Towards Transparent Distributed Execution in the Tornado Framework

Filip H.A. Claeys[1], Maria Chtepen[2], Lorenzo Benedetti[1], Webbey De Keyser[1], Peter Fritzson[3], Peter A. Vanrolleghem[1,4]

[1] Department of Applied Mathematics, Biometrics and Process Control (BIOMATH), Ghent University, Coupure Links 653, B-9000 Ghent, Belgium
fc@biomath.ugent.be

[2] Department of Information Technology (INTEC), Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
mchtepen@intec.ugent.be

[3] Programming Environments Laboratory (PELAB), Linköping University, SE-581 83 Linköping, Sweden
petfr@ida.liu.se

[4] model*EAU*, Département de génie civil, Université Laval, Pavillon Pouliot, Québec, G1K 7P4, QC, Canada
peter@modelEAU.org

Abstract. Tornado is a new advanced kernel for modelling and virtual experimentation (*i.e.*, any evaluation of a model) in the water quality domain. Although primarily intended for use within this particular domain, the kernel is generic in nature and has a *plethora* of generally applicable features. Tornado often deals with elaborate models and many of its virtual experiment types are computationally intensive. In order to alleviate the computational burden, there is a strong need for distributed execution. However, since Tornado has a heterogeneous user community consisting of both expert and non-expert users, the distributed execution process should preferably be as transparent as possible. This article focuses on the initial steps that were taken along the road to transparent distributed execution. Main achievement so far is the ability to perform semi-automated distributed execution of workload on the Typhoon cluster and LCG-2 grid infrastructures. Our approach is based on the generation of generic job descriptions and has been shown to offer sufficient transparancy for non-expert users in the scope of a Monte Carlo simulation project that was run on a 16-node Typhoon / 40-node LCG-2 setup.

1 Introduction

In water quality research, the biological and/or chemical quality of water in rivers, sewers and wastewater treatment plants (WWTP) is studied. Research in this domain is facilitated by a number of models that have received a formal or *de facto* standardization status. Most notable are River Water Quality Model No.1 (RWQM1) [1] and the Activated Sludge Model (ASM) series [2].

Water quality models typically consist of large sets of non-linear Ordinary Differential Equations (ODE) and/or Differential-Algebraic Equations (DAE). These equations

are mostly well-behaved, although discontinuities occur regularly. The complexity of water quality models is therefore not in the nature of the equations, but in the sheer number. In WWTP, smaller models such as the well-known Benchmark Simulation Model (BSM) [3] consist of approximately 150 derived variables. Larger systems have up to 1,000 derived variables and over 10,000 (partly coupled) parameters. On a typical workstation, a simulation run usually lasts minutes to hours.

Virtual experimentation with water quality models is a complex and computationally intensive process, which frequently requires 100's or 1000's of simulation runs. A continuous need for software tools that offer extended functionality and improved performance therefore exists. An example of a software tool that was recently developed is Tornado [4]. It manages to substantially improve performance with respect to its predecessors, and offers a broad range of virtual experiment types to solve various frequently occurring problems. Not withstanding the inherent efficiency of Tornado, situations still abound where complex problems cannot be solved on one single workstation in a reasonable amount of time. In order to alleviate the computational burden, distributed execution is a requirement. Fortunately, many complex problems in Tornado are composed of loosely coupled sub-problems. Coarse-grained gridification is therefore relatively easy to accomplish.

A large variety of distributed execution environments is available nowadays, ranging from elaborate inter-organizational grid infrastructures (*e.g.* LCG-2, Condor-G, UNICORE, ...[1]) to more localized cluster solutions (*e.g.* Torque, (Open)PBS, ...[2]). Solutions based on the migration of processes at the operating system level (*e.g.* (Open)-MOSIX[3]) and solutions that are specific for a particular application (*cf.* the MATLAB distributed execution toolbox) also exist. Although the number of available solutions is large, only a few are truly mature, and transparency and ease of use are often limited. In fact, the world of distributed execution is still highly dynamic and all solutions have their own particular pros and cons. For instance, MOSIX offers a high degree of transparency, but is restricted to one particular platform (*i.e.*, linux) and is not well-suited for the distribution of data-intensive tasks. On the other hand, the disadvantage of many grid and cluster solutions is that they are often heavily command-line oriented (and hence unsuitable for many types of users), highly convoluted, and again not portable across platforms.

