Cyberinfrastructure for Contamination Source Characterization in Water Distribution Systems

Sarat Sreepathi[1], Kumar Mahinthakumar[1], Emily Zechman[1], Ranji Ranjithan[1] , Downey Brill[1], Xiaosong Ma[1] and Gregor von Laszewski[2]

[1]North Carolina State University, Raleigh, NC, USA
{sarat_s, gmkumar, emzechma, ranji, xma}@ncsu.edu

[2]University of Chicago, Chicago, IL, USA
gregor@mcs.anl.gov

Abstract. This paper describes a preliminary cyberinfrastructure for contaminant characterization in water distribution systems and its deployment on the grid. The cyberinfrastructure consists of the application, middleware and hardware resources. The application core consists of various optimization modules and a simulation module. This paper focuses on the development of specific middleware components of the cyberinfrastructure that enables efficient seamless execution of the application core in a grid environment. The components developed in this research include: (i) a coarse-grained parallel wrapper for the simulation module that includes additional features for persistent execution, (ii) a seamless job submission interface, and (iii) a graphical real time application monitoring tool. The threat management problem used in this research is restricted to contaminant source characterization in water distribution systems. The performance of the cyberinfrastructure is evaluated on a local cluster and distributed clusters on the TeraGrid.

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. Identifying the source and extent of contamination ("source characterization problem") is usually the first step in devising an appropriate response strategy in a contamination incident. This paper develops and tests a preliminary grid cyberinfrastructure for solving this problem as part of a larger multidisciplinary DDDAS [1] project that is developing algorithms and associated middleware tools leading to a full fledged cyberinfrastructure for threat management in WDSs [2].

The source characterization problem involves finding the contaminant source location (typically a node in a water distribution system) and its temporal mass loading history ("release history") from observed concentrations at several sensor locations in the network. The release history includes start time of the contaminant release in the WDS, duration of release, and the contaminant mass loading during this time. Assuming that we have a "forward simulation model" that can simulate concentrations at various sensor locations in the WDS for given source characteristics, the source characterization problem, which is an "inverse problem", can be formulated as an optimization problem with the goal of finding a source that

can minimize the difference between the simulated and observed concentrations at the sensor nodes. This approach is commonly termed "simulation-optimization" as the optimization algorithm drives a simulation model to solve the problem. Population based search methods such as evolutionary algorithms (EAs) are popular methods to solve this problem owing to their exploratory nature, ease of formulation, flexibility in handling different types of decision variables, and inherent parallelism. Despite their many advantages, EAs can be computationally intensive as they may require a large number of forward simulations to solve an inverse problem such as source characterization. As the larger DDDAS project relies heavily on EA based methods [3][4] for solving source characterization and sensor placement problems, an end-to-end cyberinfrastructure is needed to couple the optimization engine to the simulation engine, launch the simulations seamlessly on the grid, and track the solution progress in real-time.

Thus the primary objective of this research is to facilitate an end-to-end solution to this simulation-optimization problem by utilizing the computational resources across the grid to effectively minimize the turnaround time with due consideration to flexibility and usability To achieve this, a communication layer is designed to facilitate interactions between the simulation and optimization components along with a middleware interface that not only abstracts the complexities of the underlying grid environment but also operates within its security constraints.

1.1 Related Work

This research focuses on developing a grid framework for an optimization driven simulation environment facilitating seamless interactions among the various components. Existing grid workflow systems such as CoG Kit [5] and Kepler [6] support pre-processing, post-processing, staging data/programs, and archival of results for a generic application on the grid. However, they do not provide custom solution to an application that requires frequent runtime interactions among its components (i.e., optimization and simulation components) at a finer granularity. They also require that the execution time of the core component (e.g., simulation) to be significantly large in order to amortize the overhead induced by the workflow system. In the WDS application, a single simulation instance can take anywhere from several milliseconds to several minutes depending on the network. If we need a system that can cater to any problem then it would not be feasible to use existing workflow systems (for smaller problems) without a significant performance penalty. To address this, a custom framework is developed in this research that can not only aggregate a large number of small computational tasks but also allows for persistent execution of these tasks during interactions with the optimization component in a batch environment. Existing workflow systems also do not provide support for real time monitoring of simulation-optimization runs from the perspective of a WDS application. Hence a real time visualization tool has been developed to inform the quantitative progress of the application to domain scientists. Moreover, this tool is intended to help enhance the understanding of the inner workings of the optimization methods and their convergence patterns, which can provide valuable feedback for optimization methods research.

