H2020-FETHPC-1-2014 ANTAREX-671623

AutoTuning and Adaptivity approach for Energy efficient eXascale HPC systems

http://www.ANTAREX-project.eu/

Deliverable D3.3: Initial Runtime Resource and Power Manager

European Commission

Horizon 2020
European Union funding
for Research & Innovation

Deliverable Title:	Initial Runtime Resource and Power Manager		
Lead beneficiary:	CINECA (Italy)		
Keywords:	Functional requirements, extra-functional requirements, use cases		
Author(s):	Daniele Cesarini (ETHZ), Andrea Bartolini (ETHZ)		
Reviewer(s):	João Bispo (UPORTO), Emanuele Vitali (POLIMI), Gianluca Palermo (POLIMI)		
WP:	WP3	**Task:**	T3.3
Nature:	Other	**Dissemination level:**	Public
Identifier:	D3.3	**Version:**	V5
Delivery due date:	February 28th, 2017	**Actual submission date:**	March 15th, 2017

Executive Summary: This deliverable represents the accompanying report of the Deliverable D3.3 (issued at M18) which is the initial release of the Power Manager. It represents the results of the activities carried out by the project partners from M07 to M18 in **Task 3.3 “Runtime Resource and Power Management”** under the leadership of ETHZ.
The Power Manager can be downloaded from the public link:
http://data-archive.ethz.ch/delivery/DeliveryManagerServlet?dps_pid=IE5768287

Approved and issued by the Project Coordinator: C Silvano	**Date:** March 15th, 2017

Project Coordinator: Prof. Dr. Cristina SILVANO – Politecnico di Milano
e-mail: silvano@elet.polimi.it - **Phone:** +39-02-2399-3692- **Fax:** +39-02-2399-3411

Table of Contents

1 Related Work

Our exploration work is focused on power modelling for HPC system and energy-aware MPI runtimes. In this paragraph, we review the state-of-the-art in this field as suggested by the reviewers' recommendation. So far, several works have investigated strategies for increasing the energy-efficiency by the changing power management states of the processing elements such as the Dynamic Voltage and Frequency Scaling.

Spiliopoulos et al. [1] presents "Green Governor" a power governor that optimizes the trade-off between energy consumption and performance of single threaded applications. The main idea behind this work is first, to generate a predictive model suitable for estimating the impact of frequency scaling on delay and energy consumption of a given application based on architectural performance counters. Second, to use this model at run-time to find the optimal frequency that minimizes the EDP or energy consumption at a given performance loss. This is possible by exploring the fact that scaling core's frequency and voltage during stall cycles caused by misses in LLC (memory-bound phases) does not slow down the application. This strategy is implemented in a Linux governor, which is called at regular interval and, each time it is invoked, it collects performance counters. These are used to generate the target frequency to be applied during the next interval.

Predicted delay is calculated as difference between the delay of total stalls in the last period and a model that calculate the delay of stalls caused by LLC misses at different frequency levels. To predict the energy consumption at different frequencies, the authors separate the total energy consumption in two components: i) static power: they use a look-up-table based on off-line measurements of the power consumption during idles state at different frequency levels; ii) dynamic power: the authors generate off-line a linear model which correlates the effective capacitance with the instruction-per-cycles for different applications. This is used at run-time to estimate the energy-consumption of a given application based to different frequency values. When dealing with a multicore processor with a single clock frequency for a set of cores, the authors propose to consider the whole chip a single-core CPU and using the average of the delay and energy prediction to find the optimal frequency selection. Results show that Green Governor can significantly improve the EDP and ED2P (which exceeds the 55% of improvement for memory bound workloads). However, for both these cases severe slowdown may happen for a large set of applications. This is a problem for scientific computing user model in which users pay for the time to solution. In addition, the paper considers only the CPU power and neglects cooling costs, which may induce higher energy penalties for induced delays.

Modern processors can run at higher frequency than the nominal one at run-time if this frequency is power and thermally feasible. This is called TurboMode (TM) for Intel processors.