The Tornado user community is very diverse. It consists of a small group of experts (typically found in academia) who have the knowledge to construct atomic models of unit processes, next to coupled models of plant layouts and virtual experiments. A second, larger group is mainly found among engineering and consultancy companies. Members of this group will also create coupled models and set up virtual experiments, however they will not construct atomic models and will merely re-use those that are provided by members of the first group. The largest user group is composed of plant operators, who will mainly only run ready-made virtual experiments.

Since, on the one hand, all distributed execution environments have their own pros and cons, and on the other hand, it is very difficult at this stage to predict which solutions

[1] http://gridcafe.web.cern.ch/gridcafe
[2] http://www.clusterresources.com
[3] http://www.mosix.org

will prevail in the end, it would not be wise to graft distributed execution in Tornado to one particular distributed execution environment. A more advisable approach is to introduce a layer of abstraction that hides the details of the actual environment that is being used. Moreover, given the heterogeneity of the Tornado user community, no assumptions can be made about the level of expertise of its users. For instance, one cannot expect users to manually write shell scripts and embed them in job descriptions that are subsequently submitted using a grid infrastructure's command-line tools. The abstraction layer should therefore allow for seamless integration of the distributed execution infrastructure into the application. Ultimately, users should not see the difference between jobs executed locally and remotely.

At this moment, Tornado has a partial abstraction layer and a generic job description format that allows for two distributed execution environments to be utilized: LCG-2 and Typhoon (formerly known as WDVE) [5]. Unfortunately, only semi-automatic processing is possible at this point, but this type of processing does constitute an important first step towards transparent distributed execution.

The sequel of this article is structured as follows: Sections 2 and 3 respectively introduce the Tornado kernel, and some general aspects of distributed execution in the scope of Tornado. Sections 4 and 5 respectively discuss the integration of Typhoon and LCG-2 with Tornado. Subsequently, a project that illustrates the benefit of distributed execution in Tornado is presented in Section 6. Finally, some conclusions are formulated.

2 The Tornado Kernel

The Tornado kernel for modelling and virtual experimentation attempts to offer a compromise between the computational efficiency of custom hard-coded (typically FORTRAN or C) model implementations and the flexibility of less computationally efficient generic tools such as MATLAB. In Tornado, hierarchical models are specified in high-level, declarative, object-oriented modelling languages such as MSL [6] and Modelica [7]. From these high-level specifications, efficient executable code is generated by a model compiler. Using the executable models generated by the model compiler, Tornado allows for running a variety of so-called *virtual experiments*. Virtual experiments are the virtual-world counterpart of real-world experiments, similar to the way models relate to real-world systems. A highly simplified conceptual diagram of Tornado is shown in Figure 1 (left).

The most common virtual experiment type in Tornado is dynamic simulation (ExpSimul). Other experiment types include optimization (ExpOptim), confidence information analysis (ExpCI), sensitivity analysis (ExpSens), scenario analysis (ExpScen), Monte Carlo (*i.e.*, LHS - Latin Hypercube Sampling) analysis (ExpMC), and steady-state analysis (ExpSS). Virtual experiments are hierarchical in nature, in the sense that one experiment can be composed of a number of other experiments. We therefore distinguish between *atomic* and *compound* experiments. Atomic experiments are not made up of constituents and therefore cannot be decomposed (examples are ExpSimul and ExpSS). Compound experiments on the other hand rely on one or more (atomic or compound) sub-experiments (examples are ExpOptim, ExpCI, ExpSens, ExpScen and

