DOI: 10.14529/jsfi140202

Predicting the Energy and Power Consumption of Strong and Weak Scaling HPC Applications

Hayk Shoukourian[1,2,*] **, *Torsten Wilde***[1]**, *Axel Auweter***[1]**, *Arndt Bode***[1,2]

Keeping energy costs in budget and operating within available capacities of power distribution and cooling systems is becoming an important requirement for High Performance Computing (HPC) data centers. It is even more important when considering the estimated power requirements for Exascale computing. Power and energy capping are two of emerging techniques aimed towards controlling and efficient budgeting of power and energy consumption within the data center. Implementation of both techniques requires a knowledge of, potentially unknown, power and energy consumption data of the given parallel HPC applications for different numbers of compute servers (nodes).

This paper introduces an Adaptive Energy and Power Consumption Prediction (AEPCP) model capable of predicting the power and energy consumption of parallel HPC applications for different number of compute nodes. The suggested model is application specific and describes the behavior of power and energy with respect to the number of utilized compute nodes, taking as an input the available history power/energy data of an application. It provides a generic solution that can be used for each application but it produces an application specific result. The *AEPCP* model allows for ahead of time power and energy consumption prediction and adapts with each additional execution of the application improving the associated prediction accuracy. The model does not require any application code instrumentation and does not introduce any application performance degradation. Thus it is a *high level* application energy and power consumption prediction model. The validity and the applicability of the suggested *AEPCP* model is shown in this paper through the empirical results achieved using two application-benchmarks on the SuperMUC HPC system (the 10^{th} fastest supercomputer in the world, according to Top500 November 2013 rankings) deployed at Leibniz Supercomputing Centre.

Keywords: adaptive prediction, energy consumption, power consumption, energy capping, power capping, AEPCP model, energy measurement, node scaling, EtS prediction, HPC.

Introduction

With the ever increasing growth of applications requiring a scalable, reliable, and low cost access to high-end computing, many modern data centers have grown larger and denser making power consumption a dominating factor for the Total Cost of Ownership (TCO) of supercomputing sites [18, 19]. This increase in power consumption not only converts into high operating costs, but also to high carbon footprint which affects the environmental sustainability, as well as straining the capacity limits of current data center's power delivery and cooling infrastructures. All these make a well-defined and efficient power management process a necessity for achieving a sustainable and cost-effective High Performance Computing (HPC) data center. Power and energy capping are two of the emerging techniques for controlling power and energy consumption in a data center [7].

Power capping limits the amount of power a system can consume when executing various applications, thus aiming to keep the system usage within a given power limit and prevent possible power overloads. Power capping covers a wide range of use cases: from limited power

[1]Leibniz Supercomputing Centre (LRZ) of the Bavarian Academy of Sciences and Humanities
Boltzmannstraße 1, 85748 Garching bei München, Germany
[2]Technische Universität München (TUM), Fakultät für Informatik I10
Boltzmannstraße 3, 85748 Garching bei München, Germany
[*]Corresponding author. Email address: hayk.shoukourian@{lrz.de; in.tum.de}
Tel.: +49 89 35831 8772; Fax: +49 89 35831 8572

deliveries and/or limited cooling capacities; through the handling of power exceptions (e.g. unexpected peaks in system utilization); to power budgeting and mitigation of 'power-hungry' or malicious applications capable of generating dangerous power surges. Two interesting possible scenarios for power capping in a HPC data center are: avoiding runtime power peak, which can be addressed by new CPU features, such as setting a hardware power bound [27]; and temporary power constraints due to infrastructure maintenance (as illustrated in Figure 1). Figure 1 shows the average power consumption behavior (blue solid line) of a given HPC system cooled with the use of the data center's cooling towers depicted on the top of the image.

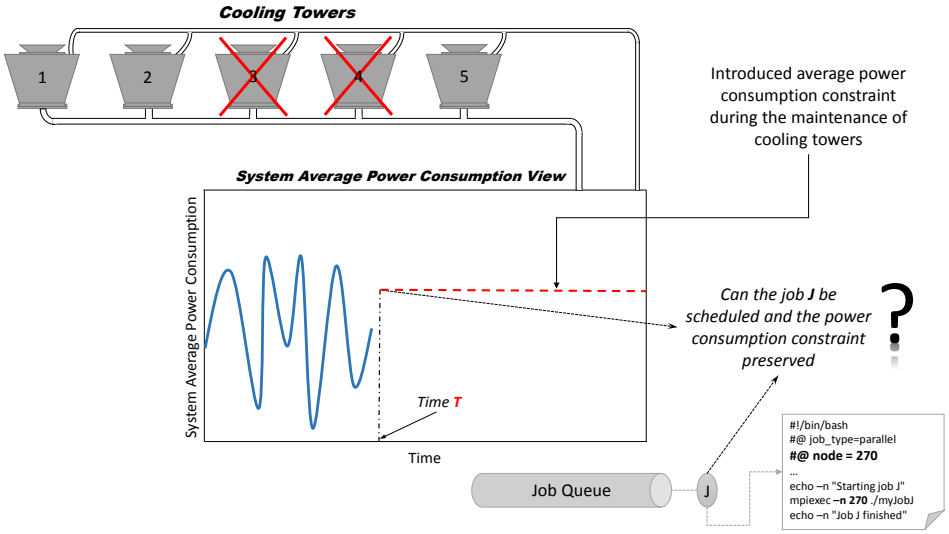


Figure 1. Power Capping use case scenario

Assume that at time T two of the data center's cooling towers are in maintenance, introducing a temporary average power consumption constraint for the system. Now, assume that there is a queued job (application) J with a utilization requirement of 270 compute nodes/servers, which needs to be scheduled for execution. In order to determine whether the execution of job J is possible within the introduced average power consumption constraint, the information on the potential power consumption of job J with 270 compute nodes is required. Without this information the scheduling of job J could overload the available cooling capacity.

While power capping is useful, the majority of current techniques (e.g. [8, 13]) that implement power capping involve dynamic voltage frequency scaling [15], that will, in most cases, increase the runtime of the application [15], thus increasing the integral of power consumption over time (energy). *Energy capping* is another management technique that limits the amount of energy a system can consume when executing applications for a given time period. In other words, energy capping limits the integral amount of power consumption over time and, in contrast to power capping, it does not limit the amount of power the system can consume at a given point in time. From a data center perspective, energy capping is currently a more important approach, since energy consumption equals costs. The knowledge on application potential energy consumption for a given number of compute nodes will allow for a power-cost optimization by shifting low priority applications with higher energy/power consumption rates to off-peak hours, when the cost of electrical power is cheaper. This knowledge will also allow for energy-driven charging policies as an alternative to currently existing CPU-hour based charging policies.

A typical use case scenario of energy capping is illustrated in Figure 2. The dashed red line in Figure 2 shows the introduced per month allocated energy budget that a system can consume

on a monthly basis (this can be for the whole system or on a per user/customer basis), whereas the blue solid line shows the ongoing energy consumption.

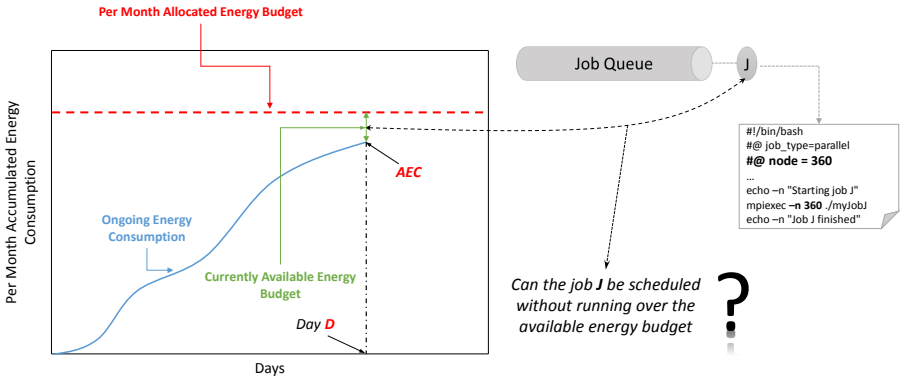


Figure 2. Energy Capping use case scenario

Assume that on day D the system has already an Accumulated Energy Consumption (AEC, Figure 2) of a given amount. Assume further, that there is a pending job J, with requested 360 compute nodes. In order to understand whether the job J can still be scheduled for execution within the available energy budget, the resource management system has to have the information on the potential energy consumption of the job J with 360 compute nodes.

