Proceedings Paper:

Singh, Amit Kumar, Dziurzanski, Piotr and Indrusiak, Leandro Soares orcid.org/0000-0002-9938-2920 (2016) Value and energy optimizing dynamic resource allocation in many-core HPC systems. In: 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom). 7th IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2015, 30 Nov - 03 Dec 2015 IEEE , CAN , pp. 180-185.

https://doi.org/10.1109/CloudCom.2015.22

Value and Energy Optimizing Dynamic Resource Allocation in Many-core HPC Systems

Amit Kumar Singh, Piotr Dziurzanski and Leandro Soares Indrusiak
Department of Computer Science, University of York, Deramore Lane, Heslington, York, YO10 5GH, UK.
Email: {amit.singh, piotr.dziurzanski, leandro.indrusiak}@york.ac.uk

***Abstract*—The conventional approaches to reduce the energy consumption of high performance computing (HPC) data centers focus on consolidation and dynamic voltage and frequency scaling (DVFS). Most of these approaches consider independent tasks (or jobs) and do not jointly optimize for energy and value. In this paper, we propose DVFS-aware profiling and non-profiling based approaches that use design-time profiling results and perform all the computations at run-time, respectively. The profiling based approach is suitable for the scenarios when the jobs or their structure is known at design-time; otherwise, the non-profiling based approach is more suitable. Both the approaches consider jobs containing dependent tasks and exploit efficient allocation combined with identification of voltage/frequency levels of used system cores to jointly optimize value and energy. Experiments show that the proposed approaches reduce energy consumption by 15% when compared to existing approaches while achieving significant amount of value and reducing percentage of rejected jobs leading to zero value.**

***Keywords*—Many-core, High Performance Computing, Resource allocation, Value, Energy consumption, Value curves.**

I. INTRODUCTION

Large scale HPC systems have become increasingly powerful by incorporating many-core architectures, but there is a huge concern about the energy required to operate such systems [1]. The reports indicate the energy consumption of data centers to be between 1.1% and 1.5% of the worldwide electricity consumption [2]. Further, the power requirements of these systems are increasing rapidly. Thus, minimizing energy consumption during the operation of these systems is of paramount importance.

Many processor cores support different power levels during operation by employing dynamic voltage and frequency scaling (DVFS) or dynamic frequency scaling (DFS) [3]. The power consumption during operation, i.e. execution, is referred to as dynamic power and a lower operating voltage and frequency represents lower dynamic power consumption level. The voltage and frequency of one or more cores can be adjusted depending upon their workload in order to reduce energy consumption while not violating timing constraints [4]. Further, cores unused for a long time can go in sleep mode to further reduce energy consumption.

In a many-core HPC system, jobs arrive at different moments of time and they need to be serviced by allocating on the available system cores at run-time. In doing so, the value (utility) achieved by servicing the jobs should be maximized while trying to minimize the overall energy consumption during system operation as mentioned earlier. A job may contain a number of dependent/independent tasks or processes to be allocated on the system cores. The allocation results for each job determine the value to be achieved and also energy consumption, and thus allocation process needs to optimize both the metrics (value and energy).

Previous researchers have introduced notion of values (economic or otherwise) of the jobs to define their importance level [5]. In overload situations where demand for available resources is higher than the supply, such a notion facilitates in deciding to hold the low value jobs for late allocation and allocating limited resources to the high value jobs. The value of a job can change over time to reflect the impact of the computation over the business processes, which adds complexity to the allocation process.

Existing dynamic resource allocation approaches allocate dynamically arriving jobs to the platform resources by employing light-weight heuristics that can find an allocation quickly. There have also been efforts to utilize design-time profiled results to facilitate efficient resource allocation and reduce the computations at run-time [6]. These efforts seem promising to design job-specific-clouds, where the clients (or customers) and their jobs to be submitted for execution are pre-defined, which can be realized from the historical data. However, they optimize only for value. Further, existing approaches optimizing for both value and energy cannot be applied to dependent tasks. Since an HPC job may contain a set of dependent tasks, there is a need to devise resource allocation approaches to be applied on dependent tasks while optimizing both value and energy.

