FLOOD PREDICTION MODEL SIMULATION WITH HETEROGENEOUS TRADE-OFFS IN HIGH PERFORMANCE COMPUTING FRAMEWORK

Antoni Portero
Radim Vavrik
Stepan Kuchar
Martin Golasowski
Vit Vondrak
IT4Innovations
VSB-Technical University of Ostrava
Ostrava, Czech Republic
Email: antonio.portero@vsb.cz

Simone Libutti
Giuseppe Massari
William Fornaciari
Politecnico di Milano
Dipartimento di Elettronica,
Informazione e Bioengegneria (DEIB)
Milano, Italy
Email: simone.libutti@polimi.it

KEYWORDS

Hydrology model simulation; High Performance Computing systems; disaster management; reliability models; computing resource management; multi and many-cores systems; parallel systems

ABSTRACT

In this paper, we propose a safety-critical system with a run-time resource management that is used to operate an application for flood monitoring and prediction. This application can run with different Quality of Service (QoS) levels depending on the current hydrometeorological situation.

The system operation can follow two main scenarios - standard or emergency operation. The standard operation is active when no disaster occurs, but the system still executes short-term prediction simulations and monitors the state of the river discharge and precipitation intensity. Emergency operation is active when some emergency situation is detected or predicted by the simulations. The resource allocation can either be used for decreasing power consumption and minimizing needed resources in standard operation, or for increasing the precision and decreasing response times in emergency operation. This paper shows that it is possible to describe different optimal points at design time and use them to adapt to the current quality of service requirements during run-time.

INTRODUCTION

Urgent computing (Beckman 2008) prioritises and provides immediate access to supercomputers and grids for emergency computation such as severe weather prediction during disaster situations. For these urgent computations, late results are useless results, because the decisions based on these results have to be made before the disaster happens and endangers human lives or property. As theHigh Performance Computing (HPC) community builds increasingly realistic models, applications that need on-demand computation are emerging. Looking into the future, we might imagine event-driven and data-driven HPC applications running on demand to predict any kind of event, e.g. running air pollution dispersion models at real time to predict hazardous substance dispersion caused by some accident. Of course, as we build confidence in these emerging computations, they will move from the scientists workbench into critical decision-making paths, but HPC clusters will still be needed to execute and analyse these simulations in a short time frame and they will have to provide the results in a reliable way based on the specified operational requirements. The intent of this paper is to describe the characteristics and constraints of disaster management (DM) applications for industrial environments.

We therefore propose an HPC system running with a Run-Time Resource Manager (RTRM) that monitors the healthiness of the platform. Our in-house disaster management application called Floreon^{+} (Martinovič, Kuchař, Vondrák, Vondrák, Šír & Unucka 2010) is running on the system. The application can run with different Quality of Service (QoS) levels depending on the situation. Taking the current situation in mind, the resource allocation can be used either for decreasing power consumption and minimizing needed resources in standard operation or increasing the precision and decreasing response times in emergency operation. Power consumption is a critical consideration in high performance computing systems and it is becoming the limiting factor to build and operate petascale and exascale systems in the future. When studying the power consumption of existing systems running HPC workloads, we find that power, energy and performance are closely related leading to the possibility to optimize energy without sacrificing performance.

This paper shows that it is possible to describe different optimal points at design time and use them to adapt to the current QoS requirements during run-time. The proposed process can then be used to prepare an operational environment with high

Proceedings 29th European Conference on Modelling and Simulation ©ECMS Valeri M. Mladenov, Petia Georgieva, Grisha Spasov, Galidiya Petrova (Editors)
ISBN: 978-0-9932440-0-1 / ISBN: 978-0-9932440-1-8 (CD)

availability and varying energy cost and model precision levels. We manage to adapt the precision of the model by changing its input parameters and configuration, e.g. changing the number of Monte-Carlo samples used for uncertainty modelling. The change of precision alters the model execution time and/or the number of required computational resources, effectively changing the energy cost of the simulation. The decision to focus on the precision or energy cost is taken by the RTRM that computes the best-trade off for the current QoS.

This paper is divided into seven sections. A section containing the current state-of the art follows this introduction section. In section three, we introduce our driving disaster management example in the form of a rainfall-runoff uncertainty model. Section four describes the run-time operating (RTRM) system that monitors the sensors and knobs of the platform. Section five shows the possible way of RTRM deployment in an HPC environment. Section six presents the results of possible optimal computation points that are obtained at design time for the described solution deployed on single computing node and on multiple nodes. These optimal points are used by the RTRM at run-time and the results show the performance and energy efficiency of the deployed solution. The final section provides a summary of the presented research and possible future work.