Though power and energy capping for these use case scenarios (as described for Figure 1, Figure 2) solve different problems, they both require the same knowledge of, potentially unknown, power and energy consumption profiles of applications to be executed. Without the access to this knowledge, the implementation of these techniques will be incomplete. This paper proposes an *Adaptive Energy and Power Consumption Prediction* (AEPCP) model capable of predicting the Energy-to-Solution (EtS) [4, 22] and the Average Power Consumption (APC) [37] metrics for any parallel HPC applications with respect to the given number of compute nodes. The $AEPCP$ model requires unique identifiers for each application and takes the *available* application historical power/energy data as an input. It is worth noting that this data is, in the most cases, already available in the current data center energy/power monitoring and resource management tools. The application can behave differently with different input data sets or if some system settings are changed (e.g. system dynamic voltage and frequency scaling governor configurations). Therefor, each substantial change needs to be treated as a different application and requires a new unique identifier. The model is validated for *strong scaling* applications (i.e. applications with fixed input problem size) as well as for *weak scaling* applications (i.e. applications with adjusted input problem size).

The remainder of this paper is structured as follows. Section 1 gives some background information on application scalability. Section 2 provides a survey on related works. Section 3 illustrates the prediction process and introduces the $AEPCP$ model. Section 4 describes the application-benchmarks as well as the compute system which were used to validate the suggested model. Section 5 presents the EtS results for application strong and weak scaling scenarios. Section 6 shows the APC prediction results, and discusses the benefits of $AEPCP$ based APC prediction as compared to the usage of vendor provided maximum power boundaries of system compute nodes. Section 7 looks at the future $AEPCP$ model enhancement directions, and finally Section 8 concludes the paper.

1. Background

The scalability of a parallel HPC application shows the relation between application execution time and the number of application utilized compute resources, e.g. nodes. Scaling is referred to as **strong** when an application input problem size (i.e. the amount of required computation) stays constant independently from the number of compute nodes which are utilized to solve that problem. This implies that an application demonstrating a strong scaling will have a smaller execution time, i.e. will solve the computation faster, as the number of compute nodes increase.

Scaling is referred to as **weak** when the input problem size of the application is fixed for each utilized compute node. This indicates that the execution time of an application under weak scaling will show a constant behavior since the input problem size increases accordingly with the number of utilized compute nodes. Figure 3 shows the execution-time, i.e. Time-to-Solution (TtS), behavior for strong and weak scaling scenarios.

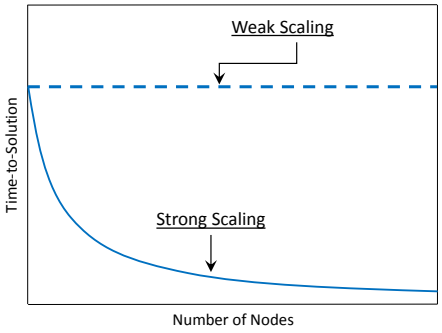


Figure 3. Theoretical TtS curves for strong and weak scaling scenarios

The limits of theoretically possible speedups achieved by parallel HPC applications in the case of strong and weak scaling and the outline of the theoretical boundaries of APC and EtS metrics under compute node scaling are presented in Subsection 1.1 and Subsection 1.2. The following denotations and definitions are used throughout these subsections:

- $t_s(n)$ - processing time of the application serial part using n nodes;
- $t_p(n)$ - processing time of the application parallel part using n nodes;
- $T(1) = t_s(1) + t_p(1)$ - processing time of the application sequential and parallel parts using 1 node;
- $T(n) = t_s(1) + t_p(n)$ - processing time of the application sequential and parallel parts using n nodes;
- $p = \frac{t_p(1)}{t_s(1)+t_p(1)}$ - the **non-scaled** fraction of the application **parallel** part [29], i.e. the parallel portion of computation on a **sequential** system ($0 \leq p \leq 1$). Thus the **non-scaled** fraction of the application **sequential** part will be $(1 - p)$;
- $p^* = \frac{t_p(n)}{t_s(1)+t_p(n)}$ - the **scaled** fraction of the application **parallel** part [29], i.e. the parallel portion of computation on a **parallel** system ($0 \leq p^* \leq 1$). Thus the **scaled** fraction of the application **sequential** part will be $(1 - p^*)$.

1.1. Strong Scaling - Amdahl's Law

Strong scaling was first described analytically by Gene Amdahl in 1967 [1]. According to Amdahl's law, the possible speedup that a parallel application can achieve using n ($n \geq 1$) compute nodes is:

$$Speedup(n) = \frac{T(1)}{T(n)} = \frac{1}{(1-p) + \frac{p}{n}} \tag{1}$$

The total $T(n)$ processing time of sequential and parallel parts using n compute nodes, according to Amdahl's law (Equation 1), can be derived as:

$$T(n) = T(1) \cdot [(1-p) + \frac{p}{n}] \tag{2}$$

A study by Woo and Lee [37], considering Amdahl's law, proposes an analytical model for calculating the average power consumption $P(n)$ of a given application when executed on n compute nodes.

$$P(n) = \frac{1 + (n-1) \cdot k \cdot (1-p)}{(1-p) + \frac{p}{n}} \tag{3}$$

where k is the fraction of power that is consumed by the compute node in idle state ($0 \leq k \leq 1$). This further means that when an application demonstrates *ideal scalability*[1], then $P(n) = n$, as illustrated in Figure 4 (dashed yellow line). While when an application demonstrates *no scalability*[2], $P(n) = 1 + (n-1) \cdot k$ (solid yellow line in Figure 4).

Having Equation 2 and Equation 3, the EtS $E(n)$ of a given application can be derived as follows:

$$E(n) = T(n) \cdot P(n) = T(1) \cdot [1 + (n-1) \cdot k \cdot (1-p)] = \mathrm{O}(n) \tag{4}$$

which means that in the case of an application demonstrating *ideal scalability*, the EtS behavior for that application will be constant with respect to the number n of utilized compute nodes. Whereas, in the case of an application with *no scalability*, the corresponding EtS behavior will be linear. The dashed and solid red lines in Figure 5 illustrate these scenarios. This further means, that the realistic EtS behavior of applications must be in between these constant and linear boundary lines.

1.2. Weak Scaling - Gustafson's Law

The speedup of applications demonstrating a week scaling was first described analytically by John L. Gustafson [11] as:

$$Speedup(n) = \frac{T(1)}{T(n)} = 1 + (n-1) \cdot p^* \tag{5}$$

Following the same observation proposed in [37], we can state that it takes $t_s(1)$ to execute the sequential portion of the computation and it takes $t_p(n)$ to execute the parallel portion of

[1] In other words the strictly serial $(1-p)$ fraction of the application is 0.
[2] In other words the computation fraction p that can be parallelized is 0.

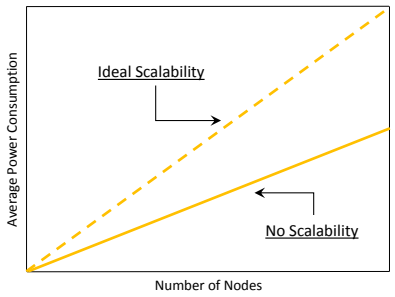


Figure 4. Theoretical APC curves for ideal and no scalability cases for **strong and weak** scaling scenario

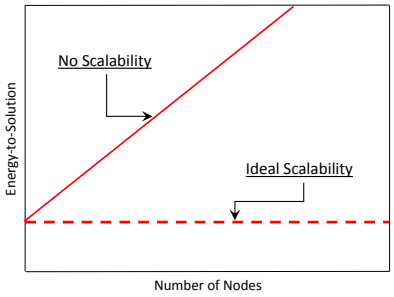


Figure 5. Theoretical EtS curves for ideal and no scalability cases for **strong** scaling scenario

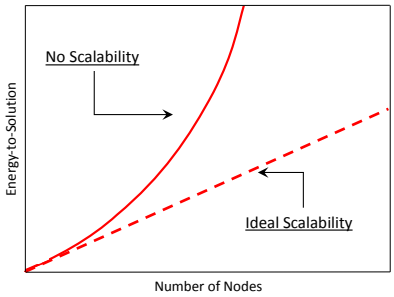


Figure 6. Theoretical EtS curves for ideal and no scalability cases for **weak** scaling scenario

the computation. Assuming that the fraction of power that is consumed by the compute node in idle state is k $(0 \leq k \leq 1)$, the average power consumption $P(n)$ with respect to the number of utilized compute nodes can be written as:

$$P(n) = \frac{t_s(1) \cdot [1 + (n-1) \cdot k] + t_p(n) \cdot n}{t_s(1) + t_p(n)} = (1 - p^*) \cdot [1 + (n-1) \cdot k] + p^* \cdot n = \\ 1 + p^* \cdot (n-1) + (1 - p^*) \cdot (n-1) \cdot k = \mathrm{O}(n) \tag{6}$$

This further means that in the case of an application that shows *ideal scalability*[3], the average power consumption $P(n)$ is: $P(n) = (n-1) + 1 = n$ (Figure 4). Since the execution time in the case of ideal scalability remains constant as the input problem sizes increases in parallel with the number of compute nodes, we can further state that the EtS behavior of the application $E(n)$ with respect to the given n number of compute nodes is of a linear order: $E(n) = P(n) \cdot TtS(n) = n \cdot \mathrm{O}(1) = \mathrm{O}(n)$. The dashed red line in Figure 6 depicts this scenario.