Contribution: This paper addresses shortcomings of existing resource allocation approaches and proposes a profiling based and a non-profiling based approach. They exploit efficient allocation combined with identification of appropriate voltage/frequency levels of used platform cores in order to jointly optimize value (utility) and energy. For each job, the profiling based approach utilizes design-time profiled results obtained by a technique that provides value and energy optimized operating points. The allocation and voltage/frequency levels are identified by exploring the search space including various allocations and possible voltage/frequency levels. In the non-profiling based approach, at run-time, the allocation is identified by considering load balancing and communication optimization, and voltage/frequency levels by exploring the search space.

Paper Organization: Section II presents related works. The models of job, value of a job and HPC platform along with the problem definition are introduced in Section III. Proposed approaches are introduced in Section IV. Section V presents experimental results and Section VI concludes the paper.

II. RELATED WORK

Market-inspired resource allocation heuristics are proven to provide promising results in the overload situation that is normally encountered in HPC system [7]. The heuristics employ notion of values of jobs, where values represent importance levels. Some researchers assume fixed value of a job [8], whereas others consider values that can change with time [5].

Market-inspired heuristics allocate jobs in several ways. For example, the highest value job is chosen first [8]. This approach might lead to small amount of available resources if

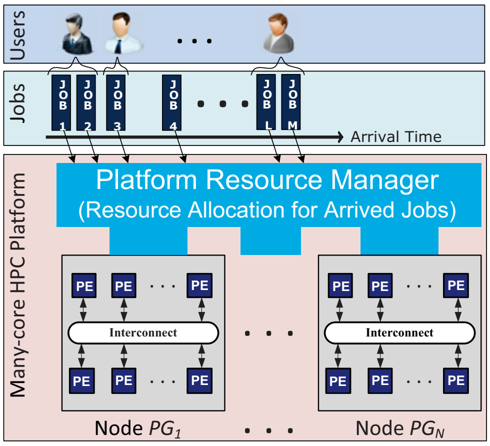

Fig. 1. System model adopted in this paper. A cloud data center containing different nodes (servers) with dedicated cores (PEs) to execute jobs submitted by multiple users.

a high value job requires a large amount of resources. To overcome above problem, the job having maximum value density can be chosen first [9], where the value density is computed as value divided by the amount of required computational resources Another heuristic to choose the job having minimum remaining value first is also proposed [10]. The remaining value is calculated as the area under the value curve from the current time to the time when its value is zero. These heuristics try to optimize overall value, but they do not consider energy consumption optimization and DVFS capable cores.

Energy optimization approaches for HPC data centers have focused mainly on VMs consolidation and DVFS. In consolidation, VMs with low utilization are placed together on a single host so that other used hosts can be freed to shut them down [11]–[13]. DVFS based approaches have been explored to reduce energy consumption is several areas, e.g., clusters [14], [15], web servers [16] and HPC data centers [4]. The approaches for HPC data centers (e.g., [4]) do not consider jobs containing dependent tasks. For other application domains, DVFS techniques for dependent tasks are explored (e.g., [17]), but optimization is not performed for value.

Some heuristics considering DVFS and optimizing both the value and energy consumption are reported in [5]. However, they consider independent tasks or jobs containing independent tasks. There are some additional multi-criteria optimization approaches, but they perform static resource allocation [18], [19]. Further, in dynamic resource allocation process, they do not use design-time profiling results, which can provide optimized value and energy. In contrast, our profiling and non-profiling based dynamic resource allocation approaches consider jobs containing dependent tasks and jointly optimizes for both value and energy while applying DVFS.