The remainder of this paper is organized as follows. The next section provides a description of the architecture including enhancements in the optimization and simulation components, middleware design, and the visualization tool. Performance results are provided in the third section followed by conclusions and future work.

2 Architecture

The high level architecture of the preliminary cyberinfrastructure developed in this paper is shown in Fig 1. The optimization toolkit (which is a collection of optimization methods) interacts with the simulation component (parallel EPANET) through the middleware. The middleware also communicates with the grid resources for resource allocation and program execution.

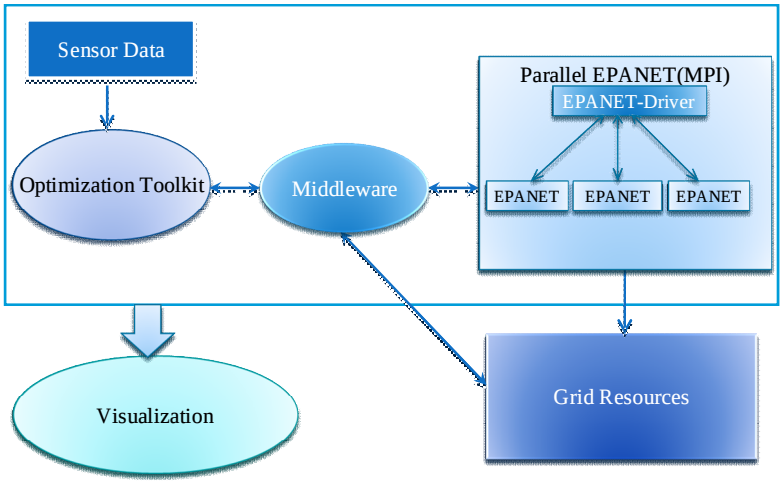

Fig. 1. Basic Architecture of the Cyberinfrastructure

Typically the user invokes a script that launches the optimization toolkit and the visualization engine from a client workstation. The optimization toolkit then receives observed data from the sensors (or reads a file that has been synthetically generated) and then calls the middleware interface to invoke the simulation engine. The middleware interface then surveys the available resources and launches the simulation engine on the available resources through batch submission scripts or interactive commands. The middleware also transmits the sets of decision variables (e.g., variables representing source characteristics) generated by the optimization engine to the simulation engine via files. The simulation engine calculates the fitness values corresponding to the sets of decision variables sent by the optimization engine. These are then transmitted back to the optimization and visualization engines via files. The optimization engine processes this data and sends new sets of decision variables back to the simulation engine for the next iteration of the algorithm. The simulation engine

maintains a persistent state until all the iterations are completed. The following subsections provide a brief description of the component developments involved in this cyberinfrastructure. Readers interested in additional details should refer to [7]. Subsequent discussions assume that the optimization engine uses EA based methods and the problem solved is source identification. However, the basic architecture is designed to handle any optimization method that relies on multiple simulation evaluations and any WDS simulation-optimization problem.

2.1 Simulation Model Enhancements

The simulation engine, EPANET [8] is an extended period hydraulic and water-quality simulation toolkit developed at EPA. It is originally developed for the Windows platform and provides a C language library with a well defined API [8]. The original EPANET was ported to Linux environments and customized to solve simulation-optimization optimization problems by building a "wrapper" around it. For testing purposes, limited amount of customization was built into the wrapper to solve source identification problems.

Prior to the development of the wrapper, several EA based optimization methods [3][4] were already developed for use in the project using diverse development platforms like Java and MATLAB respectively. Hence the primary design goal was to interoperate with existing optimization tools. The second design goal for the wrapper is to aggregate the simulations into a single parallel execution for multiple sets of source characteristics. This will amortize the startup costs as well as eliminate redundant computation. The wrapper is designed to receive multiple sets of contamination source parameters from an input file. During a single instantiation, the wrapper can perform multiple EPANET simulations for all sets of contamination source characteristics.

Parallelization

The parallel version of the wrapper is developed using MPI and referred to as '**pepanet**'. The middleware scripts are designed to invoke multiple 'pepanet' instantiations depending on resource availability. Within each MPI program, the master process reads the base EPANET input file (WDS network information, boundary conditions etc.) and an input file generated by the optimization toolkit that contains the source characteristics (i.e., decision variables). The master process then populates data structures for storing simulation parameters as well as the multiple sets of contamination source parameters via MPI calls. The contamination source parameter sets are then divided among all the processes equally ensuring static load balancing. Each process then simulates its assigned group of contamination sources successively. At the completion of assigned simulations, the master process collects results from all the member processes and writes it to an output file to be processed by the optimization toolkit.

