Computational Complexity and Distributed Execution in Water Quality Management

Maria Chtepen[1], Filip Claeys[2], Bart Dhoedt[1], Peter A. Vanrolleghem[2], and Piet Demeester[1]

[1] Department of Information Technology (INTEC), Ghent University, Sint-Pietersnieuwstraat 41, Ghent, Belgium
{maria.chtepen, bart.dhoedt, piet.demeester}@intec.ugent.be
[2] Department of Applied Mathematics, Biometrics and Process Control (BIOMATH), Ghent University, Coupure Links 653, Ghent, Belgium
{filip.claeys, peter.vanrolleghem}@biomath.ugent.be

Abstract. Modeling is considered an inherent part of design, operation and optimization of Water Quality Systems. Models are used for running so-called Virtual Experiments, such as Simulation and Uncertainty Analysis. Since the complexity of VE's is constantly increasing, there is a need for techniques that allow for timely execution of complex VE's on real-world models. This paper touches several of these: use of efficient solvers, model coupling based on the Open Modeling Interface (OpenMI), and distributed execution. Focus however is predominantly on the software system that was developed to allow for the latter. This system has thus far been applied to the WEST modeling and simulation software package. It is a stable, efficient and platform-independent tool for transparent distributed execution of workload on work nodes within an organization. It can be regarded as conceptually similar to the Grid Computing paradigm, albeit its specific nature currently makes it more geared towards Water Quality Management systems.

1 Introduction

The importance of Water Quality Management has drastically increased during the last decades as a consequence of growing environmental awareness. Compared to well-defined systems (e.g. electrical and mechanical) that can entirely be described by classical laws of physics, the behavior of ill-defined water systems is much more difficult to predict due to only partial availability of generally applicable laws. Most frequently, one has to start with approximate parameterized equations and determine the exact values of the parameters through the evaluation of a large number of measurements later on. As a result of the complexity of water systems, modeling is regarded an inherent part of design, operation and optimization. The models used in practice are typically based on a combination of general physical laws (typically laws of preservation of energy and mass) and purely empirical laws. These models form an excellent tool to summarize and increase the understanding of complex interactions in biological systems. More quantitatively, they can be used to predict the dynamic response of the system to various disturbances.

Modeling and Virtual Experimentation in the domain of Water Quality Management is computationally complex. In Section 2 several facets of this complexity will be discussed, in conjunction with the way these are typically tackled. One issue which is of special importance is the distributed execution of compound Virtual Experiments (VE's). In Section 3 this issue will therefore be discussed separately.

In order to illustrate the various sources of computational complexity and the way these are typically dealt with, the WEST [1, 2] software system will be described. WEST[1] is a versatile yet powerful tool, which thus far has mainly been applied to wastewater treatment processes. It consists of clearly separated Modeling and Experimentation Environments. Both environments are self-contained and consist of a graphical front-end, computational back-end and control logic. The Modeling Environment allows for the creation of executable models on the basis of high-level model descriptions, through the application of model compiler techniques. These executable models are subsequently used as a basis for the creation of VE's in the Experimentation Environment. The reason why the term "VE" was adopted in this case is related to the fact that WEST goes beyond plain simulation. In fact, the types of VE's that are currently supported are: Simulation, Steady-state Analysis, Optimization, Confidence Analysis, Scenario Analysis, Sensitivity Analysis and Uncertainty Analysis.