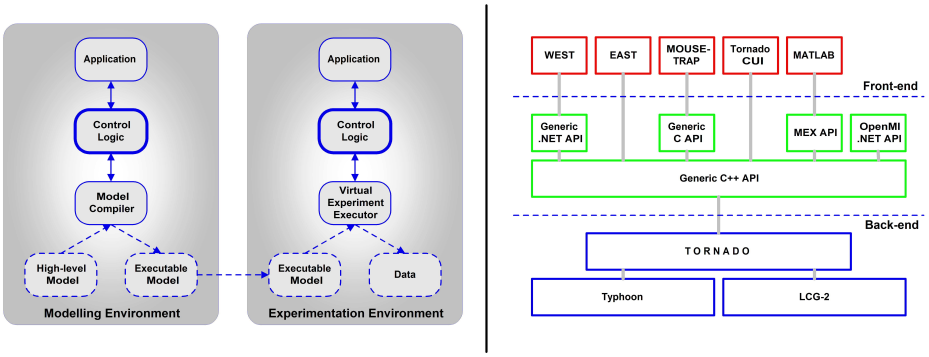

Fig. 1. Conceptual Diagram and Tornado-based Interfaces and Applications

ExpMC). As is illustrated in Figure 2, the afore-mentioned compound experiments are 2-level experiments. Tornado currently however also has two 3-level experiment types (ExpScenOptim & ExpMCOptim: repeatedly run the same ExpOptim experiment starting from different initial values). In the scope of this article, the exact semantics of the types of experiments mentioned so far are not relevant and will therefore not be further explained. One can however refer to [4] for more information.

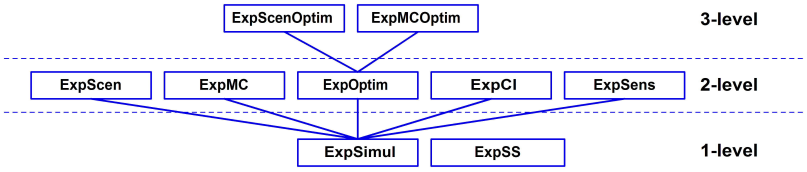

Fig. 2. Tornado Experiment Type Hierarchy

The Tornado kernel relies on a flexible input provider and output acceptor mechanism to deal with I/O for virtual experiments. In order to allow for the kernel to be deployed in a diverse array of applications, it has been equipped with multiple interfaces. Next to its native C++ interface, Tornado currently also has a C, .NET and MATLAB MEX interface (*cf.* Figure 1, right).

Tornado is portable across platforms and was designed according to the three-tier principle. Most persistent representations of information types are XML-based. The grammar of these representations is expressed in XSD (XML Schema Definition) format and mimics very closely the internal representation of the respective types of information. An interesting feature of Tornado is that it allows for dynamic loading of numerical solvers for tasks such as integration, optimization and Latin Hypercube Sampling. In order to support this principle, a generalized framework has been set up [8].

Several applications (graphical and other) can be built on top of Tornado. Examples include the next generation of the WEST® [6] commercial modelling and simulation tool for WWTP's, its research-oriented counterpart named EAST and DHI's MOUSE-TRAP. However, the most direct way of using the kernel is through the Tornado CUI

(Command-line User Interface) suite, which is a comprehensive set of tools that is included with the kernel distribution. Whereas full-fledged graphical applications such as WEST® are used by all afore-mentioned types of users, the Tornado CUI suite focuses on experts only. The most commonly used CUI tools are `tmsl` (compiles a high-level MSL model to executable model code), `tbuild` (compiles and links executable model code to a dynamically loadable binary object), `tcreate` (creates an empty XML description of a virtual experiment), and `texec` (executes virtual experiments).

3 Tornado and Distributed Execution

When faced with the computational complexity of modelling and virtual experimentation kernels such as Tornado, there are in general two approaches that can be followed with regard to gridification: either fine-grained gridification at the level of models (*i.e.*, distributed simulation), or coarse-grained gridification at the level of virtual experiments. The application of the first approach in the scope of Modelica is discussed in [9]. It can intuitively be understood that fine-grained gridification is a fairly complex and convoluted approach that will only yield good results in case of large models in which loosely coupled components can be identified. Since Tornado models usually do not contain large, loosely coupled components, fine-grained gridification has not been withheld as an option. On the other hand, given the inherent hierarchical nature of compound virtual experiments, coarse-gridification is relatively easily applicable.