In the case of an application that shows *no scalability*[4], the average power consumption $P(n)$ (from Equation 6) is: $P(n) = 1 + (n-1) \cdot k$ (Figure 4). Since the execution time of an application in the case of no scalability increases linearly with the input problem size and the number of compute nodes, the EtS $E(n)$, in the case of no scalability, will show a quadratic behavior with respect to the number of compute nodes n: $E(n) = P(n) \cdot TtS(n) = [1 + (n-1) \cdot k] \cdot \mathrm{O}(n) = \mathrm{O}(n^2)$ (solid red line in Figure 6).

As can be deducted from the above discussion, the average power consumption of an application, for both strong and weak scaling applications, is the highest when it demonstrates ideal scalability. Therefor, an artificial hardware power cap [27] might keep an application from providing the highest performance and could increase the overall TtS, and subsequently EtS as well.

Although, it was possible to derive the analytical EtS $E(n)$ and APC $P(n)$ boundary curves for strong and weak scaling applications with respect to the given n number of compute nodes, the knowledge of an application's non-scaled p (in case of strong scaling) or scaled p^* (in case of weak scaling) fractions (which are application specific information) is necessary in order to estimate the energy/power consumption for a given n number of compute nodes. The obtainment

[3] In other words the scaled fraction of the application sequential part $1 - p^*$ is 0.
[4] In other words the scaled fraction of the application parallel part p^* is 0.

of this application specific information is not trivial, and might be even impossible, in real-world scenarios where myriad of different HPC applications are run in a HPC data center.

2. Related Work

An approach aimed towards performance prediction is described by Ipek et al. in [16]. The authors introduce a similar, adaptive model for predicting the TtS of parallel applications with respect to the input problem size of the application but with a fixed number of compute nodes. Even though, it could be possible to derive the energy consumption of an application using the corresponding knowledge of TtS and vendor provided maximum thermal design power [14] of a system compute node, this approach will not be applicable for our use case of energy and power capping, since it does not provide a knowledge on TtS behavior with respect to different numbers of compute nodes.

A study directed towards cross platform energy usage estimation of individual applications is found in [6]. The authors suggest a model capable of predicting the energy consumption of a given application during the application's execution phase. This model is not applicable for implementing energy/power capping techniques since it does not provide information on energy/power consumption of a given application in advance, which is required by the system resource manager for scheduling applications and preserving the predefined system energy/power consumption constraints.

Another set of approaches focused on predicting the energy consumption of applications using analytic models is found in [12] and in [5]. These approaches focus on predicting the power consumption of a given application with respect to a given CPU frequency. They both require knowledge of either the application (e.g. scaling properties) and/or the platform characteristics for different CPU frequencies. Both models are not yet extended/validated for multi-node compute systems and are analytic predictive models, which usually do not completely capture the interactions between underlying architecture and running software, and often require additional manual tuning [16].

A technique aimed towards controlling power consumption is found in [13]. It proposes a model, called "Pack & Cap", that adaptively manages the number of cores and CPU frequency depending on the given application characteristics, in order to meet the user-defined power constraints. "Pack & Cap" model is not applicable for the HPC domain, because, *first*, "Pack & Cap" model was validated on a single, quad core server node, and, as authors mention, the suggested technique is not yet extended/validated for large scale computing systems. *Second*, it needs a large volume of application performance data to conduct power/energy capping, which could not be available in real world scenarios. *Third*, it does not predict the power/energy consumption of applications. *In the end*, the model is targeted specifically for virtual machines, and might not therefor be easily adapted for HPC systems.

Another set of works focused on application energy/power consumption prediction, given application in-depth characteristics, is found in [24] and in [32]. [24] presents an energy consumption prediction model requiring application tracing (information on floating point operation count, memory operation count, etc.) and information on the energy profile of the target compute system (e.g. average energy cost per fundamental operation), obtained through the use of several special benchmarks. Although the suggested model could be used for a cross platform application energy consumption prediction, if the required energy profile data (e.g. achievable memory bandwidths for each level of the memory hierarchy) of the target system is available,

their method involves application code instrumentation and attempts to split the application into "basic blocks" [24]. This would require a lot of effort when dealing with several hundred different applications, which is typically the case for modern HPC data centers. [32] suggests a quasi-analytical model, which combines the application analytic description (achieved through extensive application analysis) with the compute platform parameters (such as per-core power consumption of a computation unit, and power consumption during inter-processor communication) obtained through experimental benchmarks. While useful, the validation of a model was shown using a single benchmark and the suggested method requires a thorough analysis of the given application, which could be impractical in real-world scenarios, when several applications with different characteristics are queued for execution.

In summary, none of the aforecited models predicts the application energy/power consumption with respect to the number of compute nodes, and thus none of them can be applied for implementing power and energy capping techniques for our use case on large scale computing systems.

3. Framework

This section introduces the Adaptive Energy and Power Consumption Prediction ($AEPCP$) process, the $AEPCP$ model, and the monitoring tool which was used to obtain the application profile data.

3.1. The AEPCP Process

The prediction process of the approach suggested in this paper is outlined in Figure 7. The $AEPCP$ process has two inputs: the application identifier, which is used to uniquely identify an application, and the number of system compute resources (e.g. CPU, compute nodes, accelerators, etc.), which are planned to be utilized by a given application. The application identifier is used to query the application relevant history information from the system monitoring tool (step (1), Figure 7).

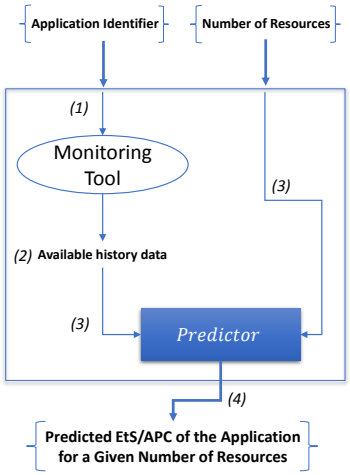


Figure 7. Overview of the $AEPCP$ process

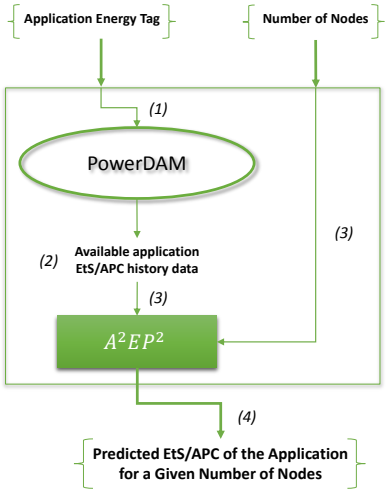


Figure 8. Overview of the $AEPCP$ model

This application-relevant history profile data (step 2), together with the number of compute resources, is passed to the predictor (step 3) for corresponding EtS/APC prediction. Using this

data, the predictor then reports the predicted EtS/APC value for the application with respect to the given node number (step 4).

3.2. The AEPCP Model

Figure 8 presents the overview of the $AEPCP$ model based on the prediction process described above. The $AEPCP$ model takes as input: *(i)* the application *energy tag* as an application unique identifier, which is supported by the IBM LoadLeveler [17] resource management system and is specified by the user on a unique-per-application basis; and *(ii)* the number of compute nodes as compute resource number (a compute node is the smallest compute unit available to an application on the SuperMUC [21] supercomputer which was used to validate the $AEPCP$ model and is briefly described in Subsection 4.2).

The Adaptive Application Energy and Power Predictor (A^2EP^2) is used by the $AEPCP$ model to estimate the application EtS/APC consumption for any given number of compute nodes. A^2EP^2 requires the application historical EtS/APC data. Figure 9 illustrates the workflow of A^2EP^2. As can be seen, if the application has already been executed for a given number of compute nodes (i.e. the EtS/APC consumption for that given number of compute nodes is known), then A^2EP^2 reports the averaged[5] value of all the available application history EtS/APC consumption data for that given number of compute nodes (step $Y1$, Figure 9).