Lo at al. [2] propose Autoturbo which uses machine learning techniques to predict optimal TM settings avoiding power waist when turbo execution does not result in a consequent performance increase. The main idea of the proposed solution is that TM is not always beneficial and deciding when to enable it, is quite complex. Authors characterize the impact of performance, power consumption, Energy Delay Product (EDP) and Energy Delay Squared Product (ED2P) for a wide set of applications including SPECCPU for CPU and

memory-bound applications, PARSEC benchmarks for multi-threaded applications and websearch workload for latency-critical application. In addition to general metrics the authors use specific metrics for server workload i.e. queries per second over power (QPS/W) and queries per second over cost (QPS/$) that are respectively used to measure the energy efficiency and the cost efficiency for server workloads. Autoturbo is implemented as a python daemon that collects system wide per-core performance counters using an off-line trained classifier to dynamically manage TM activation. The classifier is trained offline based on a set of performance counters such as IPC, L3 loads/misses, TLB misses memory requests, etc. when more than one application is running on the processor. Autoturbo uses a heuristic to determine if the applications are interfering. Autoturbo achieve up to 68% efficiency for ED2P while eliminating the cases where TM leads efficiency drops down to -25%.
As early stated EDP and ED2P metrics do not match the scientific computing user model. For this reason, both these two works are not directly comparable with the proposed ANTAREX Power Manager.

In our approach, instead of adjust DVFS selections (and TM selection) to improve EDP and ED2P of the system, we use it to achieve the best performance under power cap. The key idea is to reward cores with the highest workload with higher frequency than the one applied to the others without violating a power budget.

Focusing on scientific computing, recently have been proposed energy-aware MPI runtimes. Adagio [3] uses DVFS mechanism to reduce the CPU frequency during communication phases. This is done by predicting the duration of each task and consequently adjusting the CPU frequency. A task is identified as a period of computation between two MPI calls. Adagio tasks when iterative computational phases are interspersed with communication phases. However, several HPC applications are not characterized by similar communication patterns and this leads to mispredictions, that cause performance penalty [4].
The ANTAREX power manager approach consists of adjusting the CPU frequency at fixed scheduling points. The power management will combine MPI and application phases collected in the last period to extract core priorities and determining the optimal frequency to be applied to the different cores to maintain the desired power and thermal cap.

When focusing on large scale systems, few works have investigated DVFS mechanisms. Authors of [5], [6], [7], and [8] have proposed DVFS strategies for latency-critical systems in order to improve EDP and ED2P without violating service level objective (SLO). This workload model is typical for cloud service workloads but is significantly different from HPC scientific computing workload model. In ANTAREX, we do not consider latency-critical workload where real-time constraints bound application results. The HPC machine runs a single large workload spread over a large number of computational node interconnected with a low-latency high-bandwidth network. Warehouse-scale computer (WSC) systems are based on commodity hardware, which do not required extreme high-performance nodes and interconnection. Typical workload of HPC applications is largely composed of floating-point operations and requires execution time in the order of seconds to days. Instead, WSC workloads such as web searching and massive key-value stores, require high throughput for latency-sensitive jobs, which must be performed in millisecond time.

References:

[1] Spiliopoulos, Vasileios, Stefanos Kaxiras, and Georgios Keramidas. "Green governors: A framework for continuously adaptive dvfs." Green Computing Conference and Workshops (IGCC), 2011 International. IEEE, 2011.

[2] Lo, David, and Christos Kozyrakis. "Dynamic management of TurboMode in modern multi-core chips." High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on. IEEE, 2014.

[3] Rountree, Barry, et al. "Adagio: making DVS practical for complex HPC applications." Proceedings of the 23rd international conference on Supercomputing. ACM, 2009.

[4] Kerbyson, Darren J., Abhinav Vishnu, and Kevin J. Barker. "Energy templates: Exploiting application information to save energy." Cluster Computing (CLUSTER), 2011 IEEE International Conference on. IEEE, 2011.

[5] Lo, David, et al. "Towards energy proportionality for large-scale latency-critical workloads." ACM SIGARCH Computer Architecture News. Vol. 42. No. 3. IEEE Press, 2014.

[6] Hsu, Chang-Hong, et al. "Adrenaline: Pinpointing and reining in tail queries with quick voltage boosting." High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on. IEEE, 2015.

[7] Lo, David, et al. "Heracles: improving resource efficiency at scale." ACM SIGARCH Computer Architecture News. Vol. 43. No. 3. ACM, 2015.