III. System Model and Problem Definition

Figure 1 shows our target system model, which is based on typical industrial HPC scenario. The system contains a *many-core HPC platform* that executes a set of *jobs* submitted by various *users* at different moments of time. The jobs are submitted to the *platform resource manager* that allocates resources to them. This section provides a brief overview of the platform and workload model along with the problem definition.

A. Many-core HPC Platform Model

The HPC platform HP contains a set of nodes $(PG_1, ..., PG_N)$, where each node (server) contains a set

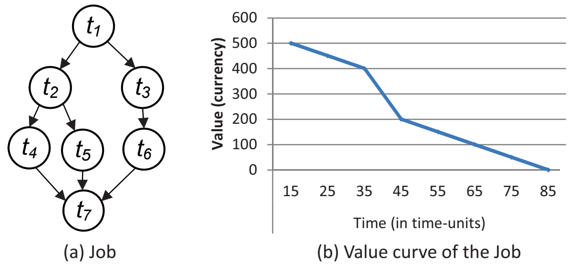

Fig. 2. An example job model and its value curve.

of homogeneous cores, referred to as processing elements (PEs), as shown in the bottom part of Figure 1. A node n is represented as a set of cores C_n, which communicate via an interconnect. Each core is assumed to support DVFS. A *platform resource manager* controls access of platform resources and coordinates the execution of jobs submitted by the users, which facilitates efficient management of resources and incoming requests.

B. Job Model

Each job j in the HPC workload is modelled as a directed graph $TG = (T; E)$, where T is the set of tasks of the job and E is the set of directed edges representing dependencies amongst the tasks. Figure 2 (a) shows an example job that contains 7 tasks (t_1,..,t_7) connected by a set of edges. Each task $t \in T$ is associated with its execution time (*ExecTime*, measured as worst-case execution time (WCET)), when allocated on a core operating at a particular voltage level. Such information can be obtained from previous executions of the tasks. Each edge $e \in E$ represents data that is communicated between the dependent tasks. A job j is also associated with its arrival time AT_j.

C. Value Curve of a Job

For each job j, the value curve VC_j is a function of the value of the job to the user depending on the completion time of the job [5]. The value curve is usually a monotonically-decreasing function and trends towards zero with the increasing completion time, as shown in Figure 2 (b). We assume a value curve is given for each job, as this reflects its business importance as assessed by the end user (i.e. domain specific economic model). The description of the economic model is orthogonal to our approach and out of scope of this paper.

Each job is considered to have a soft deadline [4]. This implies that the violation of deadline does not make the computation irrelevant, but reduces its value for the user [5], [20]. Deadlines missed by large margins may result in zero value and thus the computation becomes useless for the user. Further, the energy spent on such computation can be considered as wasted. Therefore, the job request should be rejected if no (zero) value can be obtained by executing it.

D. Energy Consumption of a Job

The total energy consumption (E_{total}) of a job is computed as the sum of dynamic and static energy as follows.

$$E_{total} = E_{dynamic} + E_{static} \quad (1)$$

The dynamic energy consumption for all the tasks in the job is estimated from equation (2).

$$E_{dynamic} = \sum_{\forall t \in T} (ExecTime[t] \rightarrow c_v) \cdot (pow \rightarrow c_v)] \quad (2)$$

where $ExecTime[t] \rightarrow c_v$ and $pow \rightarrow c_v$ are the execution time of task t mapped on core c operating at voltage v, and respective power consumption, respectively. The $ExecTime$ measures are provided in the job model. It is assumed that the power consumption at different operating voltages is known in advance and taken from chip manufacturer's data sheet.

The E_{static} for each core is computed as the product of overall execution time of the job and static power consumption of the used cores. Unused cores are considered as power gated so that they do not contribute to the overall energy consumption.

E. Problem Definition

The resource allocation problem targeted in this paper is to jointly optimize value and energy while servicing arrived jobs. To summarize, the targeted problem considers the following set of input, constraints and objective.