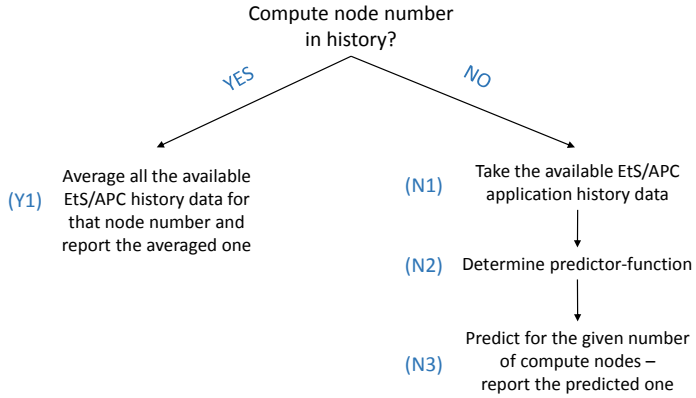


Figure 9. A^2EP^2 workflow

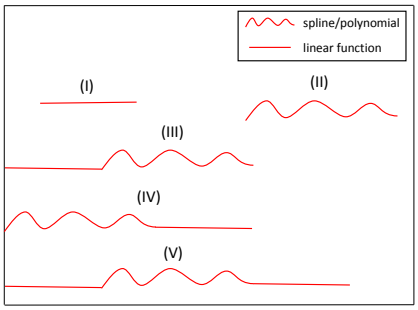


Figure 10. A^2EP^2 predictor estimation scenarios

If the history data of application EtS/APC consumption for a given number of compute nodes is not available, then A^2EP^2 queries the existing history data (step $N1$, Figure 9). This data, in our case, is obtained via a monitoring software toolset called PowerDAM [30, 31] (steps 1 and 2, Figure 8), which is an energy measuring and evaluating tool aimed at collecting and correlating data from different aspects of the data center. Once the application EtS/APC consumption history data is obtained, A^2EP^2 tries to determine a predictor-function (step $N2$, Figure 9) which will have an allowed, user-specified, percentage Root Mean Square Error (%RMSE). %RMSE is calculated from RMSE [23] as follows:

$$\%RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i^{measured} - x_i^{predicted})^2} \cdot \frac{100 \cdot n}{\sum_{i=1}^{n} x_i^{measured}} \tag{7}$$

where

[5]This can be modified to the maximum or the minimum depending on the use case.

- n is the number of available real measurements
- $x_i^{measured}$ is the i^{th} measured real value
- $x_i^{predicted}$ is the i^{th} predicted value

Several estimation techniques (e.g. ordinary least squares, spline interpolation, etc.) accompanied with energy/power consumption specific constraints (e.g. strict positivity) are used by A^2EP^2 for predictor-function determination. Knowing that EtS/APC of both strong and weak scaling applications is of the order of $O(n)$ or $O(n^2)$ (Section 1), A^2EP^2 analyzes the available history data and tries to find data points (from the obtained application history EtS/APC data) which would have a linear dependency. Depending on the found data points, A^2EP^2 divides the available history data set into linear and non-linear segments. A^2EP^2 distinguishes five different segmentations, as illustrated in Figure 10: *linear* (case I) is used for tracking the boundary curves described in Section 1; *non-linear* (case II) is used to track the transitional scaling phases between ideal scalability and no-scalability; *linear combined with non-linear* (case III) is used when to track the mixture of boundary and transitional scaling behavior; *non-linear combined with linear* (case IV) is used to track the mixture of transitional and boundary scaling behavior; and *linear combined with non-linear combined with linear* (case V) is used to track the mixture of boundary-transitional-boundary scaling behavior. For each linear segment, A^2EP^2 uses ordinary least squares to find a linear predictor-function which will have an allowed %RMSE with the available data set in that linear segment. For the non-linear segment, A^2EP^2 uses spline/polynomial interpolations (including also 1^{st} order splines/polynomials) in order to find a predictor-function which will have an allowed %RMSE rate with the history EtS/APC data points which are in that non-linear segment. Although, one could argue that there is no need for estimating higher than 2^{nd} order splines/polynomials because of known theoretical boundary, our experiments show that in the case of very limited application history EtS/APC consumption data, the higher order splines/polynomials are helpful and could result in a better prediction accuracy for a specific range of compute nodes.

Once the predictor-function is obtained from A^2EP^2, it is then used to estimate the EtS/APC values of the application for a given number of compute nodes (steps (4) and (5)). As can be observed, A^2EP^2 implementation is generic and produces individual results for each unique application. It adapts with each additionally available EtS/APC profile data-point for improving the accuracy of the determined (application-specific) predictor-function.

In summary, the described $AEPCP$ model: *(i)* is application neutral - does not need any knowledge on application type (e.g. communication, computation, or memory intensive), scaling properties, etc.; *(ii)* does not require any application code instrumentation; *(iii)* does not introduce any application performance degradation; *(iv)* allows for ahead of time EtS/APC prediction of a given application for a given number of compute nodes (does not require any partial/phase executions); and *(v)* automatically captures the complexity of the underlying hardware platform by taking the input data directly from the system [16], i.e. does not require any manual tuning of application properties or architectural peculiarities of the target platform.

4. Benchmarks and Compute System

4.1. Benchmarks

This subsection describes the two application-benchmarks which were used to validate the proposed model.

Hydro [20] is an application-benchmark extracted from the real world astrophysical code RAMSES [35]. Hydro is a computational fluid dynamics 2D code, which uses the finite volume method, with a second order Godunov scheme [9] and a Riemann solver [26] at each interface on a 2D mesh, for solving the compressible Euler equations of hydrodynamics.

EPOCH is a plasma physics simulation code developed at the University of Warwick as part of the Extendable PIC Open Collaboration Project [2]. EPOCH is based upon the particle push and field update algorithms developed by Hartmut Ruhl [28]. It uses the MPI-parallelized explicit 2^{nd} order relativistic particle-in-cell method, including a dynamic MPI load balancing option.

In contrast to many kernel and synthetic benchmarks, which are used to measure and test certain characteristics (e.g. processor power, communication rate, etc.) of the target platform, Hydro, as well as EPOCH (being application-benchmarks) provide a better measure of a real world performance. Hydro is part of the PRACE (Partnership for Advanced Computing in Europe) [25] prototype evaluation benchmark suit and EPOCH is an open-source real world application used by a large plasma physics community.

4.2. Compute System

SuperMUC (Figure 25), with a peak performance of 3 PetaFLOPS ($= 3 \times 10^{15}$ Floating Point Operations per Second), is the 10^{th} fastest supercomputer in the world (according to Top500 [36] November 2013 rankings) and is a GCS (Gauss Center for Supercomputing) infrastructure system made available to PRACE users. SuperMUC has 155.656 processor cores in 9421 compute nodes and uses IBM LoadLeveler [17] as a resource management system. It's active components (e.g. processors, memory) are directly cooled with an inlet water temperature of up to 40° Celsius [21], allowing for chiller free cooling.

Four re-executions of the EPOCH benchmark on SuperMUC using the **same** set of compute nodes for the node numbers 20, 90, 180, and 256 showed that the measurement error per node number does not exceed 1.2%. Therefor, the quality of a single measurement (independently from the number of utilized compute nodes) is relatively high and there is no strong need for a re-execution of any benchmark.

5. Predicting Energy-to-Solution

This section presents the EtS prediction results for Hydro and Epoch using the *AEPCP* model. The history data points used throughout the paper were chosen on a random basis, since: *(i)* the data center has no control on the resource configurations requested by the users; and *(ii)* to explicitly show that model is independent from any specific history data.

5.1. EtS of Hydro Under Strong Scaling

Figure 11 shows the execution time of Hydro under strong scaling, which adheres to the theoretical discussion presented in Section 1 (Figure 3). Assume that there are three EtS data points in the monitoring history for Hydro (when executed under strong scaling) namely for compute node numbers: 130 with EtS of 7.6kWh; 135 with EtS of 7.9 kWh; and 220 with EtS of 7.6 kWh. Assume further, that there is an application in a job queue, which has an energy tag of strong scaling Hydro, and has a request of 320 compute nodes. The question to

answer here is: *is it possible to predict the energy consumption of Hydro, when executed on* 320 *compute nodes, with* ***only*** *the knowledge of EtS consumption for compute nodes* 130*,* 135*, and* 220*?* Figure 12 shows that the use of $AEPCP$ model leads to a positive answer. The x-axis in Figure 12 represents the compute node number and y-axis represents the corresponding EtS in kWh. The red circle points correspond to the available EtS values. The red solid line shows the predictor-function curve, which was determined by A^2EP^2. A spline with smoothing degree of 1 having an %RMSE of 1% (with the EtS values of node numbers 130, 135, and 220) was estimated by $AEPCP$ model as a predictor function(red solid line, Figure 12). This estimated predictor-function estimates a 7.4 kWh energy consumption for compute node number 320. The green 'x' point in Figure 12 corresponds to the measured, EtS value (7.5 kWh) of Hydro when executed on 320 compute nodes. As can be seen, the prediction error rate[6] for 320 compute nodes is 1.3 %.