Persistency

The evolutionary computing based optimization methods that are currently in use within the project exhibit the following behavior. The optimization method submits

some evaluations to be computed (generation), waits for the results and then generates the next set of evaluations that need to be computed. If the simulation program were to be separately invoked every generation, it needs to wait in a batch environment for acquiring the requisite computational resources.

But if the pepanet wrapper is made persistent, the program needs to wait in the queue just once when it is first started. Hence pepanet was enhanced to remain persistent across generations. In addition to amortizing the startup costs, the persistent wrapper significantly reduces the wait time in the job scheduling system. The persistent wrapper achieves this by eliminating some redundant computation across generations. One all evaluations are completed for a given generation (or evaluation set) the wrapper maintains a wait state by "polling periodically" for a sequence of input files whose filenames follow a pattern. The polling frequency can be tuned to improve performance (see section 3). This pattern for the input and output file names can be specified as command line arguments facilitating flexibility in placement of the files as well as standardization of the directory structure for easy archival.

2.2 Job submission Middleware

Consider the scenario when the optimization toolkit is running on a client workstation and the simulation code is running at a remote supercomputing center. Communication between the optimization and simulation programs is difficult due to the security restrictions placed at current supercomputing centers. The compute nodes on the supercomputers cannot be directly reached from an external network. The job submission interfaces also differ from site to site.

In light of these obstacles, a middleware framework based on Python has been developed to facilitate the interaction between the optimization and simulation components and to appropriately allocate resources. The middleware component utilizes public key cryptography to authenticate to the remote supercomputing center from the client site. The middleware then transfers the file generated by the optimization component to the simulation component on the remote site using available file transfer protocols. It then waits for the computations to be completed at the remote sites and then fetches the output file back to the client site. This process is repeated until the termination signal is received from the client side (in the event of solution convergence or reaching iteration limit). The middleware script also polls for resource availability on the remote sites to allocate appropriate number of processors to minimize queue wait time by effectively utilizing the backfill window of the resource scheduler. When more than one supercomputer site is involved, the middleware divides the simulations proportionally among the sites based on processor availability and processor speed. A simple static allocation protocol is currently employed.

2.3 Real-time Visualization

The current visualization toolkit is geared toward the source identification problem. It was developed with the following goals in mind:

- Visualize the water distribution system map and the locations where the optimization method is currently searching for contamination sources.

- Visualize how the search is progressing from one stage (generation) of the optimization algorithm to the next.
- Facilitate understanding of the convergence pattern of the optimization method.

The tool has been developed using Python, Tkinter and Gnuplot. Fig 2 shows a screenshot of the visualization tool after the optimization method found the contamination source for an example problem instance. It shows the map of the water distribution system marking the "true" source (as it is known in the hypothetical test case) and the estimated source found by the optimization method. It also provides a plot comparing the release history of the true source and the estimated source. A multi-threaded implementation enables the user to interact with the tool's graphical interface while the files are being processed in the backend.

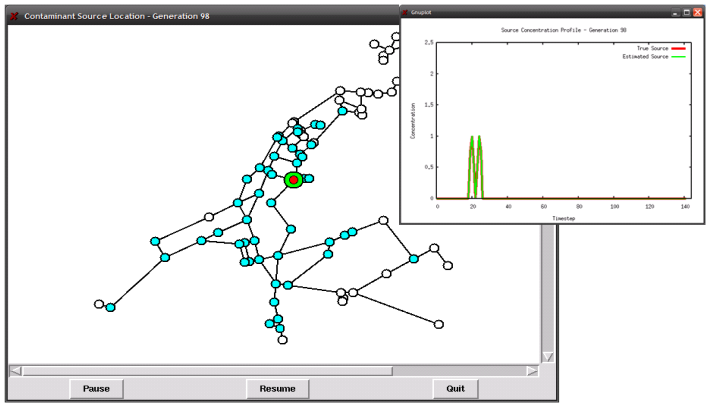

Fig. 2. Visualization Tool Interface showing the Water Distribution System Map and Concentration profile for the true (red) and estimated (green) sources.