THE STATE OF THE ART

Elastic computing cloud (Galante & de Bona 2012) can provide thousands of virtual machine images rapidly and cost effectively. Unfortunately, one of the main drawbacks of elastic computing is the application scalability due to computer limited bandwidth communication. Another problem is the feasibility of allocating resources and the infrastructure size that can be used on demand in case of emergency; sometimes, it cannot be determined fast enough. Therefore, one of the goals of the future HPC systems is to develop automated frameworks that use power and performance knobs to make applications-aware energy optimizations during execution using techniques like dynamic voltage and frequency scaling (DVFS) (Iraklis Anagnostopoulos 2013) for reducing the speed (clock frequency) in exchange for reduced power. Power gating technique then allows for the reduction of power consumption by shutting off the current to blocks of the circuit that are not in use, in case that diverse computations have different power requirements. For example, when a CPU is waiting for resources or data, the frequency can be reduced to lower power consumption with minimal performance impact (Tiwari, Laurenzano, Peraza, Carrington & Snavely 2012).

System scenarios classify system behaviours that are similar from a multi-dimensional cost perspective, such as resource requirements, delay, and energy consumption, in such a way that the system can be configured to exploit this cost. At design-time, these scenarios are individually optimized. Mechanisms for predicting the current scenario at run-time and for switching between scenarios are also derived. These are inferred from the combination of the application behaviour and its mapping on the system platform. These scenarios are used to reduce the system cost by exploiting information about what can happen at run-time to make better design decisions at design-time, and to exploit the time-varying behaviour at run-time. While application scenarios classify the application's behaviour based on the different ways the system can be used in its over-all context, system scenarios classify the behaviour based on the multi-dimensional cost trade-off during the implementation (Gheorghita, Vandeputte, Bosschere, Palkovic, Hamers, Vandecappelle, Mamagkakis, Basten, Eeckhout, Corporaal & Catthoor 2009).

RTRM: Run-Time Resource Manager

The RTRM (Run-Time Resource Manager) framework (Bellasi, Massari & Fornaciari 2015) is the core of a highly modular and extensible run-time resource manager which provides support for an easy integration and management of multiple applications competing on the usage of one (or more) shared many-core computation devices. The framework design, which exposes different plug-in interfaces, provides support for pluggable policies for both resource scheduling and the management of applications coordination and reconfiguration.

With RTRM, it is possible to make a suitable instrumentation to support Design-Space-Exploration (DSE) techniques, which could be used to profile application behaviour to either optimize them at design time or support the identification of optimal QoS requirement goals as well as their run-time monitoring. Suitable platform abstraction layers, built on top of GNU/Linux kernel interfaces, allow an easy porting of the framework to different platforms and its integration with specific execution environments.

The configuration of a run-time tunable application is defined by a set of parameters. Some of them could impact the application behaviour (e.g. the uncertainty of rainfall-runoff models can produce different results for different scenarios) while other have direct impact on the amount of required resources (e.g. the amount of Monte-Carlo method samples in uncertainty modelling and the time between simulation batches leads to different requirements for allocating system resources).

FLOOD MONITORING AND PREDICTION MODEL

The studied flood prediction model is a modular part of the Floreon$^+$ system (Martinovič et al. 2010). The main objective of the Floreon$^+$ system is to create a platform for integration and operation of monitoring, modelling, prediction and decision support for disaster management. Modularity of the Floreon$^+$ system allows for a simple integration of different thematic areas, regions and data. The central thematic area of the project is flood monitoring and prediction. The system focuses on acquisition and analysis of relevant data in near-real time and using these data to run hydrologic simulations with short-term prediction. The results are then used for decision support in disaster management processes by providing predicted discharges on river gauges and prediction and visualization of inundated areas in the landscape.

A model used in this paper is a rainfall-runoff model developed as part of the Floreon$^+$ system (Golasowski, Litschmannova, Kuchar, Podhoranyi & Martinovic 2015), (Vavrik, Theuer, Golasowski, Kuchar, Podhoranyi & Vondrak 2015). The RR models transforms precipitation to water discharge levels by modelling individual parts of the rainfall-runoff process. Common inputs of these models are an approximation of physical and spatial properties of modelled river catchment and time series of precipitations. Outputs of these models are usually represented by time series of water discharge levels (i.e. the relation between water discharge (*Q*) and time (*t*)) for modelled parts of the river course. Our RR model uses SCS-CN method (Beven 2012) for transforming rainfall to runoff with the main parameter curve value (*CN*) approximated from the hydrological soil group, land use and hydrological conditions of the modelled catchments. The contribution from river segments to a sub-basin outlet is computed using the kinematic wave approximation parametrized by Manning's roughness coefficient (*N*), which approximates physical properties of the river channel.