[8] Kasture, Harshad, et al. "Rubik: Fast analytical power management for latency-critical systems." Proceedings of the 48th International Symposium on Microarchitecture. ACM, 2015.

2 Power Manager: Initial Implementation

2.1 *Intel P-States and C-States*

The power management of Intel processors defines P-States and C-States to manage active and idle power states, respectively.

The P-States include dynamic voltage and frequency (DVFS) operating points which target the reduction of active power. C-States instead target idle power reduction strategies. Both P-States and C-States are numbered from 0 to n. Higher number means higher power saving. However, in case of C-States this means also longer transitions in and out of the state itself. In recent Intel architectures, P-States can be selected independently for each core. In case of C-States, they are defined independently for cores and the "uncore" region. Uncore region is composed to all the components of the CPU that are not cores, this means I/O, HW controllers, LLC, interconnection ring, etc. In the Linux O.S., P-States are handled by means of SW frequency governors. C-States are triggered from the firmware of the CPU but the OS can provide hints on the appropriate C-State to the hardware through O.S. idle governors. Table 1 shows the different architecture impacts and dependencies for the different cores and package C-States. The red zones are invalid combinations of core and package C-States.

		Core C-States				
		C0	C1	C3	C6	
Package C-States	C0	🟩	🟩	🟩	🟩	Active State
	C1E	🟥	🟩	🟩	🟩	Lower P-State
	C2	🟥	🟥	🟩	🟩	Only L3 Snoop
	C3	🟥	🟥	🟩	🟩	Flush L3 - Off
	C6	🟥	🟥	🟥	🟩	Low Voltage
		Active State	Clock Gated	Flush L1,L2 Off	Power Gated	

Table 1 Matrix of core/package C-States with dependencies and impacts

2.2 *Component description*

ANTAREX power manager implements power capping strategies that select the best performance point for each core to maintain a power constraint. Unlike state-of-the-art power capping mechanism such as Intel RAPL, ANTAREX power manager uses information about the application and the heterogeneity of the workload to prioritise the performance of a given core under a power cap. Each core is assigned a priority selected by the user, and priorities are used to take a decision about what frequencies should be assigned to cores in order to maintain the power constraint.
ANTAREX power manager works at the node level and it is based on a time-delay loop. At every step of the loop, the power manager solves an optimization problem, which is based on the application usage of the resources on the previous step. The optimization solver returns a set of frequencies, to be assigned to the cores in order to optimize the computation under the power constraint.
ANTAREX power manager has been developed for Linux systems based on the Intel Xeon Processor v3 or above which allows independent frequency per core.

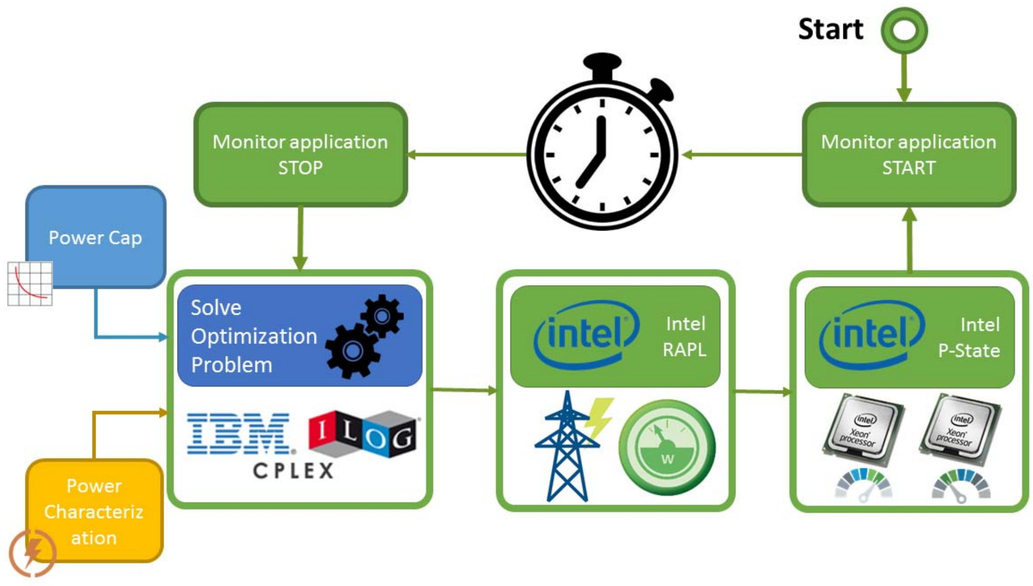

Figure 1 Module composition and interaction of power manager

The ANTAREX Power Manager is shown in Figure 1 and it composed by four modules:

1. **Monitor application module**: This component monitors the hardware level to extract all the system information (frequency, load, power consumption, idle time, etc.). It uses hardware performance counters equipped on Intel Xeon v3 processors and accessible from the O.S. through Machine Specific Registers (MSR). The monitor accesses to performance counters through the MSR module of Linux Kernel. Future implementations will use the collector block implemented in T3.1.
2. **Optimization module**: this component is responsible to build the optimization model and to call the optimization library to solve the problem. We use IBM Ilog CPLEX as the solver. The information used by the optimization model is:
    a. *System information*: about the status of the machine in the last period. The information used in the optimization model is: percentage of Core C-State C0, percentage of Core C-State C1, percentage of Package C-State C0, percentage of Package C-State C2 and idle power consumption of the sockets.
    b. *Core priorities*: these parameters are given by the user before starting the application to identify the priority of each core in the system.
    c. *Power cap*: this is the maximum power consumption in Watts for all the sockets given by the user.
    d. *Predictive power model*: consists of 5 parameters for each socket in the system. These parameters have been extracted through a linear regression on power consumptions of the node. These coefficients are used to predict the power consumption of the system.

3. **Intel RAPL module**: this module relies on the Intel RAPL system to cap the power consumption in case of bad prediction. Intel RAPL is a hardware technology of Intel CPUs that acts on the P-States of cores, overriding the requested P-States of the O.S. in case of excessive of power consumption. This module, with the aid of RAPL, assures that the total power consumption is always under the power cap.
4. **Intel P-State module**: this module is responsible to receive the frequencies calculated from the optimization module and applying them to the cores. As the name indicates, it uses Intel P-state MSRs to force cores to work at selected frequencies.

3 The Optimization Problem

The optimization model used to solve the frequency allocation is based on an integer linear programming model (ILP). We use IBM Ilog CPLEX to solve this problem.

3.1 Description

The model use coefficients given by system information through performance counters, architectural information and user parameters. In detail, coefficients used in the model are:

S : number of sockets $(s = 1, \dots, S)$
C : number of cores for socket $(c = 1, \dots, C)$
F : number of frequency levels $(f = 1, \dots, F)$
δ_{sc} : priority of core c on socket s
γ_f : Frequency value of level f
$CC0_{sc}$: Percentage of Core C-State 0 of cores c of socket s and Core C0 X coefficient of socket s
$CC1_{sc}$: Percentage of Core C-State 1 of cores c of socket s and Core C1 X coefficient of socket s
$CC6_{sc}$: Percentage of Core C-State 6 of cores c of socket s and Core C6 X coefficient of socket s
$PC0_s$: Percentage of Package C-State 0 of socket s and Package C-State 0 X coefficient of socket s
$PC2_s$: Percentage of Package C-State 2 of socket s and Package C-State 2 X coefficient of socket s
$PC6_s$: Percentage of Package C-State 6 of socket s and Package C-State 6 X coefficient of socket s
X_PC6_s : X coefficient of Package C-State 6 of socket s
P_{MAX} : Value in Watts for power capping

CCX, *PCX* and *X_PC6* are used to calculate the power consumption in the power constraint.

The model works on the following variables to find out the best solution:

$$x_{scf} = \begin{cases} 1 & \text{if core } c \text{ of socket } s \text{ works at frequency } f, \\ 0 & \text{otherwise.} \end{cases}$$

$$Xfmax_{sf} = \begin{cases} 1 & \text{if socket } s \text{ has maximum frequency } f, \\ 0 & \text{otherwise.} \end{cases}$$

The model maximizes the following object function that tries to find the higher core frequencies proportionally to the priority assigned by cores under a power cap:

$$O.F. = max \sum_{s=1}^{S} \sum_{c=1}^{C} \sum_{f=1}^{F} \delta_{sc} \gamma_f x_{scf}$$