3 Performance Results

Performance results are obtained for solving a test source identification problem involving a single source. The sensor data is synthetically generated using a hypothetical source. An evolutionary algorithm (EA) is used for solving this problem [3]. The following platforms are used for evaluating the performance of the cyberinfrastructure and for making design changes as appropriate.

- Neptune is an 11 node Opteron Cluster at NCSU. It has 22 2.2 GHz AMD Opteron(248) processors and a GigE Interconnect.
- The Teragrid Linux Cluster at NCSA has 887 IBM nodes of which 256 nodes have dual 1.3 GHz Intel Itanium 2 processors and 631 nodes have dual 1.5 GHz Intel Itanium 2 processors and are connected using a Myrinet interconnect.

Teragrid results are confined to simulations deployed on a single cluster but with the optimization component residing on a remote client site. Additional results including preliminary multi-cluster Teragrid results are available in [7]. The cyberinfrastructure has also been demonstrated on SURAgrid resources [9]. For timing purposes, the number of generations in the EA is fixed at 100 generations even though convergence is usually achieved for the test problem well before the 100th generation. The population size was varied from 600 to 6000 but the results in this paper are restricted to the larger population size.

Timers were placed within the main launch script, middleware scripts, optimization toolkit and the simulation engine to quantify the total time, optimization time, simulation time, and overhead due to file movements. Additional timers were placed within the simulation engine to break down the time spent in waiting or "wait time" (includes the optimization time and all overheads) and time spent in calculations. Preliminary tests revealed that the waiting time within the simulation code was exceedingly high when the optimization toolkit and root process of the wrapper (simulation component) are on different nodes of the cluster. When both are placed on the same compute node, wait time reduced by a factor of more than 15 to acceptable values. Additional investigation indicated that these were due to file system issues. Further optimization of the polling frequency within the simulation engine improved wait time by an additional factor of 2. Once these optimizations were performed, the wait time predominantly consisted of optimization calculations, since the overhead is relatively negligible. Fig 3 illustrates the parallel performance of the application after these optimizations on the Neptune cluster up to 16 processors. As expected, the computation time within the simulation engine (parallel component) scales nearly linearly while the overhead and the optimization time (serial component) remain more or less constant.

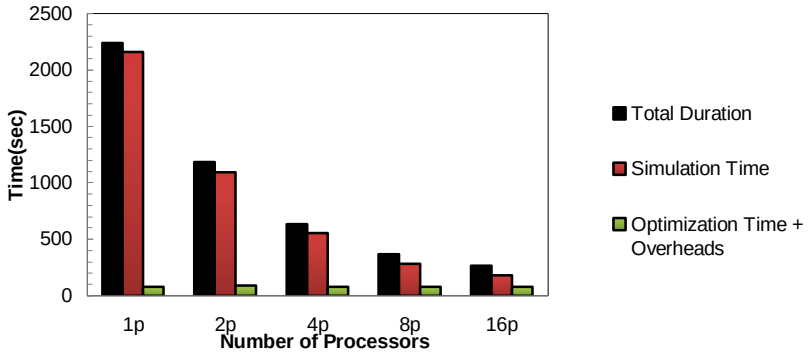

Fig. 3. Performance of the Simulation-Optimization application up to 16 processors on Neptune cluster

For timing studies on the Teragrid, the optimization engine was launched on Neptune and the simulations on the Teragrid cluster. Again, preliminary performance tests indicated that the wait times were significantly impacted by file system issues.

The performance improved by a factor of three when the application is moved from a NFS file system to a faster GPFS file system. Further improvement in wait time (about 15%) was achieved by consolidating the network communications within the middleware. Fig 4 shows the speedup behavior on the Teragrid cluster for up to 16 processors. While the computation time speeds up nearly linearly the wait time is much higher when compared to the timings of Neptune (Fig 3). Additional breakdown on the wait time indicated that 14% was spent in optimization and the remaining 86% in overheads including file transfer costs. The increased file transfer time is not unexpected as the transfers occur over a shared network between the geographically distributed optimization and simulation components. Comparison of simulation times with Neptune indicates that the Teragrid processors are also about three times slower.

It is noted that the WDS problem solved is relatively small (about 20 seconds (Neptune) or 1 minute (TeraGrid) per 6000 simulations using 1 processor) thus making the overheads appear relatively high. As the overhead will remain approximately constant with increased problem sizes we expect the results to improve considerably for larger problems. Furthermore, several enhancements are planned to minimize overheads (see next section).