- **Input:** Workload, i.e., Job set $(j_1, ..., j_M)$, Value curve of each job VC_j, Arrival time of each job AT_j $(j \in 1, \ldots, M)$, Cores of the HPC platform nodes $(PG_1, ..., PG_N)$, Voltage levels $(v_1, ..., v_l)$ supported by each core.
- **Constraints:** Limited resources (cores) on each node of HP.
- **Objective:** Maximize overall value Val_{total} and minimize energy consumption E_{total}.

For an arrived job, the allocation process followed by the global resource manager needs to identify the node to execute the job, tasks to cores allocation inside the node, and the voltage/frequency levels of the cores executing tasks of the job. We assume negligible time for switching between voltage/frequency levels of a core as it is in the order of nanoseconds while tasks execution is in the order of minutes or hours [21]. Since there are several possible allocations (tasks to cores assignment) for a job and several voltage scaling (VS) options for each allocation, exploring the complete design space to identify the optimal design in terms of value and energy might not be feasible within acceptable time. Therefore, only efficient allocations and appropriate VS options need to be evaluated. Further, for dependent tasks, applying VS on a core is rather challenging as one needs to capture the VS effect on the execution of dependent tasks allocated on other cores.

IV. Proposed Value and Energy Optimizing Resource Allocation Approaches

This section describes our proposed approaches. In contrast to conventional existing efforts, our approaches differ mainly in following aspects: *1)* consider jobs containing dependent tasks, *2)* apply DVFS, *3)* jointly optimize value and energy, and *4)* utilize profiling results in case the jobs are known in advance and profiled.

In order to allocate platform cores to the incoming jobs at run-time, the platform resource manager is invoked to find allocations. The manager follows profiling or non-profiling based approach, as shown in Figure 3. The details of these approaches are as follows.

A. Profiling Based Approach (PBA)

This approach uses design-time profiling results of the jobs in the historical data to perform run-time resource allocation for the incoming jobs, as shown in Figure 3 (a). For each job, the profiling process identifies the allocation and voltage/frequency levels leading to optimized response time (determines value) and energy consumption when utilizing different amount of computing power in terms of number

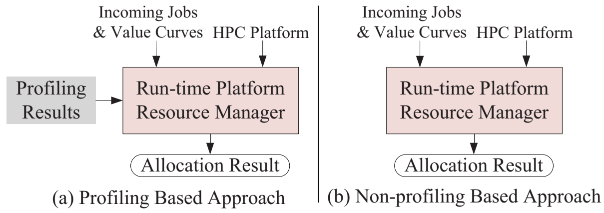

Fig. 3. Profiling and non-profiling based approaches.

of cores. The response time is calculated as the difference between the end and start time of the job execution after allocating resources to it and should be minimized to optimize value. To jointly optimize value and energy, we consider to minimize the product of response time and energy consumption. At different number of cores, the allocation and voltage/frequency levels leading to minimum product value are identified by employing a genetic algorithm (GA) based evaluation, similarly as in [22]. The number of cores is varied from one to the number of tasks in the job. Such variation can exploit all the potential parallelism present in the job as each task can occupy only one core. For each job, the allocation, voltage/frequency levels, value corresponding to the response time and energy consumption at different number of cores are stored as the profiling results.

To perform resource allocation by using the profiling results, the manager follows Algorithm 1. The algorithm takes profiling results of the jobs from the storage along with their value curves and arrival times, and the HPC Platform HP as input and identifies the value and energy optimizing allocation for each job based on the number of available cores at different nodes in the platform. The algorithm checks mainly for two events as follows: 1) *any already allocated job(s) finish execution* to update the platform resources (lines 1-3), and *2) any job(s) arrive into the platform* to put into a job queue (lines 4-6). If any of the two events or both of them occurs, the algorithm tries to perform resource allocation for the queues job(s) having non-zero values (lines 7-17).