To maintain the model coherent with the architectural information, we explicit the following constraints which allows selecting only a frequency level for the core.

$$\sum_{f=1}^{F} x_{scf} = 1$$

$$(c = 1, \dots, C)(s = 1, \dots, S)$$

The second constraint force the *Xfmax* variable to be true only for a frequency level.

$$\sum_{f=1}^{F} Xfmax_{sf} = 1$$
$$(s = 1, \ldots, S)$$

The third constraint finds the maximum frequency among all the cores of a socket. This constraint is used to calculate the power consumption of uncore region using the *Xfmax* variable.

$$\sum_{f=1}^{F} \gamma_f x_{scf} <= \sum_{f=1}^{F} \gamma_f Xfmax_{sf}$$
$$(s = 1, \ldots, S)(c = 1, \ldots, C)$$

The last constraint is the power cap. This constraint maintains the power cap for all possible solutions. This constraint calculates the total power consumption when different levels of frequencies are selected. *CC0* coefficients are multiplied for selected frequencies to discover the impact of the core during active states. *PC0* is multiplied by the *Xfmax* variable, which represents the maximum core frequency for that socket, to calculate the power consumption of the uncore component. The other coefficients represent the impact of idle states in the total power consumption.

$$\sum_{s=1}^{S} \left(\sum_{c=1}^{C} \left(\sum_{f=1}^{F} CC0_{sc} \gamma_f x_{scf} + CC1_{sc} \right) + \sum_{f=1}^{F} PC0_s \gamma_f Xfmax_{sf} + PC2_s + X\_PC6_s \right) \leq P_{MAX}$$

Moreover, we are currently developing an optimized version of the above model that takes into account groups of cores with the same priority instead of single cores. This way, we can reduce the number of variables and constraints. The difference between models is only in the number of variables, coefficients and constraints remain unmodified.

4 How to install

The ANTAREX power manager is contained in a single Python script. Python 2.7 must be installed in the system for a correct execution. The script must be invoked with an input text file, which contains all the parameters needed for the execution. During run-time, the script prints in the standard output some information about the evolution of its execution. An output file created in the same directory of the script logs all the information regarding performance counters and solution parameters of the optimization problem.

The system must have installed IBM Ilog CPLEX in a version equal or higher 12.0. The script get access to the solver through Python APIs, for this reason CPLEX Python APIs must be properly installed in the system.
The ANTAREX power manager should have the possibility to change the core frequencies independently of the power manager of the system. To allow this, CPUFREQ must be installed in the machine, Intel P-state driver is not supported. The script automatically configures CPUFREQ to avoid interference during the execution. At the end of the run, the script restores the original configuration of CPUFREQ. The ANTAREX power manager interacts with Linux kernel modules, for this reason must be executed with root privileges.

The ANTAREX power manager automatically checks if all dependencies are respected and starts its execution only if all the configurations are properly applied. The script can be stopped sending a signal INT to the process, this can be done using the “Ctrl^c” control key if the script is attached to a shell otherwise can be sent an “INT” signal using “kill” command.

Content from README

This README contains dependencies, configurations and a usage example of the ANTAREX power manager. The following instructions are targeted for Red Hat based distribution, in particular it has been tested on Centos 7.1.

Download

The ANTAREX Power manager can be downloaded from:
http://data-archive.ethz.ch/delivery/DeliveryManagerServlet?dps_pid=IE5768287

Dependencies

It requires the following dependencies to work:

- *CPUFREQ*: By default, on Intel systems, the default driver for power management is “intel_pstate”. To replace intel_pstate with CPUFREQ, the system must be started passing at boot time a parameter to the Linux kernel to disable intel_pstate, this parameter is: “intel_pstate=disable”. At next boot, the power manger will be automatically replaced with CPUFREQ, which is the standard power manager for Linux system.
- *MSR kernel module*: usually MSR module is loaded by default on Intel system, if not:

```
$> sudo modprobe msr
```

- *Python 2.7*: usually Python 2.7 runtime is installed by default in the system, if not:

```
$> sudo yum install -y python27

$> python -V

Python 2.7.3
```