The input precipitation data for short-term prediction are provided by numerical weather prediction models and can be affected by certain inaccuracy. Such inaccuracy can be projected into the output of the model by constructing confidence intervals. These intervals provide additional information about possible uncertainty of the model output and are constructed using the Monte-Carlo (M-C) method. Data sets are sampled from the model input space and used as input for a large number of simulations. Precision of uncertainty simulations can be affected by changing the number of provided M-C samples. The precision of the simulations can be determined by computing the Nash-Sutcliffe (*NS*) model efficiency coefficient between the original simulation output and one of the percentiles selected from the Monte-Carlo results. The *NS* coefficient is computed by formula (1) and is often used for estimating precision of given model by comparing its output with the observed data, but it can also be used for comparison of any two model outputs.

$$NS = 1 - \frac{\sum_{t=1}^{T}(Q_o^t - Q_s^t)^2}{\sum_{t=1}^{T}(Q_o^t - \overline{Q_o})^2} \qquad (1)$$

Where:
Q_o is the observed flow in a specific time-step.
Q_s is the simulated flow in a specific time-step.
NS is the Nash-Sutcliffe model efficiency coefficient.
$NS = 1$ means that the simulation matches observed data perfectly.
$NS = 0$ means that the simulation matches a median of the observed data.
$NS < 0$ means that the simulation is less precise than the median of the observed data.

Fig. 1 shows the precision of the simulated uncertainty results based on the number of Monte-Carlo samples and different combinations of modelled parameters in the experimental model. *P* shows the standard deviation of uncertainty results where only uncertainty of input precipitations were taken into account, *N* stands for the uncertainty model of the Manning's coefficient, *CN* for the CN uncertainty model, *N + CN* describes the combination of Manning's coefficient and CN uncertainty models, and so on. These results show that the standard deviation is unstable for low numbers of M-C samples, but starts to decrease steadily from around 5-8 samples. Precipitation uncertainty models and their combinations also show much higher standard deviations than *CN* and *N* based models. To obtain the best precision (i.e. standard deviation close to 0), the number of samples of the precipitation based models has to be in the order of 10^4 to 10^5 depending on the number of modelled parameters and the complexity of the model, while the *CN* and *N* based models are very close to 0 even for orders of 10^1. This was mainly caused by small deviations in uncertainty of *CN* and *N* when compared to precipitations and also to the fact, that precipitation uncertainty was sampled for each time-step of the simulation and each observed gauge, while *CN* and *N* did not depend on time.

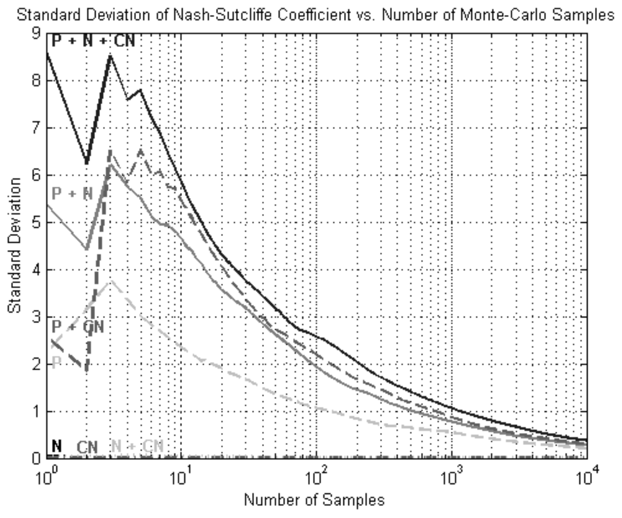

Fig. 1. Comparison of the precision provided by standard deviation of Nash-Sutcliffe coefficient and the number of Monte-Carlo samples for different model parameters and their combinations.

Application Scenarios

Application scenarios describe different triggers and states of the application that influence the system responsiveness and operation (e.g. critical flooding level, critical state of patient's health, voice & data, etc.). Based on these scenarios, the system can be in different states with different service level requirements that can also be translated to the parameters of the uncertainty modelling and its required resources. Shorter response time in critical situations can for example be acquired by decreasing the number of Monte-Carlo samples (also decreasing the precision of the results), or by allocating more computational resources (maintaining the same precision level if the load is rebalanced appropriately). We identified two main application scenarios (Portero, Kuchař, Vavřík, Golasowski & Vondrák 2014) that support the different workload of the system based on the flood emergency situation.