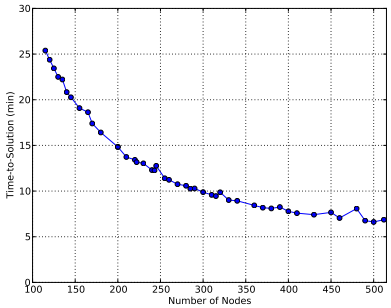
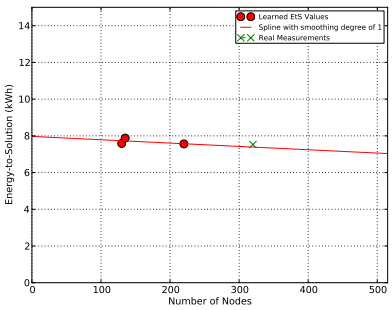
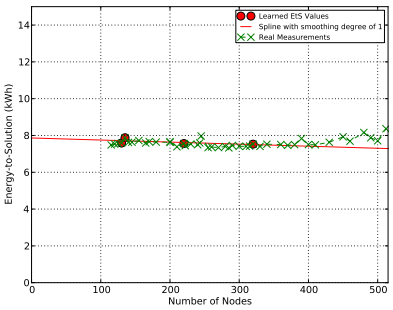


Figure 11. Measured TtS of Hydro under strong scaling



Note: available data points are for node numbers: 130*,* 135*, and* 220

Figure 12. EtS prediction curve and the measured EtS for node number 320



Note: available data points are for node numbers: 130*,* 135*,* 220*, and* 320

Figure 13. EtS prediction curve of Hydro under strong scaling

Figure 13 illustrates the case, when in addition to the Hydro EtS consumption data of compute node numbers 130, 135, 220, the EtS consumption value for already executed 320 compute node number is available to the A^2EP^2. In this case, a spline with smoothing degree of 1 (but with a different angle) having an %RMSE of 1% (the corresponding EtS value of 320 compute nodes was added to the original set of EtS data points for 130, 135, and 220 compute node numbers) was determined by the A^2EP^2 as a predictor-function. The red solid line in Figure 13 illustrates the curve of the predictor-function. The green '-x-x-' curve in Figure 13[7] corresponds to the measured (and **not available** to A^2EP^2) Hydro EtS values for different compute node numbers. As can be seen, the determined predictor-function (the red solid line in Figure 13) shows a relatively small deviation error rate from the measured data (the green '-x-x-' curve in Figure 13). Table 1 summarizes the detailed EtS prediction results for a random set of compute node numbers.

Figure 15 illustrates the real measurements of Hydro, again under strong scaling (Figure 14), but with a smaller input problem size. As usual, the green '-x-x-' points correspond to the real measured EtS data for different compute node numbers, whereas the red line corresponds to the determined predictor-function by A^2EP^2 using the available EtS values for node numbers: 1, 2, 4, 8, 16, 60, and 165 (red circles in Figure 15). As can be seen, a spline with a smoothing degree 2 (having an %RMSE of 1% with the available EtS values of node numbers: 1, 2, 4, 8,

[6] Calculated as: (| predicted value − measured value | /measured value) ∗ 100.
[7] Subsequently presented figures adhere to the same denotations.

Number of Nodes	Measured EtS Value (kWh)	Predicted EtS Value (kWh)	Prediction Error[6] (%)
115	7.5	7.7	2.7
200	7.7	7.6	1.3
285	7.5	7.3	2.7
300	7.4	7.5	1.4
340	7.5	7.5	0
400	7.5	7.4	1.3
460	7.7	7.3	5.1
500	7.7	7.3	5.2

Table 1. EtS prediction results for Hydro (strong scaling)

16, 60, and 165) was determined as a predictor-function by A^2EP^2. Although this determined quadratic behavior contradicts the estimated theoretical linear boundary (Equation 4, Figure 5), it provides an approximation with relative small error rate when compared with the measured data. On the other hand this estimated quadratic predictor starts to deviate from the real measurement data when the application approaches the saturation point, by transitioning to a non-scaling behavior, and thus according to Equation 4, shows a linear behavior of energy consumption with respect to the number of utilized compute nodes.

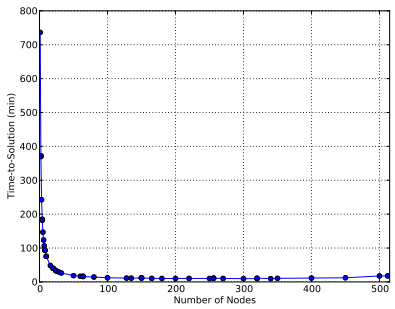
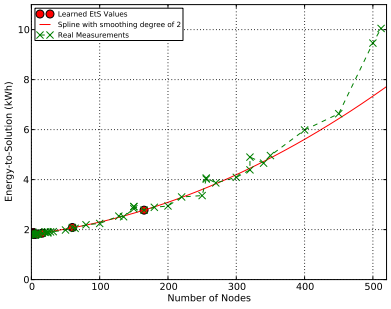
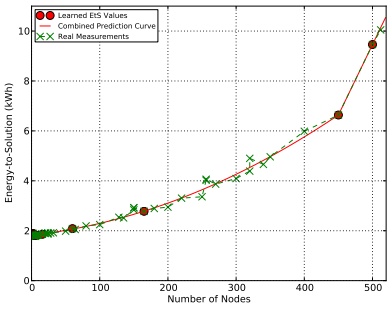


Figure 14. Measured TtS of Hydro under strong scaling *(smaller input problem size)*



Note: available data points are for nodes: 1, 2, 4, 8, 16, 60, *and* 165

Figure 15. EtS prediction curve of Hydro under strong scaling *(smaller input problem size)*



Note: additionally available data points are for nodes: 450 *and* 500

Figure 16. Revisited EtS prediction curve of Hydro under strong scaling *(smaller input problem size)*

One could argue, that there is no reason for executing an application (and thus conducting a prediction) on a higher number of nodes than the node number on which the saturation point for a given application was observed, since no performance increase for that application will be recorded. While true, A^2EP^2 tries to capture this behavior when sufficient data is available. Figure 16 illustrates this option, when EtS values for node numbers 450 and 500 were additionally available to A^2EP^2 for capturing this transitional behavior. As can be seen, the transitional-boundary behavior is tracked at the node number 450, and the quadratic function (illustrating the transitional[8] behavior) is now combined with the linear function illustrating the boundary behavior (case IV, Figure 10).

5.2. EtS of Hydro Under Weak Scaling

Figure 17 illustrates the expected (Section 1) execution behavior of Hydro under weak scaling[9]. Two Hydro EtS values were available for conducting the prediction (for nodes: 6 with EtS of 0.54 kWh; and for 32 with EtS of 2.84 kWh).

[8] In other words from ideal scaling to non-scaling.

[9] For weak scaling, also in Subsection 5.4, we assume that the problem sizes were uniformly adjusted for each compute node number.

As can be observed, a linear predictor-function was estimated by A^2EP^2 (Figure 18) showing a relatively small error rate for up to 512 compute nodes. Table 2 recaps the prediction results for a random set of compute node numbers.

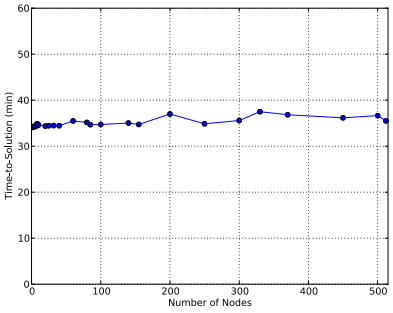
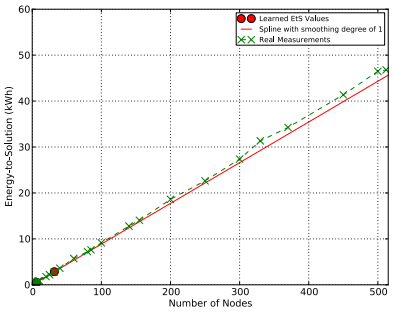


Figure 17. Measured TtS of Hydro under weak scaling



Note: available data points are for nodes: 6, and 32

Figure 18. EtS prediction curve of Hydro under weak scaling

Number of Nodes	Measured EtS Value (kWh)	Predicted EtS Value (kWh)	Prediction Error[6] (%)
10	0.9	0.9	0
25	2.2	2.2	0
40	3.6	3.6	0
80	7.2	7.1	1.4
100	9.1	8.9	2.2
200	18.6	17.7	4.8
370	34.3	32.8	4.4
450	41.4	39.9	3.6
512	46.8	45.4	3

Table 2. EtS prediction results for Hydro under weak scaling

5.3. EtS of EPOCH Under Strong Scaling

Four EtS values (for node numbers: 64 with EtS of 8.6 kWh; 75 with EtS of 8.8 kWh; 90 with EtS of 8.7 kWh, and 128 with EtS of 8.7 kWh) were available for conducting the EtS prediction. Figure 19 shows the measured TtS of EPOCH under strong scaling.