When investigating the dependency relationships between the components of compound virtual experiments in Tornado, two classes can be identified. The first class of experiments (ExpScen, ExpMC, ExpSens) is based on the execution of a number of simulations that are independent of each other. In the other class of experiments (ExpOptim, ExpCI) a simulation can only be run after a number of other simulations have been executed, hence leading to a sub-experiment dependency relationship. The ExpScenOptim and ExpMCOptim experiments are special in the sense that they belong to the first class when optimizations are considered as the smallest undividable unit of work. In case however simulations are considered to be smallest undividable unit, they belong to the second class. As an example, the flow of execution of the ExpOptim and ExpScen experiment types is represented in Figure 3. The meaning of the tasks mentioned in this figure is as follows:

- *Generate value vector*: Create a vector composed of initial values for a number of identified objects (typically model parameters and/or initial conditions)
- *Set value vector*: Apply previously generated initial values to their corresponding objects
- *Run simulation*: Run a simulation using the newly applied initial values
- *Compute criteria*: Compute a number of criteria (objective values) from the simulated trajectories
- *Store simulated trajectories*: Store the simulated trajectories of a number of identified model quantities for future reference

As a first step towards distributed execution in Tornado, a generic job specification format was chosen, taking the following considerations into account: simplicity, maintainability, extensibility, application-independence, platform-independence, support for

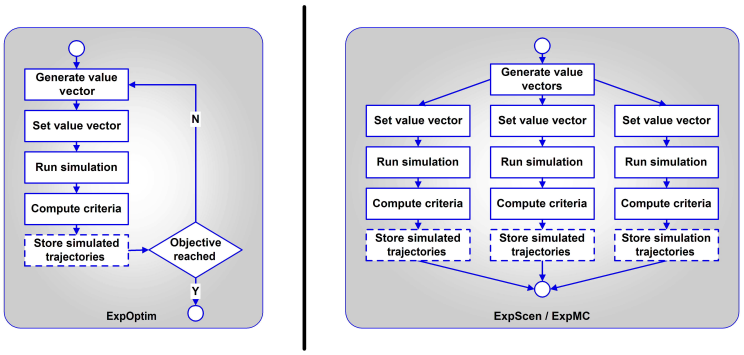

Fig. 3. Flow of Execution of the ExpOptim and ExpScen Experiment Types

the notion of sets of related jobs. The format that was withheld is the XML-based Typhoon format, which is loosely based on the Global Grid Forum's JSDL format (Job Submission Description Language[4]). Actually, JDSL was still under development at the time the Typhoon format was defined, however it recently has been put forward as a recommended standard.

A graphical representation of the XSD description of the Typhoon job specification is shown in Figure 4. From the figure follows that a "Jobs" entity consists of a number of Properties (basically attribute-value pairs) and a set of "Job" entities. These "Job" entities are further composed of sets of input and output resources, and a structure that describes the application that the Job is related to. Input resources are items (typically files) that need to be transferred to the remote execution node in order to be able to execute the job. Output resources are items generated during job execution that need to be transferred from the remote execution node back to the user's workstation. The structure of the "App" (Application) entity is application-dependent. In the case of Tornado, it consists of a reference to the input resource that should be considered as the start-up Tornado XML experiment description, and a list of initial values that are to be applied to the experiment loaded.

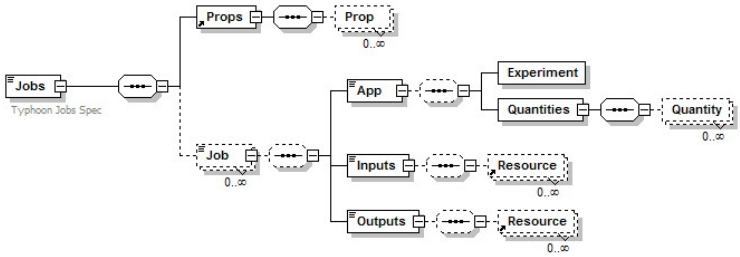

Fig. 4. Grammar of Typhoon Job Description

As a second step towards distributed execution, the implementations of relevant compound experiments (*i.e.*, experiments that rely on independent sub-experiments)