Standard Operation: In this scenario, the weather is favourable and the flood warning level is below the critical threshold. In this case the computation can be relaxed and some kind of errors and deviations can be allowed. The system should only use as much power as needed for standard operation. One batch of Rainfall-Runoff simulations with uncertainty modelling only has to be finished before the next batch starts. The results do not have to be available as soon as possible, so no excess use of resources is needed.

Emergency Operation: Several days of continuous rain raise the water in rivers or a very heavy rainfall on a small area creates new free-flowing streams. These conditions are signalled by the river water level exceeding the flood emergency thresholds or precipitation amount exceeding the flash flood emergency thresholds. Much more accurate and frequent computations are needed in this scenario and results should be provided as soon as possible even if excess resources have to be allocated.

RTRM METHODS: ABSTRACT EXECUTION MODEL

The RTRM exposes a Run-Time Library (RTLib) (Bellasi et al. 2015), (Bellasi, Massari, Libutti & Fornaciari 2014), which is linked to the integrated applications. Besides controlling the execution of the application, the library transparently monitors the application and profiles its execution. This activity is useful at both design-time and run-time. In the former case, it serves as an informative support for the developer. In the latter, profiling data can be used as an input to the RTRM resource allocation policy, or external monitoring tools. The RTLib provides the Abstract Execution Model, which encapsulates the execution flow of the application in a way that lets the RTRM govern its life-cycle. Generally, we can structure an application by splitting it into several Execution Contexts. From the RTRM point of view, an Execution Context is seen as a task that has to be scheduled. Having more contexts in the same application can come from the need to schedule different parts of the application with different priorities and resource usages. This is possible using the recipe that is associated with specific contexts.

To fully exploit the RTRM, the application has to be implemented in a way that allows its reconfiguration during runtime. From the RTRM side, the configuration of an application is characterized by a set of resource requirements, and is called Application Working Mode (AWM). Fig. 2 summarizes the Abstract Execution Model and shows different methods that have to be supported by the application:

- *onSetup*: Setting up the application (initialize variables and structures, starting threads, etc.).
- *onConfigure*: Configuration / re-configuration code (adjusting parameters, parallelism, resources etc.).
- *onRun*: Single cycle of computation (e.g., computing a single rainfall-runoff simulation for one Monte-Carlo sample).
- *onMonitor*: Performance and QoS monitoring.
- *onRelease*: Code termination and clean-up.

onSetup method

Initialization of the framework is done in the *onSetup* method. In order to allow the RTRM to carry out a correct structure initialization, it has to support the run-time statistics recollection.

In our experimental system, the *onSetup* method sets all initial input parameters of the uncertainty model:

- *Schematization*: Description of all hydrologic and geographic parameters of the modelled area.

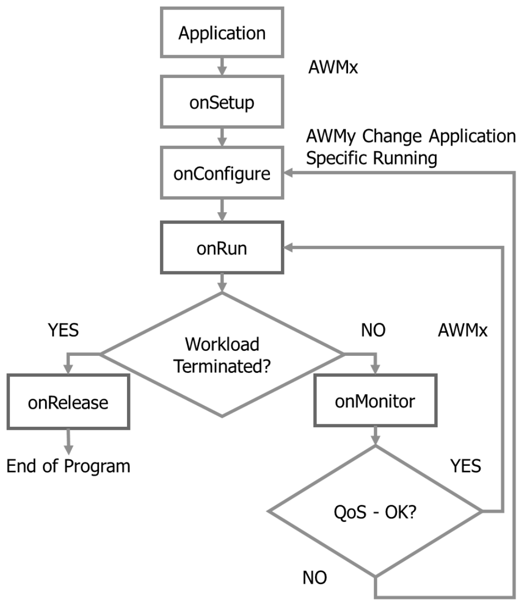

Fig. 2. Diagram of the RTRM methods.