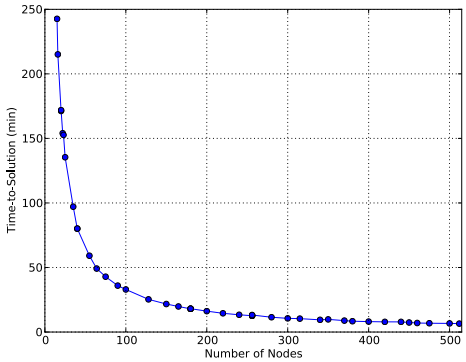
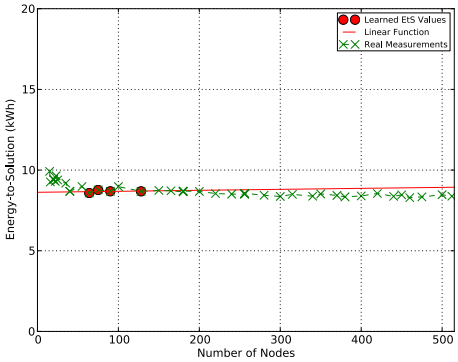


Figure 19. Measured TtS of EPOCH under strong scaling



Note: available data points are for nodes: 64, 75, 90 *and* 128

Figure 20. EtS prediction curve of EPOCH under strong scaling

A linear predictor-function, with an %RMSE of 0.8 with the available data points (red circle points, Figure 20) was determined by A^2EP^2. Figure 20 shows the curve of the determined linear predictor-function (red solid line). The green '-x-x-' curve corresponds to the real measured EtS data of EPOCH under strong scaling.

5.4. EtS of EPOCH Under Weak Scaling

Three EtS values (for node numbers: 16 with EtS of 1.1 kWh; 40 with EtS of 2.9 kWh; and 64 with EtS of 4.5 kWh) were available for conducting EtS prediction.

Figure 21 shows the measured TtS behavior for EPOCH under weak scaling. As can be seen it adheres to the theory. A linear function, having an %RMSE of 2.2% with the available data, was constructed by $A^E P^2$. Figure 22 shows the curve of the constructed predictor-function.

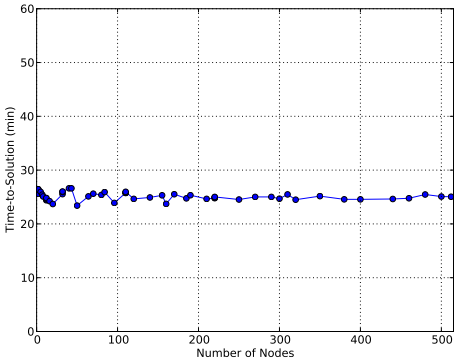
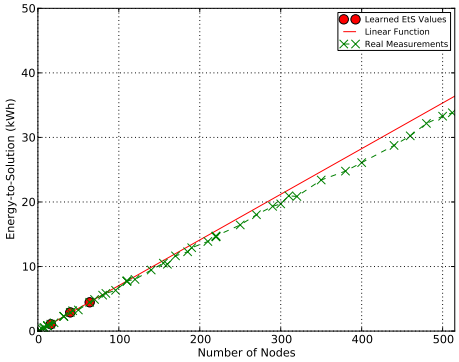


Figure 21. Measured TtS of EPOCH under weak scaling

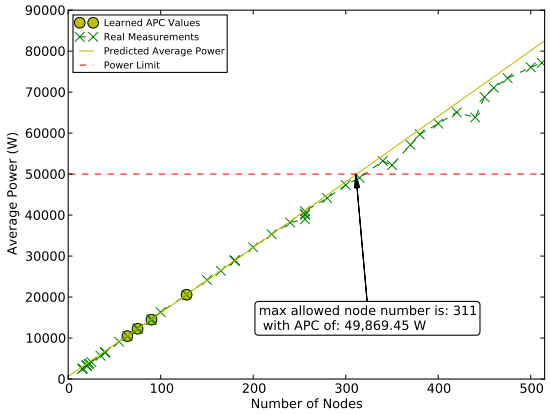


Note: available data points are for nodes: 16, 40, *and* 64

Figure 22. EtS prediction curve of EPOCH under weak scaling

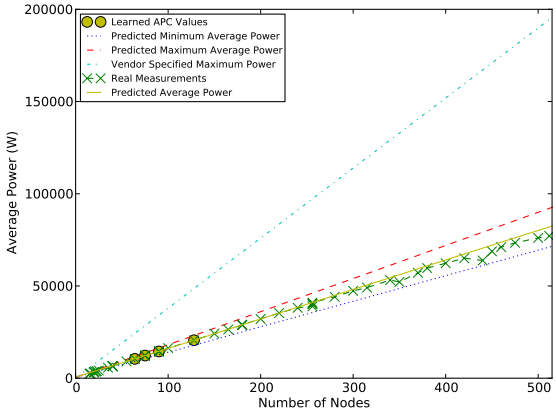
6. Predicting Average Power Consumption

Figure 23 shows the Average Power Consumption (APC) prediction results using the available APC values of four node numbers. As can be seen, the estimated linear predictor-function shows a relatively small error rate for up to 512 compute nodes. Another observation that can be inferred from Figure 23 is that the $AEPCP$ model can suggest the maximum compute node number that can be utilized by the application while preserving the introduced power consumption constraint. Figure 23 illustrates this option in the case of a $50,000$ W power consumption limit. As can be seen, the maximum allowed compute node for running Epoch on SuperMUC, in the case of $50,000$ W constraint, is 311 with predicted APC of $49,869.45$ W.



Note: available data points are for nodes: 64, 75, 90 *and* 128

Figure 23. APC prediction curve for EPOCH under strong scaling



Note: available data points are for nodes: 64, 75, 90 *and* 128

Figure 24. Max and Min APC values for EPOCH under strong scaling

Our observations on SuperMUC supercomputer (Figure 25) show that the average power draw of the individual compute nodes differ when running the same application. This could be due to manufacturing tolerances and variations (e.g. processors [27], memory, power supplies, voltage regulators, etc.). Figure 26 shows the average power draws of different compute nodes

of one of the SuperMUC's Islands (which consists of 516 compute nodes) when running a single MPrime[10] [10] benchmark. As can be seen, despite the hardware homogeneity across the SuperMUC's Island, there is a maximum of 41 W difference (nodes *i05r05a19-ib* with 188 W and *i05r03c28-ib* with 229 W) in average power draw of compute nodes.

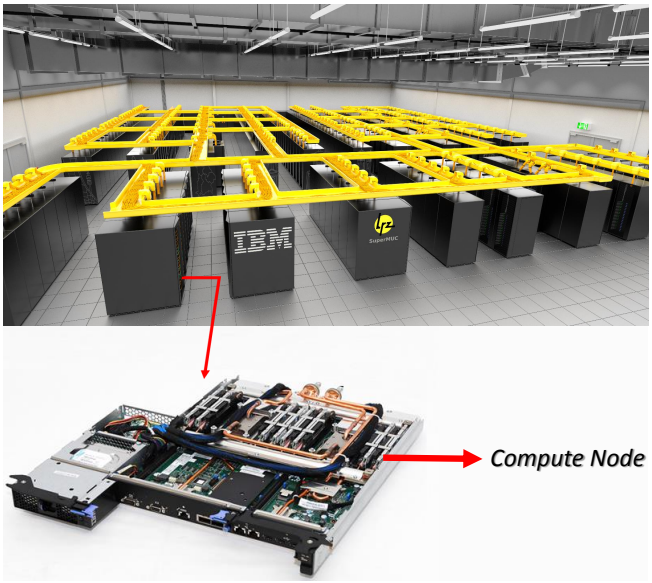


Figure 25. The SuperMUC Supercomputer

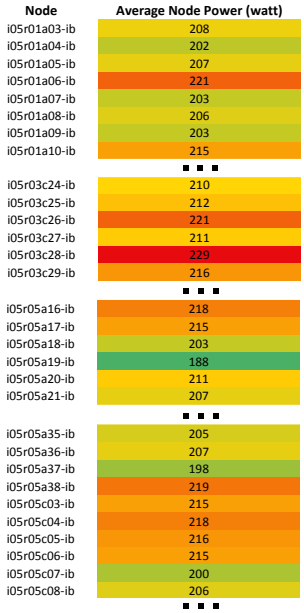


Figure 26. Power draw of compute nodes of the SuperMUC Island

The same behavior of compute node average power draw deviation under the same application execution was observed on CoolMUC [3] (shown in Figure 27). CoolMUC is a direct warm water cooled AMD processor based Linux cluster built by MEGWARE [33] and equipped with 178 compute nodes (2×8-core AMD CPU). It is connected to a SorTech [34] adsorption chiller allowing the exploration of further possibilities of waste heat reuse of the system. CoolMUC has closed racks, and therefore does not require room air conditioning (Figure 27). All heat is removed solely via the chiller-less water cooling loop of the LRZ computer center infrastructure.