2 Computational Complexity of Water Quality Systems

In various scientific disciplines models are typically directly coded in general purpose programming languages such as FORTRAN. Also, in many cases models and their solution methods are coded into the same software in an indistinguishable manner. However, in water system modeling, this approach is undesirable. Software tools such as WEST therefore provide a high-level modeling language that allows for the specification of models at a higher level of abstraction. The MSL [1] language currently adopted by WEST is declarative, object-oriented and offers various high-level constructs and type-safety mechanisms. In order to further process this high-level model specification, a model compiler is used. The model compilation process consists of two stages. During the first stage, also named *flattening*, the model's decomposition and inheritance hierarchies are resolved. The second stage then consists of optimization of the flattened model and generation of executable code (typically C or FORTRAN). Both stages are potentially computationally complex and consist of mainly symbolic manipulations. During the first stage sorting and expansion of equations, as well as causality assignment is to be applied. During the second stage, the various forms of model optimization (such as expression simplification, aliases substitution, lifting of equations, and index reduction) also require a substantial number of symbolic

[1] World-wide Engine for Simulation and Training

computations. Total model building times range from a number of seconds to several hours on a high-end PC.

In systems such as WEST, an executable model generated by a model compiler is subsequently loaded into a highly configurable execution environment. Such an execution environment is capable of simulating the model over time, as well as running various other types of VE's. VE's such as Simulation and Steady-state Analysis are said to be *atomic* since they are not hierarchically composed of other VE's. On the other hand, *compound* VE's such as Optimization, Confidence Analysis, Scenario Analysis, Sensitivity Analysis and Uncertainty Analysis all are based on the repeated execution of simulations for different parameter sets. Optimal Experimental Design (OED) is yet another type of compound VE, which is currently under development for WEST. It basically consists of an Optimization experiment that contains an embedded Sensitivity Analysis experiment, which on turn contains an embedded Simulation experiment.

Compound VE's quickly become computationally complex. There is therefore a need for powerful numerical solvers, especially at the lowest (i.e. simulation) level. Tools such as WEST consequently offer a wide variety of fixed- and variable-step ODE (Ordinary Differential Equation) solvers, as well as highly optimal stiff system solvers. For the purpose of optimization, classical analytical techniques as well as novel evolutionary algorithms are made available. Choosing the numerical solver that best matches the situation at hand is quite important, since the effect of choosing a inappropriate solver on the total execution time can be dramatic. Typical execution times for compound VE's range from a number of seconds to several days or weeks on a high-end PC.

The domain of Water Quality Management is divided into a number of sub-domains that each focus on the various stages of the water cycle: river basins, treatment plants, sewer systems, ... For each of these sub-domains, various specialized software systems have made their way onto the market. In order to perform integrated Water Quality Management, one is therefore confronted with two options. On the one hand, one could build a comprehensive model of the entire water system and use one specific tool for its execution. This approach has proven to be problematic, which is why the principle of *divide and conquer* is currently considered to be more practical. In this approach specialized tools are used to run models pertaining to the various individual stages of the water cycle. These tools are then coupled in order for data to flow from one model implementation to the next. Evidently, this can only be accomplished in case all tools comply to the same interface specification. In order to alleviate the need for a standardized model coupling interface, the OpenMI interface was recently developed in the scope of the EU's HarmonIT project [3].

3 Distributed Execution Aspects

3.1 Introduction

Zooming in on compound VE's, one comes to the conclusion that these can further be subdivided into two broad categories. One the one hand are those that rely on the execution of an *unordered* set of sub-experiments. Indeed, compound VE's such as Scenario Analysis and Sensitivity Analysis require a number of simulations to be run for different parameter sets. Since each of these parameter sets is known beforehand, all simulations are independent and therefore do not have to be run in any particular order. However, examples of compound VE's that rely on an *ordered* set of sub-experiments can also be given. In the case of optimization on the basis of classical analytical algorithms for instance, each parameter set depends on the outcome (i.e. the objective value) of previous simulation runs.

It seems evident that the total execution time of unordered compound VE's can be greatly improved through the concurrent execution of unrelated sub-experiments on different work nodes. A plethora of distributed execution environments has been worked out in the past - recently culminating in the Grid Computing paradigm [5] - that would seem appropriate in the scope of unordered compound VE's. In this respect a new framework developed for distributed execution of VE's can be seen as a light-weight Grid system dedicated to the Water Quality Management application area. This framework was named WDVE [4] and was built on top of technologies such as C++, STL, XML, and SOAP. WDVE was initially implemented for the WEST software system, however thanks to generic design it can be used for execution of any computationally intensive task consisting of an unordered set of jobs.