- *CN, N range and precision*: Probability function and its parameters used for generating samples of the *CN* and *N* values.
- *Input rainfall data*: Rainfall input data from the modelled area received from the meteorological gauges and prediction models. In the experimental system, new observed data are received every 10 to 60 minutes, new meteorological predictions are received every 6 hours.
- *Predicted rainfall error distribution*: Probability distribution of the used rainfall prediction models' errors that are taken into account in uncertainty simulations. This distribution is constructed statistically from the past prediction results and their comparison with observed values.
- *Threads for uncertainty*: The number of threads for parallel Monte-Carlo method execution in each computational node.

onConfigure method

This method is called when the RTRM assigns resources to the application for the first time or changes the allocated resources to provide expected QoS. The RTRM selects the most appropriate Application Working Mode (AWM) based on the current scenario, execution environment, state of the application and required QoS and passes it to this method so the application can reconfigure its parameters (e.g. number of threads, application parameters, data structures, etc.) to properly run on the allocated resources.

In our experimental system, the number of threads for uncertainty computation along with the number of Monte-Carlo samples and modelled parameters can change during this method. This can effectively change the amount of computation needed as well as the expected precision level of the results.

onRun method

This method runs the uncertainty model with the parameters set in the onSetup and onConfigure methods (for more information about the experimental uncertainty model see (Golasowski et al. 2015)). Aside from the standard uncertainty percentiles computed by the model, the method also computes the precision of the current execution and saves it for monitoring and control purposes in the onMonitor method.

onMonitor method

After each computational run, the application checks whether the level of QoS is acceptable or not. This method is in charge of monitoring the results received from the uncertainty model. It decides if the AWM is enough to compute the next uncertainty loop or if different AWM has to be selected to adapt the system to the new situation during run-time.

onRelease method

This is a finalization method that is called when the whole application ends and contains clean-up of the used resources and data structures. The system eventually does not have to arrive to this point, because it is expected to be running 24/7, but it still has to be implemented for the case when the system has to be shut down for maintenance or updates.

Application Working Modes (AWMs)

The application also has to specify a list of its AWMs and it has to be provided by a file called *recipe* which is basically an XML file with information about each possible optimal point (i.e. amount of memory allocated for the application, number of allocated x86-64 cores and cluster nodes, allocation of accelerators, etc.).

An example of AWMs for RR uncertainty modelling on one of our testing SMP machines with 42 cores could be:

- AWM_1 for Standard Operation - basic resources needed for simulation execution in the allotted time - 16 cores and 8 GB of RAM. No accelerators are used. The number of M-C samples is 1000 to ensure good enough precision of the results while maintaining lower resource needs.
- AWM_2 for Emergency operation - allocated CPU and memory quotas are higher, using all available resources of the machine 42 cores and 220 GB RAM. GPGPU accelerator can be used by the application to further enhance vectorized operations. The number of M-C samples is 20000 to increase the precision of the results.
- $AWM_{i>2}$: A series of intermediate optimal points or Pareto points are set for providing other possible QoS, i.e. intermediate numbers of Monte Carlo iterations and required resources.

Specific AWMs for the production platform will be based on the experiments described in the following sections.

The novelty of the developed system lies in the utilization of the RTRM in the HPC environment for systems that consider a trade-off between quality of the result (precision) and resource utilization. The solution provided in this paper helps to decrease the power consumption and the cost of running the system with high availability. The use of extensive resources with high energy cost is only authorized in emergency situations that are identified automatically in run-time using pre-specified rules. The application therefore does not have to be reconfigured manually and can quickly react to changes in the environment.

RTRM DEPLOYMENT ON THE HPC SYSTEM

To validate the use of the proposed solution, we have deployed a hybrid OpenMP (Board 2013) and MPI (Forum 2012) implementation of the uncertainty modelling (Golasowski et al. 2015) to the Anselm supercomputer operated by IT4Innovations Czech National Supercomputing Center (Sliva & Stanek 2013). The cluster contains 209 nodes each with 2 Intel(R) Sandy Bridge E5-2665 @ 2.4GHz CPU sockets with 8 cores and 20480 KB L2 cache. These nodes are equipped with 64GB RAM and are connected by high speed InfiniBand (3600MB/s) in a fat tree organization network. All nodes share a Lustre parallel file system with a throughput of 6 GB/s. The flood monitoring and prediction application runs under a GNU/Linux operating system with the RTRM support. The tool manages the workload of the application and will also help in the attenuation of hardware errors in the future.

The cluster is formed by two basic types of nodes. Nodes of the first type are standard computation nodes where executions of applications are made through a scheduler. The job allocation in the scheduler is realized via the PBS Pro job workload manager software, which efficiently distributes workload across all computation nodes of the supercomputer.