[4] https://forge.gridforum.org/projects/jsdl-wg

were modified in order to support 3 modes of operation. Mode 1 executes the experiment in a non-distributed fashion: first pre-processing is performed, followed by sequential execution of sub-experiments, and finally post-processing. Mode 2 also first performs pre-processing, but then simply generates job descriptions for each sub-experiment. Mode 3 only performs post-processing on sub-experiment data that was generated beforehand. For ExpScen & ExpMC, Figure 5 shows the tasks that are performed during the various modes of operation, in the form of Nassi-Schneiderman diagrams. In the case of distributed execution, experiments are first to be run in Mode 2, followed by processing of all generated job descriptions in a distributed execution environment, and completed with a run of the same experiment in Mode 3.

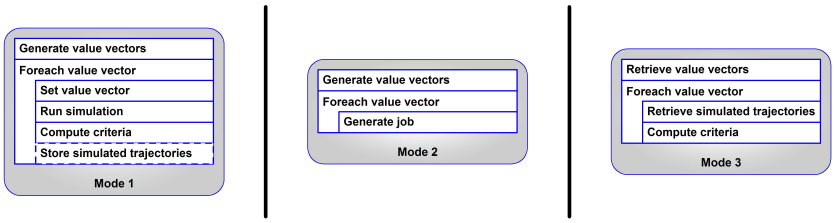

Fig. 5. Modes of Operation for ExpScen/ExpMC

Figure 6 (left) represents the types of nodes that are commonly found in grid & cluster systems. Jobs generated on the user's workstation (WS) are transferred to the distributed execution environment's user interface (UI), from where they are submitted to a resource broker (RB). The resource broker subsequently assigns jobs to computational elements (CE), using a particular scheduling policy. Input resources usually travel from WS to UI, from where they are uploaded to a storage element (SE). CE's will retrieve the input resources required for the execution of jobs from the SE. Output resources follow an inverse path: they are uploaded to the SE from CE's and are then transferred to the WS through the UI. Figure 6 (right) represents the relationship between the various experiment modes of operation (again taking ExpScen/ExpMC as an example) and the grid nodes.

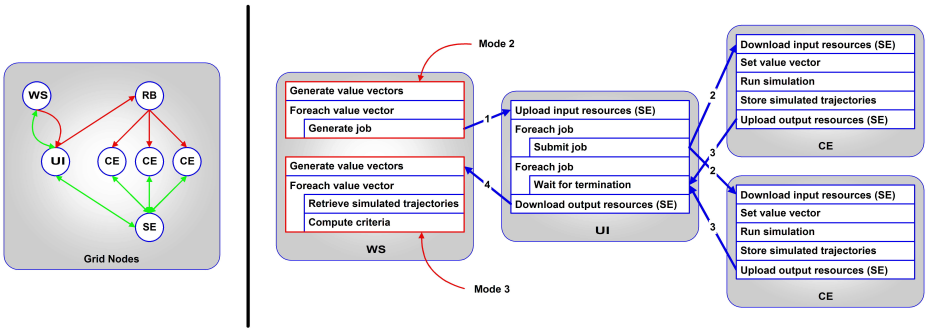

Fig. 6. Relationship between Distributed Execution and Modes of Operation

4 Tornado and Typhoon

Typhoon [5] is a lightweight distributed execution environment for clusters that was developed at BIOMATH, using technologies and design principles similar to Tornado. It consists of two types of components: a Master that directly interprets Typhoon XML job descriptions and provides a static scheduler, and Slaves that execute jobs handed over by the Master. Next to generic (*i.e.*, application-independent) information on input and output resources, Slaves also receive the application-specific section of job descriptions (*i.e.*, the XML content below the "App" tag in Figure 4). In order to parse and interpret this application-specific information, application-specific executor plug-ins can be loaded into each Slave. At BIOMATH, Typhoon is available on a 16-node cluster.

5 Tornado and LCG-2

LCG-2 is the middleware that has been developed at CERN in the scope of the Large Hadron Collider project. On the basis of the LCG-2 middleware, the EGEE[5] grid is operated. EGEE is the largest grid to date and consists of over 20,000 CPU's in addition to about 5 Petabytes of storage. At Ghent University, a 40-node inter-faculty LCG-2 grid has been set up, which has recently been linked to EGEE.