Design and implementation of the WDVE system started from a number of important requirements. These requirements were established taking into account the typical needs of an organization in the area of Water Quality Management.

Financial. The first demand imposed on the new environment states that its deployment shouldn't require any additional investments in terms of hardware and staffing. This implies that deployment on any existing - possibly heterogeneous - pool of work nodes should be possible. It also implies that initial installation and further maintenance should be straightforward, so that no additional personnel is to be hired.

Functional. From a functional point of view, the system should evidently be efficient and robust, i.e. it should deliver fast execution of jobs and should account for machine crashes and network drop-outs. Because of its deployment in environments with a largely non-computer science staffing, it should also be transparent, user-friendly and flexible. In order to be able to seamlessly integrate the system with applications, it was to be conceived as a library that applications can link against, rather than as a stand-alone system. Lastly, application-independence was a major goal. The fact that the system was initially developed for WEST does not imply that its deployment in the scope of other applications should be hampered.

Technical. Striving for the code to be maintainable over longer periods, a strict demand for elegant design and coherent implementation was imposed. Also, full OS portability, interoperability and limited dependence on third-party tools and libraries was required.

3.2 Architecture

WDVE consists of two major modules: Master and Slave. The Master's main functionalities are to receive VE's (Jobs) from the Application front-end, to store Jobs until they are processed, to match them against available work nodes and to initiate execution. Slaves run (possibly concurrent) Jobs on their computational resources and collect the outputs requested by the user.

Master as well as Slave were further decomposed into a number of sub-modules (see Fig. 1), each implementing a particular, well-defined functionality. The provided module arrangement was chosen in order to ensure appropriate interaction between a Master and its Slaves, and to ease possible future extensions of WDVE.

Dispatcher. The Dispatcher is located at the heart of the Master. It acts as an interface between WDVE and the Application and manages the other modules of the system. The Dispatcher's main task is to administer the Jobs from the moment they are submitted by the Application until execution is completed and results are available to the user.

Registry. Before the Dispatcher can initiate remote Job executions, Slaves must be registered with the system. The goal of the Registry is to allow for work nodes to be statically or dynamically registered as Slaves. A Registry module is part of most distributed environments in one way or another. In most cases, it is implemented as a separate process running on a dedicated machine. However, in order to reduce complexity, the WDVE Registry was integrated into the Master.

Selector. At present the Selector is a rather simple sub-module of the Master, since it only implements the Round Robin algorithm for selecting a Slave for the next Job to be executed. This module is intended to be extended with more intelligent algorithms in the future.

Checker. The consistency of the state of the system is ensured by the Checker background thread, which constantly monitors the status of various entities and triggers appropriate actions.

Acceptor. This module is at the heart of each Slave. It evaluates requests from the Master for the execution of Jobs. In case a Job is accepted, a new instance of the Application back-end is created and Job execution is started.

For the transfer of input, output and status data between Master and Slaves, the gSOAP implementation of the SOAP protocol was used. Reasons for electing SOAP from the wide range of the available middleware technologies are performance, portability and the open source nature of gSOAP. In order for the specifics of the middleware not to be visible by the rest of the code, an abstraction layer was added on top of the gSOAP wrapper classes as shown in Fig. 2. Transparency was obtained by deriving Proxy classes from the same abstract C++ interfaces as those the actual remote classes are derived from.