The second type of nodes are the management nodes that are used to support and monitor the system. The RTRM infrastructure is installed on two of these nodes and any of these two nodes can execute and monitor the application. If one node is down, the other can replace it. The RTRM monitors the healthiness of its both nodes and uses one of the cores for each socket for monitoring the system and the sensors of the platform (i.e. power, temperature etc.). Therefore, there are only 14 real cores left for executing managed applications. If the resources available on one node are not enough to fulfil the QoS requirements, then the execution can also run in the second node. If the situation becomes critical, and the system cannot meet the requested QoS, the RTRM asks for more resources through a special AWM. This AWM contacts the HPC scheduler and distributes the workload to the rest of the HPC cluster with high priority.

EXPERIMENTS IN THE CLUSTER

We have performed resource allocation experiments with the RTRM in a single node and in multiple nodes of the cluster. Experiments presented in the following subsections show that model executions with 10^3 to 10^4 samples can be fitted in one node. But when a more accurate precision is needed and the number of samplings has to be higher than 10^4, more nodes of the cluster have to be utilized to fulfil the constraint on the execution time.

Single node results

Fig. 3 and Fig. 4 present estimated power consumption of the uncertainty model for 10^3 and 10^4 samples respectively. Different points represent simulations with different CPU frequencies and various numbers of cores, which result in different execution times and appropriately consumed energy. This leads to trade-offs for running the system with various AWMs that represent diverse resource utilization (cores, frequencies). The optimal values are the ones that minimize the power and execution time but still meet the time constraints. Fig. 3 shows the results with lower precision levels that could be used in the standard operation scenario and several possible optimal points are highlighted as (1a) and (1b). As the execution time limit in the standard operation is 3600 seconds, the optimal point 1a can be selected for its lower power consumption and still fulfil the QoS specified by this scenario. We can then define an AWM of one or two cores and the lower CPU frequency (1.2Ghz) based on this optimal point and the RTRM will use this AWM for the standard scenario in run-time.

Fig. 4 shows the results with higher precision for the emergency operation scenario. As the execution time limit for this scenario is only 10 minutes, the number of optimal points is reduced to the set highlighted as (2). This defines the AWM with 8 to 14 cores with the maximal CPU frequency (2.6GHz).

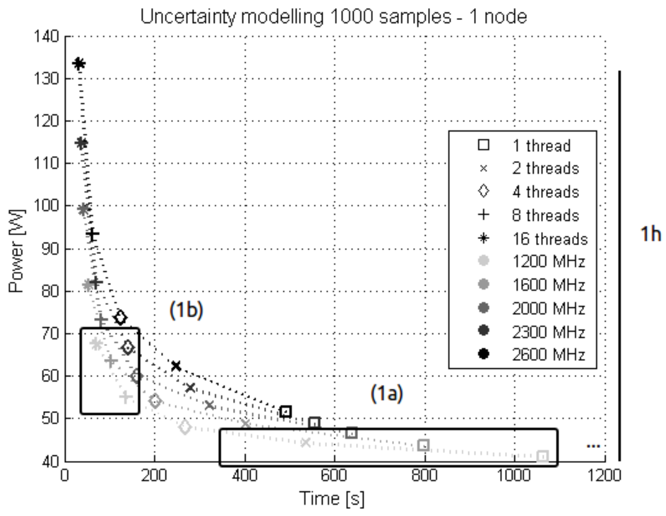

Fig. 3. Uncertainty execution time versus power consumption for 10^3 samples.

The overhead time between the execution with and without RTRM is less than 10% in average, which is shown in Fig. 5. This overhead is produced in the *onConfigure* and *onMonitor* methods. Currently, these two methods are not very complex and the number of lines is relatively low, but when more complex monitoring and reconfiguration requirements are introduced, the overhead can increase significantly. Our further development will focus on keeping the implementation of these requirements lean and simple to retain low overhead.

Multi-node results

One important goal of our development is that the RTRM has to be aware of the global architecture of the infrastructure and be able to submit urgent jobs to the cluster. Hence, when the emergency scenario is active, there is a possibility to support urgent computing on the cluster with high priority and high precision, e.g. 10^5 Monte-Carlo samples.

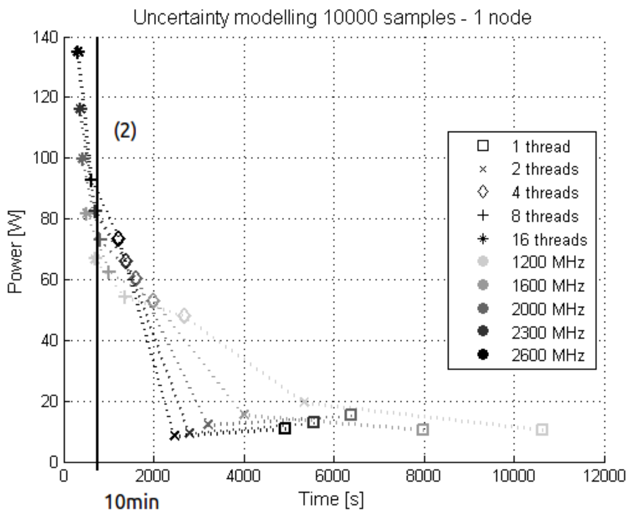

Fig. 4. Uncertainty execution time versus power consumption for 10^4 samples.