Figure 28 shows the power draws of different compute nodes of the CoolMUC Linux cluster when running the same single MPrime benchmark. As can be seen, despite the hardware homogeneity across the cluster, a maximum of 21 W difference in average power draw of compute nodes was observed (nodes *lxa11* with 240 W and *lxa46* with 261 W) during the MPrime benchmark.

If a *system compute node power classification* (Figure 26, Figure 28) is available, then the *AEPCP* model also predicts an application's possible maximum and minimum APC values for the scheduler application-assigned *"best"* and *"worst"* (in terms of power consumption) compute nodes. Using the APC history profile data of a given job J, *AEPCP* normalizes these values to the usage of the best compute node using Equation 8, and to the usage of the worst compute node, using Equation 9.

[10]Mprime is an application-benchmark that searches for Mersenne prime numbers, i.e. prime numbers of form $2^p - 1$, using Fast Fourier Transform algorithm. It introduces an intense workload to processor and memory, and because of that reason is usually used for system stability testing.

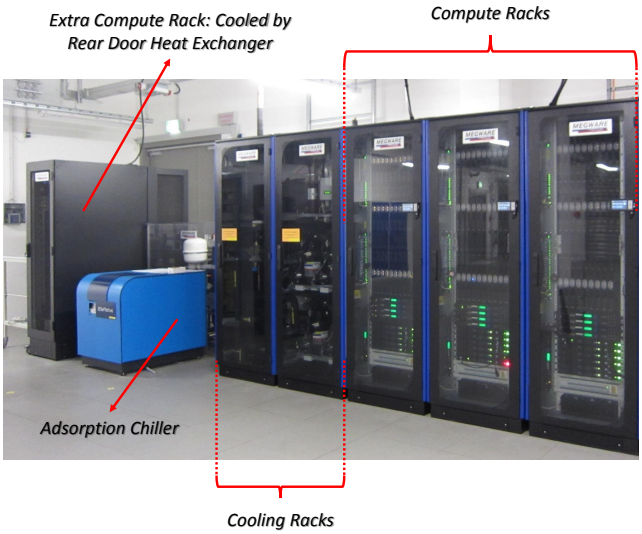


Figure 27. The CoolMUC Linux Cluster

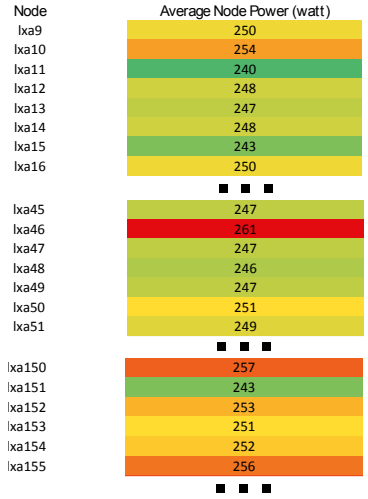


Figure 28. Power draw of compute nodes of the CoolMUC Linux Cluster

$$\| APC(J)^i \|_{min} = APC(J)^i - \sum_{\text{u utilized node of J}} (P_u - P_{min}) \tag{8}$$

$$\| APC(J)^i \|_{max} = APC(J)^i + \sum_{\text{u utilized node of J}} (P_{max} - P_u) \tag{9}$$

where

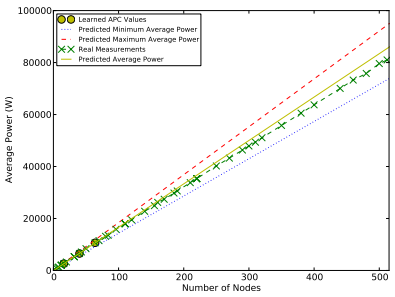
- $APC(J)^i$ - is the average mean power draw of job J using i compute nodes
- P_u - is the average power draw of compute node u obtained from the system compute node power classification (Figure 26)
- P_{min} - is the average power draw of the most efficient (in terms of power) system compute node
- P_{max} - is the average power draw of the least efficient (in terms of power) system compute node

Figure 24 illustrates this option of $AEPCP$ for EPOCH under strong scaling. The dashed '- -' red line illustrates the predicted maximum APC behavior, the bottom dotted '...' blue line the predicted minimum APC behavior, the green '-x-x-' line the real measurement data (obtained from executions on the compute nodes of SuperMUC's Island Figure 26), and finally the yellow straight line depicts the $AEPCP$'s APC predictions.

Several things can be observed from Figure 24. First, that the real measurements do not deviate much from the predicted APC values and stay in between predicted maximum and minimum APC values. Second, one could argue that the maximum APC value of an application for a given number of compute nodes can be derived by multiplying the given compute node number n by $P_{\text{one node}}$ (maximum APC value of one node obtained from system-vendor provided compute node peak power consumption specification). The cyan '-.-' line in Figure 24 illustrates this $n \cdot P_{\text{one node}}$ approximation. While correct, this approximation gives a very rough boundary. For example, the usage of system-vendor provided approximation will lead to $118,108.47$ W power consumption estimation for 311 compute nodes on SuperMUC. Whereas the $AEPCP$ predicted maximum power consumption for the same 311 compute nodes, when running EPOCH under

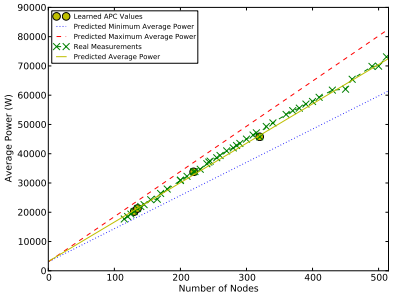
strong scaling, is 55,993.13 W. As can be seen, the vendor specification based approximation is roughly two times larger as compared to the one estimated by the *AEPCP* model.

Figure 29, Figure 30, and Figure 31 illustrate the APC prediction results for EPOCH weak scaling, Hydro strong scaling and Hydro weak scaling correspondingly. As can be seen, all the three predictor-function curves show very small deviation rates from the measured values.



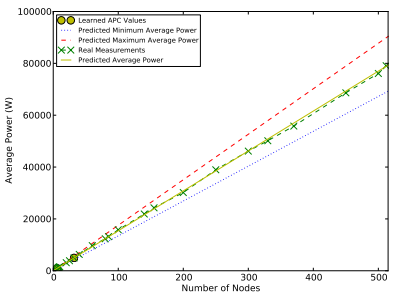
Note: available data points are for nodes: 16, 40 *and* 64

Figure 29. Max and Min APC values for EPOCH under weak scaling



Note: available data points are for nodes: 130, 135, 220, *and* 320

Figure 30. Max and Min APC values for Hydro under strong scaling



Note: available data points are for nodes: 6, *and* 32

Figure 31. Max and Min APC values for Hydro under weak scaling

7. Future Work

As was seen in Section 6, the power draw of the same application on different sets of compute nodes can differ despite hardware homogeneity across the HPC system. Thus, the possibility of compute resource set specific prediction, i.e. the support for exact declaration of compute resources for which the EtS/APC of the given application should be predicted, will produce more accurate results. It is worth noting that some of the EtS/APC measurements might not be completely accurate (e.g. due to possible noisy power sensor readings from which EtS/APC are calculated), and at the same time are not completely false. The specification of measurement "quality" as a weight in the set of available measurements, will allow for a better accuracy in prediction.

In addition to these two points, it is planned to develop an interface between the resource management system(s) and the *AEPCP* model. This interface will allow to dynamically track the possible violations of predefined energy and power consumption constraints depending on *(i)* the current workload information (obtained from the resource management system) and *(ii)* the predicted EtS/APC values for that workload (obtained from the *AEPCP* model).

This work will be included in a toolset at LRZ in order to support energy efficient supercomputing covering and optimizing the full set of influencing parameters: building and cooling infrastructure, supercomputer hardware, application and algorithms, systems software and tools.