LCG-2 is heavily command-line oriented and does not support the notion of sets of jobs. It has its own job description language (JDL - Job Description Language). For the execution of job content, it does not have a plug-in mechanism but relies on plain operating system commands. As a consequence, a job description generated by Tornado that describes a set of N jobs, must be translated to N JDL files and N shells scripts defining the actual task to be executed by the work nodes. The program that performs this conversion was named `t2jdl`. In order to facilitate the process of submitting the JDL files generated by `t2jdl`, the latter also generates a number of convenience scripts, allowing for LCG-2 to be used without specific knowledge of its command-line suite.

6 Adoption of Distributed Execution in the CD4WC Project

CD4WC[6] stands for "Cost-effective development of urban wastewater systems for water framework directive compliance" and is an EU project that deals with optimizing the efficiency of the urban wastewater system with regard to ecological consequences in rivers on the one hand, and investment and operation costs at the other. The need to solve this problem is a direct consequence of the European Water Framework Directive (WFD) which requests to achieve good quality for ground and surface waters on a river-basin scale. With this new water-quality based approach, the design of the systems is by far less pre-determined and the options to meet the goals become much more widespread. Criteria to assess the ecological consequences are - besides the water quality - also secondary resource inputs such as energy, materials and chemicals. Various options and strategies to develop the wastewater system are to be evaluated. Main

[5] http://www.eu-egee.org
[6] http://www.tu-dresden.de/CD4WC

emphasis is on the dynamic interactions between the sewer, treatment plant and river subsystems as well as on the possibilities of taking measures in the receiving water and at the sources.

BIOMATH's responsibility in the scope of CD4WC was to evaluate the impact of a number of WWTP plant design and upgrade scenarios [10]. To this end, 100-shot Monte Carlo simulation was adopted (using LHS - Latin Hypercube Sampling). For design, 10 plant layouts had to be examined for 4 climates and 3 capacities (as listed in Table 1, left), yielding a total number of 100*10*4*3 = 12,000 simulations. In addition, 12 upgrade scenarios had to be evaluated, however only for 2 climates and 1 capacity (*cf.* Table 1, right), yielding a total number of 100*12*2*1 = 2,400 simulations. Given the fact that one simulation on average lasts for approximately one half hour (on a reference workstation - INTEL x86 3GHz), sequential execution of all cases would require (12,000+2,400)*0.5h = 7,200h or 300 days. Clearly, sequential execution in this case is not a tractable solution.

Layouts (design)	**Climates**	**Capacities**
Anaerobic-anoxic-oxic (A2O)	Mediterranean	300k PE
Anaerobic-oxic (AO)	Continental	30k PE
Biodenipho	Alpine	3k PE
Biodenitro	Oceanic	
High loaded AS (HLAS)		
Low loaded AS with chemical P removal (LLAS)		
LLAS with primary settler (LLAS_PS)		
Oxidation ditch with bio-P removal (OD_bioP)		
Oxidation ditch with chemical P removal (OD_simP)		
University of Cape Town process (UCT)		

Layouts (upgrade)	**Climates**	**Capacities**
Increase of aerated tank volume by 33% (U1)	Mediterranean	300k PE
U1+increase of final clarifier area by 33%	Continental	
U1+pre-anaerobic tank+C dosage+lower DO setpoint (U3)		
Dosage of external C (U4)		
DO control based on ammonia (U5)		
Internal recycle control based on nitrate (U6)		
U4+U6 (U7)		
Spare sludge storage (U8)		
Sludge wastage control (U9)		
Dynamic step feed (U10)		
Increase in anoxic volume, decrease in aerated volume (U11)		
Buffering ammonia peak loads with the storm tank (U12)		

Table 1. Design Cases & Upgrade Scenarios (PE = People Equivalents)

Since BIOMATH has access to two distributed execution environments (its own 16-node cluster running Typhoon and the 40-node inter-faculty grid running LCG-2), the computational load generated by the CD4WC project was split over both environments. In a first instance, Tornado Monte Carlo experiments for each of the (10*4*3 + 12*2*1) = 144 parameterized plant layouts (= models) were created. All experiments were configured for 100 shots. Afterwards, generic job set descriptions were generated from all experiments by running the experiments in Mode 2. The 24 upgrade-related job set descriptions that were generated were subsequently directly presented to the Typhoon cluster. The remaining 120 design-related job set descriptions were automatically converted to a total of 120*100 = 12,000 individual LCG-2 jobs through `t2jdl`.