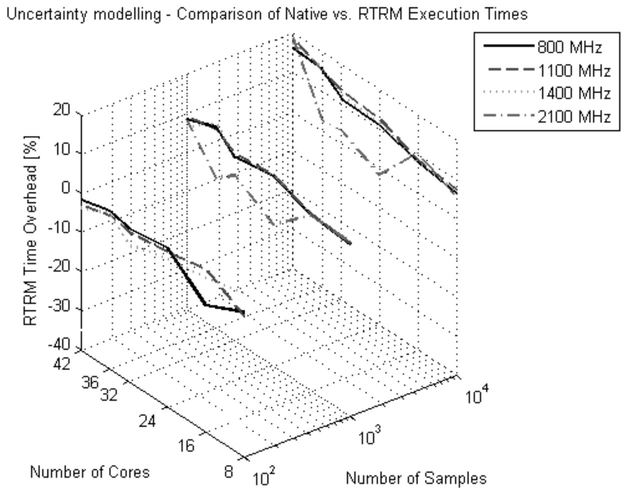

Fig. 5. Comparison of the RTRM time overheads for different configurations. Experiments performed in a testing 42 core SMP machine.

Fig. 6 compares the execution time and power consumption of the uncertainty model with 10^5 M-C samples running in the Anselm cluster. We use the cluster in the case, when the resources of one node do not satisfy the performance requirements of the emergency scenario. The highlighted optimal points (3) show that using 8 to 32 nodes of 16 cores at maximum frequency is the best compromise to fulfil the required service level with an execution time lower than 10 minutes.

These results show that we do not have to strictly abide by the two simple application scenarios (standard and emergency operation), but it is possible to refine them further. The system could for example provide a low energy mode for low precipitation intensities and standard river discharges (single-node optimal point 1a), awareness mode when the weather gets worse (single-node optimal point 1b), warning mode when the parameters are nearing emergency (single-node optimal point 2) and emergency mode (multi-node optimal point 3).

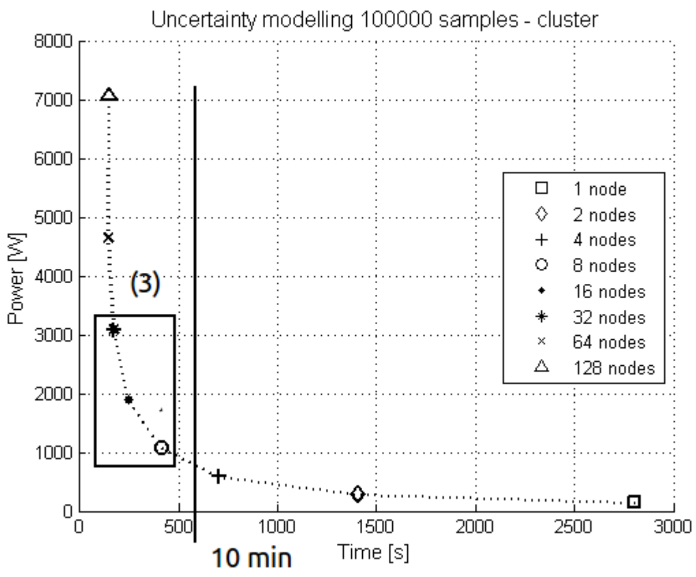

Fig. 6. Comparison of power consumption depending on the number of nodes (16 cores@2.4GHz) and execution time of the uncertainty model with (10^5 Monte-Carlo samples).

CONCLUSIONS AND FUTURE WORK

This paper presented an environment that allows to effectively run a disaster management application with high availability while trading the accepted loss of precision for lower energy consumption in the standard operation and increasing the number of allocated resources to provide higher precision levels and lower execution times for emergency operation. The uncertainty modelling module of the disaster management application was first executed in one node of the HPC cluster and dependency of precision on the number of Monte-Carlo samples was measured to provide exact information about attainable precision levels. This precision is then taken into account in deciding the Application Working Modes that balance the trade-offs between precision, executed time and consumed energy for different application scenarios. Finally, the Run-Time Resource Manager is deployed on the HPC cluster to monitor the current workload of the platform and select the most appropriate working mode to reconfigure the application in run-time. The proposed process can then be used to prepare an operational environment with high availability while saving energy at the cost of lower precisions during non-critical situations.