To perform resource allocation for all valuable queued jobs (i.e., jobs having positive values), all of them (*count* = 0 to *JobQueue.size()*, line 8) are tried to be allocated on the platform resources as along as any core is available. It is ensured that a queued job having zero value at the allocation time is dropped from the queue as no value can be made out of it. The allocation process continues until all the arrived jobs are allocated or dropped due to having zero value while waiting in the job queue. First, bids (in terms of number of available cores) from different platform nodes are collected, then the maximum bid ($maxBid$) and the corresponding node is selected (line 9). Choosing such a node to use its cores helps to achieve better load balancing amongst nodes and thus better resource utilization. In case more than one nodes have the same amount of bid, any of them is chosen. If the estimate of $maxBid$ is greater than zero ($maxBid > 0$, line 10), i.e., at least one core is available in the platform, the value/energy estimates of jobs utilizing $maxBid$ cores are computed and the job leading to maximum value per energy consumption ($maxValuePerEnergyJob$) is selected to be scheduled to the node having $maxBid$ cores by following the allocation and voltage/frequency levels leading to the optimized value and energy. The computation of value/energy for each job considers its value at the allocation time and the exact number of cores to be used by the job computed as minimum between $maxBid$ and the number of cores to be used to achieve maximum value/energy. The platform resources are updated after

ALGORITHM 1: Profiling Based Resource Allocation

Input: Incoming Jobs with arrival times, Jobs' profiling results and value curves, HPC Platform HP.
Output: Resource Allocation for Incoming Jobs.

1 **if** *allocated_job(s) finish execution* **then**
2   Update platform resources;
3 **end**
4 **if** *job(s) arrive* **then**
5   Put the *job(s)* in *JobQueue*;
6 **end**
7 **if** *JobQueue contains job(s) having positive values* **then**
8   **for** *count = 0 to JobQueue.size()* **do**
9   Collect $bids$ from all nodes and select $maxBid$;
10   **if** $maxBid > 0$ **then**
11   Compute $value/energy$ estimates of unscheduled jobs when utilizing $maxBid$ cores;
12   Select $maxValuePerEnergyJob$ and its $value$, $energy$, $allocation$, and $voltage/frequency\ levels$ from profiling results ;
13   Schedule $maxValuePerEnergyJob$ on node having $maxBid$ cores by following the $allocation$ to perform execution at $voltage/frequency\ levels$;
14   Update platform resources;
15   **end**
16   **end**
17 **end**

scheduling each job to have up to date resources' availability information for the next allocation instance. This helps to achieve an accurate and efficient allocation. Similar process is repeated for all the arrived jobs.

B. Non-profiling Based Approach (NBA)

The NBA approach does not use profiling results as no historical pattern of jobs is available to perform advance profiling. Rather, all the computations are performed at run-time. The steps followed by the NBA are similar to PBA and sketched in Algorithm 2. Here, if $maxBid$ is greater than zero ($maxBid > 0$), the following two main steps are employed: *i)* Compute $values$ of unscheduled jobs by finding allocations on $maxBid$ cores (line 6), and *ii)* Identify voltage/frequency levels of used cores to execute allocated tasks to maximize value over energy (line 8), which are described subsequently.

In step *i)*, firstly, an appropriate allocation for each job is identified by allocating on $maxBid$ cores. The allocation considers the exact number of cores to be used, which is the minimum between $maxBid$ cores and the number of cores equivalent to the number of tasks in the job. The exact number of cores could be higher than that of PBA as no profiling information is available to identify it exploiting the maximum parallelism. To find an efficient allocation, we try to balance load across the used cores. Every task of the job is allocated to a core such that the processing load is balanced over the cores. In case the number of tasks in the job is higher than the number of cores, the approach allocates highly communicating tasks on the same core to reduce the communication overhead. These considerations can lead to minimal response time and thus completion time of the job, resulting in maximum value. After finding the allocation, the value is computed as the value in the corresponding value curve at the completion time by taking the arrival time into account. Similarly, value achieved by each job is computed.

