Run-Time Application Developing Phases for Energy Trade-Offs in HPC System Node

A. Portero, R. Vavrik, S. Kuchar, M. Golasowski, V. Vondrak
IT4Innovations National Supercomputing Center
VŠB-Technical University of Ostrava, Students káOstrava – Poruba
Czech Republic

Abstract--**In this paper is proposed a HPC system running with an RTOS that is aware and monitors the health of the platform. On the system, an application for disaster management is running. The application can be executed with different QoS depending on the situation. We defined two main situations normal execution when there is no possibility of disaster. And, a scenario where the possibilities for a disaster is very high. Then, the resource allocation for improving the precision and human decision has to be taken into account. The paper shows that at design time, it is possible to describe the different phases for developing a run-time application that is aware of the power consumption and tries to minimized when the situation is not in the scenario of criticality.**

Since, 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.

Keywords-HPC systems; disaster management; reliability models; computing resource management; multi and many-cores systems; parallel systems

I. INTRODUCTION

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. When studying the power consumption of existing systems running HPC workloads, we find power, energy and performance are closely related leading to the possibility to optimize energy without sacrificing (much or at all) performance. The paper shows that at design time, it is possible to describe the different phases for developing a run-time application that is aware of the power consumption and tries to minimized when the situation is not in the critical scenario. The paper is divided in five sections, after the introduction and state-of the art. In section three, we introduce our driving disaster management example named the uncertainty of rainfall-runoff model. Section four describes design phases for our system. Section five, shows some results and conclusions.

II. STATE OF THE ART

One of the goals of future HPC systems is to develop automated frameworks that uses power and performances to make applications-aware energy optimizations during execution using techniques like dynamic voltage and frequency scaling (DVFS) for reducing the clock frequency in exchange for reducing power. And, power gating technique that allows reducing power consumption, by shutting off the current to blocks of the circuit that are not in use.

Different computations have different power requirements. For computations where the CPU is waiting for resources the frequency can be reduced to lower power with minimal performance impact. A framework [3] 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 MIMD 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. Applications integrated with this framework get a suitable instrumentation to support Design-Space-Exploration (DSE) techniques, which could be used to profile application behaviours to either optimize them at design time or support the identification of optimal QoS [6] requirements goals as well as their run-time monitoring.

III. DISASTER MANAGEMENT: UNCERTAINTY IN RAINFALL RUN-OFF MODEL EXAMPLE

Rainfall-runoff (RR) model is a dynamic mathematical model which transforms rainfall to flow at the catchment outlet. The main purpose of the model is to describe rainfall-runoff relations of a catchment area. Common outputs are surface runoff hydrographs which depict relations between discharge (Q) and time (t). Rainfall-runoff models are usually used for predicting behaviour of river discharge and water level and one of the inputs of rainfall-runoff models is the information about weather conditions in the near future. These data are provided by numerical weather prediction models [2, 5]. Common method for estimating precision of the output of Rainfall-Runoff simulation is computation of the Nash-Sutcliffe model efficiency coefficient.

$$E = 1 - \frac{\sum_{t=1}^{T} \left(Q_o^t - Q_m^t\right)^2}{\sum_{t=1}^{T} \left(Q_o^t - \overline{Q_o}\right)^2} \qquad (1)$$

Where

Q0 - is the measured flow in one timestep

Qm- is the simulate flow in one timestep

E – is the Nash-Sutcliffe coefficient – its value range from -inf to -1

E = 1 means that the simulation matches observed data perfectly

E = 0 simulation matches median of the observed data

E < 0 simulation is less precise than the media of the observed data

The precision (E) in the results are completely related with the amount of computation. At design time, it is possible to simulate situations with different trade-offs in terms of quality of services (QOS), execution time, power-consumption that later the RTOS chose the optimal working point depending in the application scenarios.

A. 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 iterations (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 [4] that support the different workload of the system based on the flood emergency situation.

B. 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 discharge volume 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.

IV. Developing Stage Phases

The current implementation of uncertainty modelling is done in C/C++, uses OpenMP and MPI for parallelization and is not in any way enabled for the RTRM environment. As the current implementation of the RTRM environment only supports resource management on one node of the cluster, the use of MPI for parallelization will still be preserved but limited only to one running MPI process. The number of OpenMP threads of this process will then be directly influenced by the number of available computation cores during runtime. The enabling of application source code is done in several iterations to improve our understanding of the RTRM environment and progressively introduce more details to the integration and more types of resources (accelerators, multiple nodes).

