflow

- 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 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 [22]. 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 for 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 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.

## VII. Performance Evaluation and Discussion

Our experiments are conducted on a number of TeraGrid compute resources [26]. TeraGrid includes 11 supercomputing

TABLE I
TESTBED SETUP IN TERAGRID

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

centers across the 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 did not provide the right libraries or software stacks to run the application successfully. The test bed is described in Table I. We executed 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.

### A. 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.

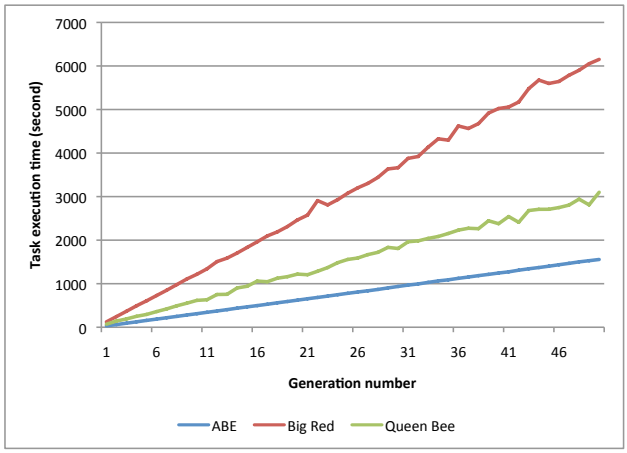


Fig. 5. Task execution time vs. generation number

## VIII. 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 hadoop based solution.

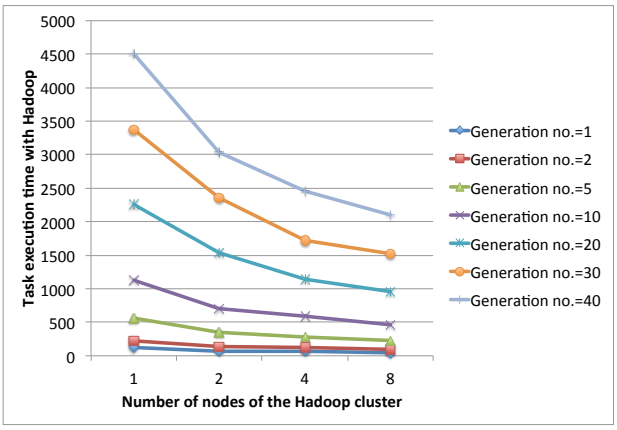


Fig. 6. Task execution time in hadoop vs. number of nodes

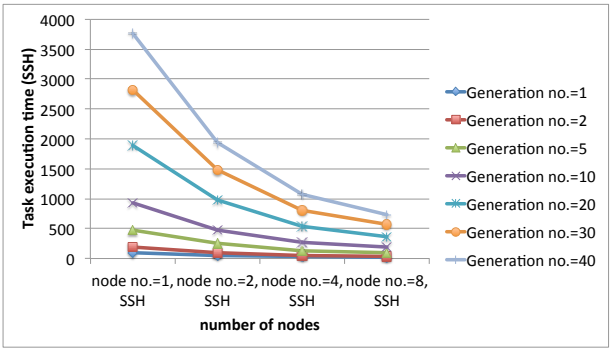


Fig. 7. Task execution time using ssh vs. number of nodes

## IX. Conclusion

The Contaminant Source Characterization (CSC) in a Water Distribution System 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 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 magnified 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

Dr. Lizhe Wang's work is funded by "One-Hundred Talents Program" of The Chinese Academy of Sciences.

Dr. Dan Chen is funded by National Natural Science Foundation of China (grant No. 60804036), the Hundred University Talent of Creative Research Excellence Programme (Hebei, China), the Fundamental Research Funds for the Central Universities (CUGL100608, CUG, Wuhan), the Specialized Research Fund for the Doctoral Program of Higher Education (grant No. 20110145110010), the Programme of High-Resolution Earth Observing System (China), and the Program for New Century Excellent Talents in University (grant No. NCET-11-0722). Dr. Dan Chen gratefully acknowledges support from the Birmingham-Warwick Science City Research Alliance.