Our future work will focus on extending the implementation of the application to support GPGPU and Xeon Phi accelerators and to improve the support of multi-node communication and monitoring. This will enable the creation of additional working modes with accelerators for high performance computations that will lead to even shorter execution times with high precision levels. Comparing the power consumption of these accelerator-enabled working modes with standard multi-node working modes will also be of interest to further support the run-time decisions and reconfigurations.

ACKNOWLEDGEMENTS

This article was supported by Operational Programme Education for Competitiveness and co-financed by the European Social Fund within the framework of the project New creative teams in priorities of scientific research, reg. no. CZ.1.07/2.3.00/30.0055, by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), by the project Large infrastructures for research, development and innovation of Ministry of Education, Youth and Sports of Czech Republic with reg. no. LM2011033, and by 7th EU framework programme project no. FP7-612069 HARPA - Harnessing Performance Variability.

REFERENCES

Beckman, P. (2008), urgent computing: Exploring supercomputings new role , *CTWatch Quarterly* 4(1), 3–4.

Bellasi, P., Massari, G. & Fornaciari, W. (2015), Effective runtime resource management using linux control groups with the BarbequeRTRM framework, *ACM Trans. Embed. Comput. Syst.* 14(2), 1–17.

Bellasi, P., Massari, G., Libutti, S. & Fornaciari, W. (2014), Bosp: The barbeque opensource project. http://bosp.dei.polimi.it

Beven, K. (2012), *Rainfall-Runoff Modelling*, Wiley-Blackwell.

Board, O. A. R. (2013), OpenMP application program interface version 4.0. http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf

Forum, M. P. I. (2012), Mpi: A message-passing interface standard version 3.0. http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

Galante, G. & de Bona, L. C. E. (2012), A survey on cloud computing elasticity, 2012 IEEE Fifth International Conference on Utility and Cloud Computing, Institute of Electrical & Electronics Engineers (IEEE).

Gheorghita, S. V., Vandeputte, F., Bosschere, K. D., Palkovic, M., Hamers, J., Vandecappelle, A., Mamagkakis, S., Basten, T., Eeckhout, L., Corporaal, H. & Catthoor, F. (2009), System-scenario-based design of dynamic embedded systems, *ACM Trans. Des. Autom. Electron. Syst.* 14(1), 1–45.

Golasowski, M., Litschmannova, M., Kuchar, S., Podhoranyi, M. & Martinovic, J. (2015), Uncertainty modelling in rainfall-runoff simulations based on parallel monte carlo method, *International Journal on non-standard computing and artificial intelligence NNW* . Accepted for publication 2015.

Iraklis Anagnostopoulos, Jean-Michel Chabloz, I. K. A. B. A. H. D. S. (2013), Power-aware dynamic memory management on many-core platforms utilizing DVFS, *ACM Trans. Embed. Comput. Syst.* 13(1s), 1–25.

Martinovič, J., Kuchař, Š., Vondrák, I., Vondrák, V., Šír, B. & Unucka, J. (2010), Multiple scenarios computing in the flood prediction system FLOREON, ECMS 2010 Proceedings edited by A Bargiela S A Ali D Crowley E J H Kerckhoffs, European Council for Modeling and Simulation.

Portero, A., Kuchař, Š., Vavřík, R., Golasowski, M. & Vondrák, V. (2014), System and application scenarios for disaster management processes, the rainfall-runoff model case study, Computer Information Systems and Industrial Management, Springer Berlin Heidelberg, pp. 315–326.

Sliva, R. & Stanek, F. (2013), *Best Practice Guide Anselm: Bull Extreme Computing at IT4Innovations / VSB*, IT4Innovations, PRACE.

Tiwari, A., Laurenzano, M., Peraza, J., Carrington, L. & Snavely, A. (2012), Green queue: Customized large-scale clock frequency scaling, 2012 Second International Conference on Cloud and Green Computing, Institute of Electrical & Electronics Engineers (IEEE).

Vavrik, R., Theuer, M., Golasowski, M., Kuchar, S., Podhoranyi, M. & Vondrak, V. (2015), Automatic calibration of rainfall-runoff models and its parallelization strategies, AIP Publishing.