An important architectural question is how to represent Jobs in their various incarnations, starting from the point when a Job is submitted to the system and ending with the completed Job and its output data. It was decided to consistently use XML to this purpose. In the context of WDVE, Jobs descriptions consist of Executor-dependent and Executor-independent information. The Executor-

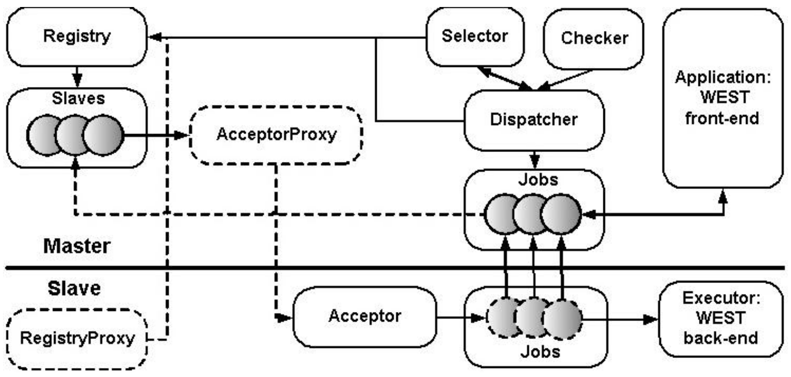

Fig. 1. WDVE: Architecture

independent information consists of a set of Resources including various types of input and output files. The Executor-dependent information contains various other pieces of information that are relevant to the specific Executor that is to execute the Job.

3.3 Test results

Tests were run comparing the efficiency of WDVE with monolithic solutions. The results that were found are in line with the typical performance behavior of other distributed systems. In Fig. 3 are some of the results that were obtained from a test using three identical Slaves that execute Jobs assigned by a Master that has a total of 1 to 100 Jobs to execute. From Fig. 3 (left) one easily concludes that in case the Jobs are short (in this case: artificially short), there is nothing to be gained from distributing Jobs over multiple Slaves. Actually, the total execution time of distributed execution is worse than in case the Jobs are sequentially executed on one machine. However, in case of Jobs of realistic complexity, one does get a significant reduction of the total execution time (Fig. 3, right). As the number of Slaves increases, this reduction becomes more pronounced, up to a point however where no further improvement is possible. The speed-up factors obtained in practice are also always lower than those that one would theoretically expect. Evidently, this is due to the overhead inflicted by the WDVE software itself and the transmission of data over the network.

3.4 WDVE and Grid Computing

There are some obvious similarities between the Grid Computing paradigm, which is recently receiving widespread attention, and WDVE. These similarities include the high-level functionality (i.e. the distribution of a number of tasks over a number of work nodes), the importance of the fail-safety aspect, and