As it is a much simpler system, Typhoon has a higher efficiency than LCG-2, *i.e.*, the amount of overhead is lower and the number of compute cycles that can be spent on real workload is higher. The efficiency of the current BIOMATH Typhoon cluster is approximately 75% (empirical number), whereas for the inter-faculty LCG-2 grid an efficiency of 60% can be put forward. Taking into account these efficiency values, the total processing time of the Typhoon and LCG-2 workload is as follows:

– Typhoon: (2,400*0.5h) / (0.75*16) = 100h or 4.2 days
– LCG-2: (12,000*0.5h) / (0.6*40) = 250h or 10.4 days

Instead of the 300 days that would have been required in case of sequential execution, using Typhoon & LCG-2 only required a total of 4.2 + 10.4 = 14.6 days of processing. This is less than 5% of the total sequential execution time. In addition, thanks to the semi-automated distributed execution facilities of Tornado, the work on the CD4WC project could be performed by a small team of water quality experts with little or no computer science knowledge.

7 Conclusions

Tornado is a computationally-intensive software system that is well-suited for distributed execution. In order to facilitate adoption by its diverse user community, transparent integration of distributed computing infrastructures into the kernel is required. The article shows that the semi-automated procedures that have so far been implemented constitute an important step towards full transparency. Still lacking however is the fully automated transfer of job descriptions to the distributed execution environment of choice.

References

1. Reichert, P., D., B., M., H., W., R., P., S., L., S., Vanrolleghem, P.: River Water Quality Model No.1. Scientific and Technical Report No.12. IWA Publishing, London, UK (2001)
2. Henze, M., Gujer, W., Mino, T., van Loosdrecht, M.: Activated Sludge Models ASM1, ASM2, ASM2d, and ASM3. Scientific and Technical Report No.9. IWA Publishing, London, UK (2000)
3. Copp, J., ed.: The COST simulation benchmark. European Commission (2002)
4. Claeys, F., De Pauw, D., Benedetti, L., Nopens, I., Vanrolleghem, P.: Tornado: A versatile efficient modelling & virtual experimentation kernel for water quality systems. In: Proceedings of the iEMSs 2006 Conference, Burlington, VT (2006), *Accepted*
5. Claeys, F., Chtepen, M., Benedetti, L., Dhoedt, B., Vanrolleghem, P.: Distributed virtual experiments in water quality management. Water Science and Technology **53**(1) (2006) 297–305
6. Vanhooren, H., Meirlaen, J., Amerlinck, Y., Claeys, F., Vangheluwe, H., Vanrolleghem, P.: WEST: modelling biological wastewater treatment. Journal of Hydroinformatics **5**(1) (2003) 27–50
7. Fritzson, P.: Principles of Object-Oriented Modeling and Simulation with Modelica 2.1. Wiley-IEEE Press (2004)
8. Claeys, F., Vanrolleghem, P., Fritzson, P.: A generalized framework for abstraction and dynamic loading of numerical solvers. In: Proceedings of the 2006 European Modeling and Simulation Symposium, Barcelona, Spain (2006), *Accepted*
9. Aronsson, P.: Automatic Parallelization of Equation-Based Simulation Programs. PhD thesis, Linköping Studies in Science and Technology, Dissertation No. 1022, Linköping University, Sweden (2006)
10. Benedetti, L., Bixio, D., Claeys, F., Vanrolleghem, P.: A model-based methodology for benefit/cost/risk analysis of wastewater systems. In: Proceedings of the iEMSs 2006 Conference, Burlington, VT (2006), *Accepted*