- *IBM Ilog CPLEX*:
    - *Register in the IBM web site*: IBM Ilog CPLEX is free of charge for all students, researchers, professors, etc. that work in academia through the IBM Academic Initiative program. https://www-304.ibm.com/ibm/university/academic/pub/jsps/assetredirector.jsp?asset_id=1070
    - *Download software*: when you have been accepted to the IBM Academic Initiative program can you download the software through the software download page.
    - *CPLEX Installation*: The installer for IBM ILOG CPLEX Optimization Studio is distributed as a .bin file cplex_studioXXX.linux-x86.bin. This installer can be used on both 32-bit and 64-bit Linux platforms. .bin files are Linux self-extracting files. Once you download the installer, follow the steps below:
        - Make sure the .bin file is executable. If necessary, change its permission using the chmod command from the directory where the .bin is located:
          ```
          $> chmod +x cplex_studioXXX.linux-x86.bin
          ```
        - Enter the following command to start the installation process:
          ```
          $> ./cplex_studioXXX.linux-x86.bin
          ```
    - *Python APIs installation:* when CPLEX has been installed in the system should be installed the Python APIs. To accomplish this, go to the installation directory of CPLEX and execute the python script, by default:
      ```
      $> cd /opt/ibm/ILOG/CPLEX_Studio_Community127/cplex/python/2.7/x86-64_linux
      $> sudo python setup.py install
      ```
      More info: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.5.0/ilog.odms.cplex.help/CPLEX/GettingStarted/topics/set_up/Python_setup.html

Configuration

The "input.txt" file in the script directory contains the default parameters needed for execution. The parameters are:

- *sleep_time*: this parameter identifies the interval time in seconds of the time-delay loop. At every iteration, the optimization problem is solved and new frequencies are applied to the core.
- *time_limit_cplex*: this parameter limits the computation time of CPLEX to solve the problem (in seconds).
- *power_cap*: this is the power cap of the machine (in Watts).
- *job_priority*: this parameter identifies an array of job priorities where each job ID identifies the position of the array and the priority to the value of the array.
- *job_mapping*: this parameter is an array that represent the job-to-core pinning. The job ID identifies the position in the array, while the core ID identifies the number contained in the array.
- *K_socket_X*: these parameters identify the coefficients of power consumption for each socket.

Usage

Run the script before launching your application:

```
• $> sudo python power_capper.py -i input.txt
```

```
*******************************************
*******************************************
Time to solve the optimization problem: 1.270 sec
Socket 0 - Core 0: 2400 MHz
Socket 0 - Core 1: 2400 MHz
Socket 0 - Core 2: 2400 MHz
Socket 0 - Core 3: 2400 MHz
Socket 0 - Core 4: 2400 MHz
Socket 0 - Core 5: 2400 MHz
Socket 0 - Core 6: 1400 MHz
Socket 0 - Core 7: 1200 MHz
Socket 1 - Core 0: 1200 MHz
Socket 1 - Core 1: 1200 MHz
Socket 1 - Core 2: 1200 MHz
Socket 1 - Core 3: 1200 MHz
Socket 1 - Core 4: 1200 MHz
Socket 1 - Core 5: 1200 MHz
Socket 1 - Core 6: 1200 MHz
Socket 1 - Core 7: 1200 MHz
REAL POWER - Previous step
socket 0: 49.41 W
socket 1: 28.50 W
Total: 77.91 W
PREDICTED POWER - Next step
socket 0: 50.29 W
socket 1: 29.68 W
Total: 79.96 W
Overhead time: 1.301 sec
*******************************************
*******************************************
```

To stop the execution, type “Ctrl^c” on the shell attached to the script, or:

```
$> kill -SIGINT $PID
```

At the end of execution, an output file called “output.csv” located in the same directory of the script contains all the information of the system and on the optimization problems. The output file is organized as a standard CSV file. All the parameters on a single run are in lined and separated with “;”. The syntax and the field of each parameter is described from the script using the invoking with parameter “-s”.

```
$> sudo python power_capper.py -s
```

5 Ongoing work

This release of the ANTAREX Power Manager is the initial prototype and it provides only limited functionalities. Next activities will be focused on:

- **Priority Generation:** Generation of the per core priority based on application profiling and integration with the *mARGOt* framework.
- **Power and Thermal Modelling**: Algorithm to extract the power and thermal model from the controlled device.
- **Comparison with HW power capping**: Study of the benefit of the proposed solution w.r.t. HW power capping strategies.