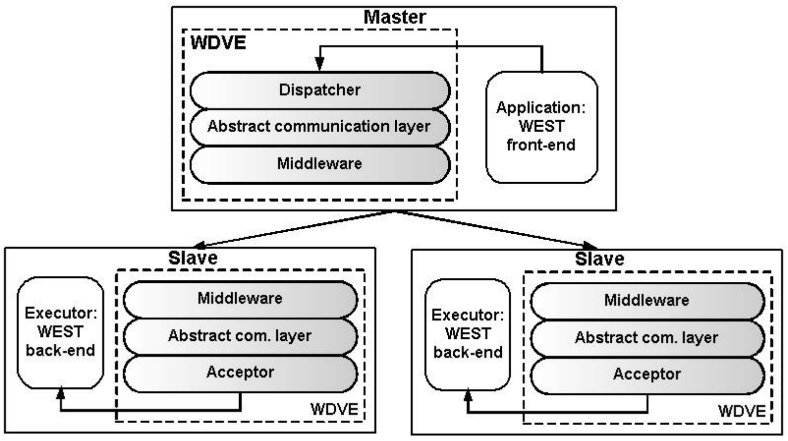

Fig. 2. WDVE: Layers

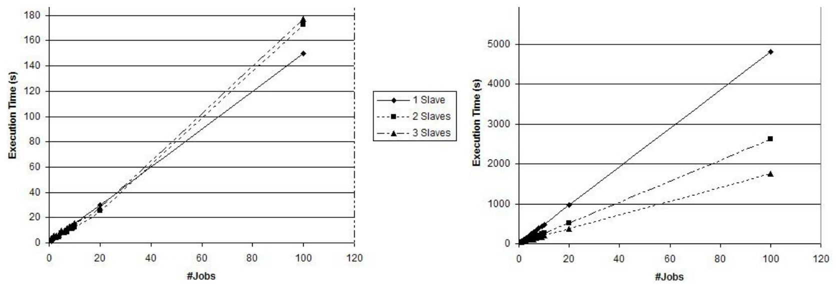

Fig. 3. WDVE: Test results for short Jobs (left) and long Jobs (right)

the fact that work nodes should be able to dynamically register and unregister. However, there are also a number of important conceptual differences between Grid Computing and WDVE. These differences include the following:
Scope. WDVE software was designed for simplicity and efficiency. Its attempts to deliver go less far than the full Grid functionality. WDVE's code base is less than 50 kLOC, which contributes to the manageability and maintainability of the system.
Consistency. WDVE was developed in a consistent manner, using as few third-party components as possible, which again increases the manageability of the software. On the other hand, the Grid software is built on top of a large number of external tools.
Installation. WDVE installs in a matter of minutes. To make use of the system, the Master package should be installed on all machines utilized for Jobs submission and the Slave package should be installed on all work nodes. To speed up this process an automatic installation procedure for both modules is available.

Intelligence. The intelligence claimed by WDVE when distributing Jobs is quite limited at this point, but work is ongoing on including job dependencies and other associated scheduling issues. In this respect Grid software goes much further.
Authentication. WDVE is mainly intended for deployment within one organization. Issues such as security and confidentiality of data have therefore currently not been taken into account.
Portability. Grid software is intended to be portable, however portability proves to be a difficult issue for such a complex system. It was demonstrated that WDVE software currently installs and operates on platforms such as Windows NT, Windows 2000, Windows XP and Linux.

4 Conclusion

There are several approaches to dealing with complexity in Water Quality Systems: optimization of high-level model descriptions, adopting efficient solvers, model coupling on the basis of standard interfaces, and distributed execution of VE's on a pool of work nodes. All of the methods mentioned can be used separately or combined, however since the complexity of VE's and their underlying models is constantly increasing, performance of monolithic solutions is rapidly becoming insufficient. Hence the importance of the WDVE distributed execution environment, which provides transparent execution of VE's on a potentially heterogeneous pool of work nodes. It was initially designed for use in the scope of the WEST software system, but the framework is sufficiently general to be applicable to other types of problems, not necessarily related to Water Quality Management. In many ways its concept can be regarded as similar to the Grid Computing paradigm, although it is mostly intended for deployment within a single organization and consequently does not go as far as Grid with regard to security and intelligence. However, as a result of these simplifications WDVE succeeds in delivering a low cost, stable, expandable, performant, platform-independent and easy to use platform.

References

1. H. Vanhooren, J. Meirlaen, Y. Amerlinck, F. Claeys, H. Vangheluwe, and P. Vanrolleghem. WEST: Modelling Biological Wastewater Treatment. *Journal of Hydroinformatics, IWA Publishing*, 5(1), 2003.
2. HEMMIS N.V. WEST website. *http://www.hemmis.com.*
3. HarmonIT. OpenMI website. *http://www.openmi.org.*
4. F. Claeys, M. Chtepen, L. Benedetti, B. Dhoedt, and P.A. Vanrolleghem. Distributed Virtual Experiments in Water Quality Management. In *6th International Symposium on Systems Analysis and Integrated Assessment in Water Management (WATERMATEX)*, pages 485–494, Beijing, China, November 3–5 2004. International Water Association.
5. EGEE. Enabling Grids for E-science in Europe (EGEE) website. *http://egee-intranet.web.cern.ch/egee-intranet/gateway.html.*