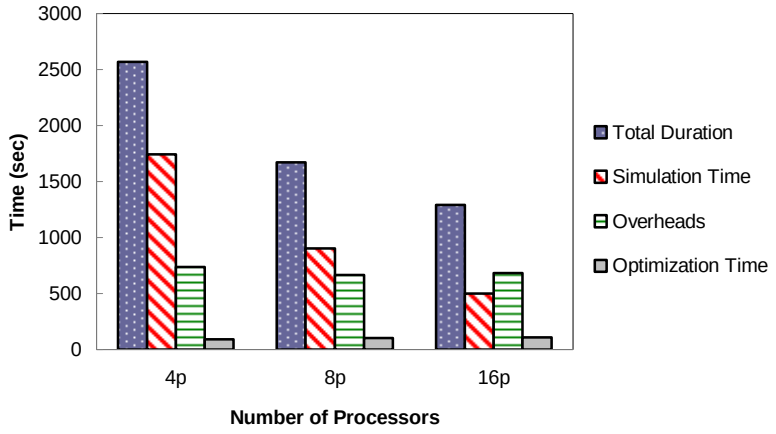

Fig. 4. Performance of the Simulation-Optimization application up to 16 processors on NCSA Teragrid Linux Cluster.

4 Conclusions and Future Work

An end-to-end solution for solving WDS contamination source characterization problems in grid environments has been developed. This included coarse-grained parallelization of simulation module, middleware for seamless grid deployment and a visualization tool for real-time monitoring of application's progress. Various performance optimizations such as improving processor placements, minimizing file system overheads, eliminating redundant computations, amortizing queue wait times,

and multi-threading visualization were carried out to improve turnaround time. Even with these optimizations, the file movement overheads were significant when the client and server sites were geographically distributed as in the case of Teragrid. Several future improvements are planned including optimization algorithm changes that can allow for overlapping file movements and optimization calculations with simulation calculations, localizing optimization calculations on remote sites by partitioning techniques, and minimizing file transfer overhead using grid communication libraries.

Acknowledgments. This work is supported by National Science Foundation (NSF) under Grant Nos. CMS-0540316, ANI-0540076, CMS-0540289, CMS-0540177, and NMI-0330545. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

1. Darema, F. Introduction to the ICCS 2006 Workshop on Dynamic data driven applications systems. *Lecture Notes in Computer Science 3993*, pages 375-383, 2006.
2. Mahinthakumar, G., G. von Laszewski, S. Ranjithan, E. D. Brill, J. Uber, K. W. Harrison, S. Sreepathi, and E. M. Zechman, An Adaptive Cyberinfrastructure for Threat Management in Urban Water Distribution Systems, *Lecture Notes in Computer Science, Springer-Verlag*, pp. 401-408, 2006.(International Conference on Computational Science (3) 2006: 401-408)
3. Zechman, E. M. and S. Ranjithan, "Evolutionary Computation-based Methods for Characterizing Contaminant Sources in a Water Distribution System," *Journal of Water Resources Planning and Management*, (submitted)
4. Liu, L., E. M. Zechman, E. D. Brill, Jr., G. Mahinthakumar, S. Ranjithan, and J. Uber "Adaptive Contamination Source Identification in Water Distribution Systems Using an Evolutionary Algorithm-based Dynamic Optimization Procedure," *Water Distribution Systems Analysis Symposium*, Cincinnatti, OH, August 2006
5. von Laszewski, G. and Hategan, M., Java CoG Kit Workflow Concepts, *Journal of Grid Computing*, Jan. 2006. http://www.mcs.anl.gov/~gregor/papers/vonLaszewski-workflow-jgc.pdf.
6. Kepler: An Extensible System for Design and Execution of Scientific Workflows, I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, S. Mock, system demonstration, 16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'04), 21-23 June 2004, Santorini Island, Greece.
7. Sreepathi, S., Cyberinfrastructure for Contamination Source Characterization in Water Distribution Systems, Master's Thesis, North Carolina State University, December 2006.
8. Sreepathi, S., Application Driven Design for a Large-Scale, Multi-Purpose Grid Infrastructure: Simulation-Optimization for Threat Management in Urban Water Systems, Demo, Fall 2006 Internet2 Meeting, December 2006.
9. Rossman, L.A.. The EPANET programmer's toolkit. In Proceedings of Water Resources Planning and Management Division Annual Specialty Conference, ASCE, Tempe, AZ, 1999.