The original versions of MGA and PEPANET software implemented with MPI were developed by Dr. Jitendra Kumar and Mr. Sarat Sreepathi at 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] "The Globus Toolkit." [Online]. Available: http://www.globus.org

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

[5] 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.

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

[7] L. Wang, G. von Laszewski, A. J. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New Generation Comput.*, vol. 28, no. 2, pp. 137–146, 2010.

[8] L. Wang and C. Fu, "Research advances in modern cyberinfrastructure," *New Generation Comput.*, vol. 28, no. 2, pp. 111–112, 2010.

[9] L. Wang, M. Kunze, J. Tao, and G. von Laszewski, "Towards building a cloud for scientific applications," *Advances in Engineering Software*, vol. 42, no. 9, pp. 714–722, 2011.

[10] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, January 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[11] "Apache hadoop project," Web Page, http://hadoop.apache.org/.

[12] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, "Mars: a mapreduce framework on graphics processors," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, ser. PACT'08. New York, NY, USA: ACM, 2008, pp. 260–269. [Online]. Available: http://doi.acm.org/10.1145/1454115.1454152

[13] R. Chen, H. Chen, and B. Zang, "Tiled-mapreduce: optimizing resource usages of data-parallel applications on multicore with tiling," in *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, ser. PACT'10. New York, NY, USA: ACM, 2010, pp. 523–534. [Online]. Available: http://doi.acm.org/10.1145/1854273.1854337

[14] Y. Shan, B. Wang, J. Yan, Y. Wang, N. Xu, and H. Yang, "Fpmr: Mapreduce framework on fpga," in *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, ser. FPGA'10. New York, NY, USA: ACM, 2010, pp. 93–102. [Online]. Available: http://doi.acm.org/10.1145/1723112.1723129

[15] C. Ranger, R. Raghuraman, A. Penmetsa, G. R. Bradski, and C. Kozyrakis, "Evaluating mapreduce for multi-core and multiprocessor systems," in *13st International Conference on High-Performance Computer Architecture*, 2007, pp. 13–24.

[16] R. M. Yoo, A. Romano, and C. Kozyrakis, "Phoenix rebirth: Scalable mapreduce on a large-scale shared-memory system," in *Proceedings of the 2009 IEEE International Symposium on Workload Characterization*. Austin, TX, USA: IEEE, 2009, pp. 198–207.

[17] M. M. Rafique, B. Rose, A. R. Butt, and D. S. Nikolopoulos, "Supporting mapreduce on large-scale asymmetric multi-core clusters," *SIGOPS Oper. Syst. Rev.*, vol. 43, pp. 25–34, April 2009. [Online]. Available: http://doi.acm.org/10.1145/1531793.1531800

[18] S. Pallickara, J. Ekanayake, and G. Fox, "Granules: A lightweight, streaming runtime for cloud computing with support for map-reduce," in *CLUSTER*. New Orleans, Louisiana, USA: IEEE, 2009, pp. 1–10.

[19] C. Miceli, M. Miceli, S. Jha, H. Kaiser, and A. Merzky, "Programming abstractions for data intensive computing on clouds and grids," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID'09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 478–483. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2009.87

[20] H. Lin, X. Ma, J. Archuleta, W.-c. Feng, M. Gardner, and Z. Zhang, "Moon: Mapreduce on opportunistic environments," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC'10. New York, NY, USA: ACM, 2010, pp. 95–106. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851489

[21] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikainen, and V. H. Tuulos, "Misco: a mapreduce framework for mobile systems," in *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA'10. New York, NY, USA: ACM, 2010, pp. 32:1–32:8. [Online]. Available: http://doi.acm.org/10.1145/1839294.1839332

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

[23] 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.

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

[25] L. Rossman, "EPANET 2 users manual," US Environmental Protection Agency, Cincinnati, Ohio, Tech. Rep., 2000.

[26] "TeraGrid," 2001. [Online]. Available: http://www.teragrid.org/