From all the jobs, the one leading to the maximum value ($maxValuableJob$), corresponding $allocation$ and $value$ is

ALGORITHM 2: Non-profiling Based Resource Allocation

Input: Incoming Jobs with arrival times, Value curves of Jobs, HPC Platform HP.
Output: Resource Allocation for Incoming Jobs.

1 Steps 1 to 6 of Algorithm 1;
2 **if** *JobQueue contains job(s) having positive values* **then**
3   **for** *count = 0 to JobQueue.size()* **do**
4   Collect $bids$ from all nodes and select $maxBid$;
5   **if** $maxBid > 0$ **then**
6   Compute $values$ of unscheduled jobs by finding allocations on $maxBid$ cores;
7   Select $maxValuableJob$, its $allocation$ and respective $value$;
8   Identify voltage/frequency levels of used cores in the $allocation$ to execute allocated tasks to optimize value and energy;
9   Schedule $maxValuableJob$ on node having $maxBid$ cores by following the $allocation$ to perform execution at found voltage/frequency levels;
10   Update platform resources;
11   **end**
12   **end**
13 **end**

selected (line 7). Then, voltage/frequency levels are identified in step *ii)* as described subsequently.

Step *ii)* follows Algorithm 3, which takes the set of voltage scaling (VS) levels V available for cores as input and identifies the VS levels to be applied on cores to execute allocated tasks. For each task t, available VS levels are applied, and response time and value of the job at its completion is computed. From here onwards, applying voltage scaling on a task implies applying voltage scaling on the allocated core for the task. Similarly, VS level of a task implies VS level of the allocated core to execute the task. The value at completion is estimated by looking into the corresponding value curve while taking the arrival time of the job into account. If an applied VS on a task is valuable ($value_{job\_completition} > 0$), then total energy consumption of the job is calculated from Equation (1). Next, value at per unit of energy consumption ($ValPerUnitEnerg$) is computed. Thereafter, the task and its VS level corresponding to maximum $ValPerUnitEnerg$ is found to fix the voltage level to execute the task. The same process is repeated to find VS levels of other tasks. Once voltage/frequency levels are identified, the $maxValuableJob$ is scheduled on the node having $maxBid$ cores based on the $allocation$ to perform execution at the identified voltage/frequency levels (Algorithm 2).

V. EXPERIMENTAL RESULTS

The proposed value and energy optimizing resource allocation approaches have been implemented in a C++ prototype and integrated with a SystemC functional simulator. As a workload, job models from historical data of an industrial HPC system at High Performance Computing Center Stuttgart (HLRS) are considered. The jobs in the workload have varying arrival time. It is considered that higher numbers of jobs arrive in peak times as compared to off-peak times. To sufficiently stress the platform, we consider all the jobs arriving over a day, i.e., 24-hour period. Each job contains a set of dependent tasks as described earlier. For each task, the worst-case execution time (WCET) is known a priori and specified in the job model. The number of tasks in the jobs varies from 5 to 10. Further, it is assumed that the value curve of each job is given.

To evaluate our approaches under different load conditions,

ALGORITHM 3: Voltage/frequency Identification

Input: $V = \{v_i | \forall i \in [1, \cdots, n]\}$.
Output: VS levels of tasks.

repeat
  for *each task t whose VS level is not fixed* **do**
    for *each VS level* v_i **do**
      Apply VS v_i on t, and compute $response\_time$ and $value_{job\_completion}$;
      if $value_{job\_completion} > 0$ **then**
        Calculate total energy consumption E_{total} (by Equation 1) when applying v_i on t;
        $ValPerUnitEnerg = \frac{value_{job\_completion}}{E_{total}}$;
      end
    end
  end
  Find task t_f & VS level v_f corresponding to maximum $ValPerUnitEnerg$;
  Fix voltage of t_f to v_f;
until *VS levels of all tasks are not fixed*;