Phase 1: In the first phase, the current code for uncertainty modelling will be used without any changes in its structure. The whole module will be wrapped with a wrapper that will integrate it into the RTRM Framework. This wrapper will allow automatic adjustment of input parameters for the uncertainty modelling module according to the currently available resources in the selected AWM.

Phase 2: The second step will be much more coherent integration of the uncertainty modelling module which will lead to restructuring of its code. The goal is to increase granularity of the uncertainty algorithm. This will enable the possibility of adjusting certain input parameters, e.g. number of Monte Carlo iterations during the computation. The result will be a more flexible application which can quickly adapt to the changing runtime environment.

Phase 3: In this phase will be enabling MPI to run several processes of one module on several computing nodes. First, each node will be managed by independent RTRM environments. Later, they should cooperate with each other or only one distributed RT instance will be able to manage several computing nodes.

Phase 4: The last phase we will adapt the uncertainty module for execution on computing nodes with GPU (OpenMP 4.0/OpenCL) and/or MIC (Intel Xeon Phi) accelerators. This can speed up certain parts of our algorithms and provide new area for testing the RTMR environment with accelerators. The main objective is to get an extended set of optimal execution points where accelerators are taken into account.

V. Results in the Target Platform

The platform that serves for development and testing of the integration of the example exposed previously is a 48 core SMP machine HP ProLiant DL785 G6. A Linux version 3.13.0-36-generic with a gcc version 4.8.2 is installed. It is a 8 sockets with Six-Core AMD Opteron Processor 8425 HE 64bits 2.1@GHz [1].

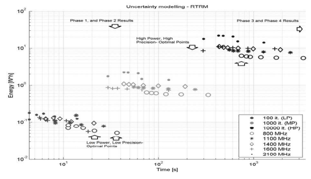

FIGURE 1. UNCERTAINTY EXECUTED WITH RTRM, TIME VERSUS ENERGY CONSUMPTION. MORE ITERATIONS (100, 1000, 1E4) MEANS MORE PRECISION.

TABLE I. MAIN FEATURES OF THE TARGET SMP PLATFORM.

8 sockets	**Six cores, x86-64 architecture**
Working Frequencies	2.1, 1.6, 1.4, 1.1 GHz, 800MHz
Cache L1	64KB Data/ 64KB Instructions
Cache L2	512KB
Cache L3	5118 KB
TLB	1024 KB
DRAM	256 GB DDR2@533 MHz ECC

VI. CONCLUSIONS AND FUTURE WORK

The paper presents an environment that helps to an x86-64 architecture platform to run 24/7 a model saving energy with the trade-offs of losing precision with the different resources configurations. This produces a range in time to execute the model of two orders (Phase 1 and 2). The decision of precision/executed time/consumed energy is taken by a Run-Time Resource Manager monitoring the workload of the platform. At design time, it is possible to describe the optimal points that are going to be used at runtime. The environment also permits to execute the module 24/7 in nodes of a cluster and/or with accelerators (Phase3 and 4). This enables to use more resources and higher precision for the future.

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 the European FP7 projects HARPA id. 612069.

REFERENCES

[1] HP ProLiant DL785 Generation 6 (G6) Quick Specifications/ http://www8.hp.com/h20195/v2/GetDocument.aspx?docname=c042844 72

[2] Martin Golasowski et al., "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 2014.

[3] Patrick Bellasi et al. A rtrm proposal for multi/many-core platforms and reconfigurable applications. In ReCoSoC, 2012.

[4] Antoni Portero, et al. "System and Application Scenarios for Disaster Management Processes, the Rainfall-Runoff Model Case Study." CISIM 2014: 315-326

[5] Matyáš Theuer, et al., "Efficient methods of automatic calibration for rainfall-runoff modelling in the Floreon+ system", submitted to Neural Network World. Accepted for publication 2014.

[6] Antoni Portero,et al.: Methodology for Energy-Flexibility Space Exploration and Mapping of Multimedia Applications to Single-Processor Platform Styles. IEEE Trans. Circuits Syst. Video Techn. 21(8): 1027-1039 (2011)