8. Conclusion

The following contributions have been made in this paper:

- demonstration of the concept applicability for application power/energy consumption prediction for unknown number of compute nodes from previously observed data;
- explanation of how the application power/energy boundary curves can be defined from the known theoretical works and how this information can be applied in practice;

- exploration of the potential of the presented *Adaptive Energy and Power Consumption Prediction (AEPCP)* model for HPC data center power and energy capping use-cases;
- discussion on how the differences in HPC system compute node power can be used for power prediction;
- provision of a process and a generic implementation that provides application-specific power/energy consumption prediction results without need of the *AEPCP* model-implementation changes;
- since the *AEPCP* model is part of the PowerDAM toolset, this prediction can be done automatically for each application (queued or running) on the HPC system without any application specific adjustments.

The presented AEPCP model is a very interesting solution for HPC data centers, since it requires no application specific knowledge or information. The achieved accuracy is sufficient for the presented two most important use cases. By validating the model, we are just starting to scratch the surface for future possibilities. We are particularly looking forward to apply the model for system/user/data center energy budgeting and system peak power prediction. The suggested model can be an ideal building block for a real-world implementation of energy-aware resource management systems. It can also be used to help users/customers to actively take control over their power/energy budget and can help data centers to move to energy-driven charging policies alternatively to currently existing CPU-hour based charging policies.

The work presented here has been carried out within the PRACE Second Implementation Phase project PRACE-2IP in the Work Package "Prototyping" which has received funding from the European Commission's Seventh Framework Program (FP7/2007-2013) under grant agreement no. RI-283493 and within the SIMOPEK project which has received funding from the German Federal Ministry of Education and Research (BMBF) under grand agreement no. 01IH13007A. The work was achieved using the PRACE Research Infrastructure resources at BAdW-LRZ with support of the State of Bavaria, Germany.

The EPOCH code used in this research was developed under UK Engineering and Physics Sciences Research Council grants EP/G054940/1, EP/G055165/1 and EP/G056803/1.

The authors would like to thank Jeanette Wilde for her valuable comments, Dr. Alexander Block for his kind support in performing experiments, and Dr. Anupam Karmakar for helpful discussions on EPOCH usage.

References

1. Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM, 1967.

2. T. Arber and et al. EPOCH: Extendable PIC Open Collaboration. http://ccpforge.cse.rl.ac.uk/gf/project/epoch/, 2014.

3. Axel Auweter and Herbert Huber. Direct Warm Water Cooled Linux Cluster Munich: CoolMUC. http://inside.hlrs.de/htm/Edition_01_12/article_26.html, 2012.

4. Arndt Bode. Energy to solution: A new mission for parallel computing. In Felix Wolf, Bernd Mohr, and Dieter Mey, editors, *Euro-Par 2013 Parallel Processing*, volume 8097 of

Lecture Notes in Computer Science, pages 1–2. Springer Berlin Heidelberg, 2013.

5. Luigi Brochard, Raj Panda, and Sid Vemuganti. Optimizing performance and energy of hpc application on POWER7. *Computer Science - Research and Development*, 25(3-4):135–140, 2010.

6. G.L. Tsafack Chetsa, L. Lefèvre, J.M. Pierson, P. Stolf, and G. Da Costa. Exploiting performance counters to predict and improve energy performance of HPC systems. *Future Generation Computer Systems*, 2013.

7. Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, pages 13–23, New York, NY, USA, 2007. ACM.

8. Anshul Gandhi, Mor Harchol-Balter, Rajarshi Das, and Charles Lefurgy. Optimal power allocation in server farms. *SIGMETRICS Perform. Eval. Rev.*, 37(1):157–168, June 2009.

9. Sergei Konstantinovich Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 89(3):271–306, 1959.

10. Great Internet Mersenne Prime Search. http://www.mersenne.org/freesoft/, 2014.

11. John L Gustafson. Reevaluating Amdahl's law. *Communications of the ACM*, 31(5):532–533, 1988.

12. Georg Hager, Jan Treibig, Johannes Habich, and Gerhard Wellein. Exploring performance and power properties of modern multi-core chips via simple machine models. *Concurrency and Computation: Practice and Experience*, 2014.

13. Can Hankendi and Ayse K Coskun. Adaptive power and resource management techniques for multi-threaded workloads. In *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, pages 2302–2305. IEEE Computer Society, 2013.

14. John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2012.

15. Chung-hsing Hsu and Wu-chun Feng. A power-aware run-time system for high-performance computing. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, pages 1–, Washington, DC, USA, 2005. IEEE Computer Society.

16. Engin Ipek, Bronis R De Supinski, Martin Schulz, and Sally A McKee. An approach to performance prediction for parallel applications. In *Euro-Par 2005 Parallel Processing*, pages 196–205. Springer, 2005.

17. Subramanian Kannan, Mark Roberts, Peter Mayes, Dave Brelsford, and Joseph F Skovira. Workload Management with LoadLeveler. http://www.redbooks.ibm.com/abstracts/sg246038.html, 2001.

18. Jonathan G Koomey. Estimating total power consumption by servers in the US and the world, 2007.

19. Jonathan G Koomey. Worldwide electricity used in data centers. *Environmental Research Letters*, 3(3), 2008.

20. Pierre-François Lavallée, Guillaume Colin de Verdière, Philippe Wautelet, Dimitri Lecas, and Jean-Michel Dupays. Porting and optimizing hydro to new platforms and programming paradigms-lessons learnt. http://www.prace-ri.eu/IMG/pdf/porting_and_optimizing_hydro_to_new_platforms.pdf, December 2012.

21. Leibniz Supercomputing Centre (LRZ) of the Bavarian Academy of Sciences and Humanities. http://www.lrz.de/, 2014.

22. Timo Minartz, JulianM. Kunkel, and Thomas Ludwig. Simulation of power consumption of energy efficient cluster hardware. *Computer Science - Research and Development*, 25(3-4):165–175, 2010.

23. Kevin P Murphy. *Machine learning: a probabilistic perspective*. The MIT Press, Cambridge, MA, 2012.

24. Catherine Mills Olschanowsky, Tajana Rosing, Allan Snavely, Laura Carrington, Mustafa M Tikir, and Michael Laurenzano. Fine-grained energy consumption characterization and modeling. In *High Performance Computing Modernization Program Users Group Conference (HPCMP-UGC), 2010 DoD*, pages 487–497. IEEE, 2010.

25. Partnership For Advance Computing In Europe. http://www.prace-ri.eu/, 2014.

26. Philip L Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics*, 43(2):357–372, 1981.

27. Barry Rountree, Dong H Ahn, Bronis R de Supinski, David K Lowenthal, and Martin Schulz. Beyond DVFS: A First Look at Performance Under a Hardware-Enforced Power Bound. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 947–953. IEEE, 2012.

28. Hartmut Ruhl. Classical Particle Simulations with the PSC code. https://www.physik.uni-muenchen.de/lehre/vorlesungen/wise_09_10/tvi_mas_compphys/vorlesung/Lecturescript.pdf.

29. Yuan Shi. Reevaluating Amdahl's law and Gustafson's law. *Computer Sciences Department, Temple University (MS: 38-24)*, 1996.

30. Hayk Shoukourian, Torsten Wilde, Axel Auweter, and Arndt Bode. Monitoring power data: A first step towards a unified energy efficiency evaluation toolset for HPC data centers. http://www.sciencedirect.com/science/article/pii/S1364815213002934, 2013.

31. Hayk Shoukourian, Torsten Wilde, Axel Auweter, Arndt Bode, and Petra Piochacz. Towards a unified energy efficiency evaluation toolset: an approach and its implementation at Leibniz Supercomputing Centre (LRZ). http://dx.doi.org/10.3929/ethz-a-007337628, 2013.

32. Shuaiwen Leon Song, Kevin Barker, and Darren Kerbyson. Unified performance and power modeling of scientific workloads. In *Proceedings of the 1st International Workshop on Energy Efficient Supercomputing*, E2SC '13, pages 4:1–4:8, New York, NY, USA, 2013. ACM.

33. MEGWARE Computer Vertrieb und Service GmbH. http://www.megware.com/en/default.aspx, 2014.

34. SorTech AG. http://www.sortech.de/en/, 2014.

35. Romain Teyssier. The RAMSES Code. http://irfu.cea.fr/Phocea/Vie_des_labos/Ast/ast_sstechnique.php?id_ast=904, 2013.

36. Top500. http://top500.org/, 2013.

37. Dong Hyuk Woo and Hsien-Hsin S Lee. Extending Amdahl's Law for Energy-Efficient Computing in the Many-Core Era. *IEEE computer*, 41(12):24–31, 2008.