we conducted experiments with varied arrival rates of jobs while keeping higher number of arrivals during peak times over off-peak times. We have considered low, moderate and high arrival rates, where jobs arrive in the orders of a few seconds, dozens of seconds and minutes, respectively. It is assured that the total number of jobs for different arrival rates remains the same as the number of jobs considered for 24 hours. To evaluate our approaches for different number of available servers (nodes), varying number of nodes are considered in the HPC platform. Further, the number of cores at each node is also varied to evaluate the approaches for assorted chip manufacturing technologies, where different number of cores can be integrated within a physical chip. The platform cores are assumed as the cores of Intel Core M processor, which supports 6 voltage/frequency levels of operation.

In addition to overall value achieved by executing the arrived jobs and required energy consumption for the execution, we also evaluate the percentage of rejected jobs that are removed from the job queue as their value becomes zero before the resources become available to allocate them. The rejected jobs also include jobs achieving zero value after their execution, which can be prevented by employing proper admission control and schedulability analysis.

A. Experimental Baselines

As discussed in Section II, the existing algorithms applying DVFS to execute jobs optimize either only for value [8] or energy [17], and both value and energy optimizing approaches do not consider jobs containing dependent tasks [5]. We compare results obtained from our approaches (PBA and NBA) to those of [8] and [17] as they can be applied to jobs containing dependent tasks. In [8], the cores are assumed to operate at the highest supported voltage level to optimize the value. This approach is referred to as *ValOpt* and helps to recognize energy savings by approaches applying DVFS. To employ *ValOpt*, the voltage/frequency identification step (line 8, in Algorithm 2) has been removed. The approach of [17] identifies voltage/frequency levels of cores to execute the allocated tasks in order to optimize only energy consumption and all the allocated tasks on a core execute on a fixed identified voltage/frequency level, referred to as fixing cores power states (FCPS). Therefore, it has been extended to optimize both the value and energy for a fair comparison. To employ this approach, the greedy algorithm of [17] that fixes voltage/frequency levels of cores one-by-one during consecutive iterations is called for voltage/frequency identification

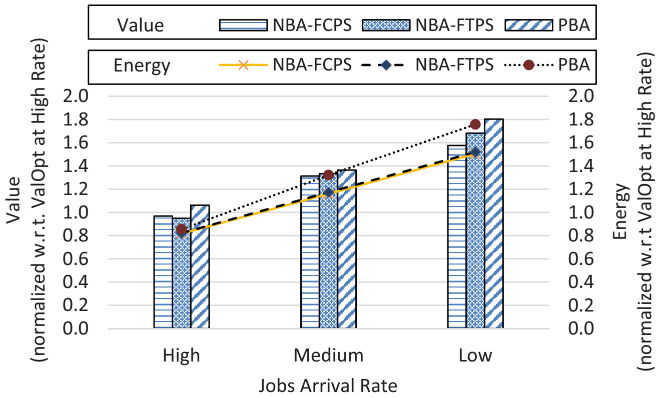

Fig. 4. Value and Energy at different arrival rates.

in Algorithm 2 (line 8) and the approach is referred to as NBA-FCPS. Our approach identifies voltage/frequency levels of tasks in the similar manner, where tasks scheduled on a core can be executed on different voltages, referred to as fixing tasks power states (FTPS). In this case, our NBA approach has been referred to as NBA-FTPS.

B. Value and energy consumption at different arrival rates

Figure 4 shows the overall value and energy consumption when various approaches are employed for different arrival rates of jobs. A high arrival rate indicates that the jobs arrive quite frequently, whereas less frequently in low arrival rate. The value and energy estimates are normalized with respect to (w.r.t.) the value and energy by ValOpt approach at high arrival rate. The shown results have been computed for 3 nodes, where each node contains 8 cores. A couple of observations can be made from the figure. *1)* The value obtained by all the approaches increases from high to low arrival rates as more jobs are processed before their value becomes zero due to late availability of cores. *2)* The value obtained by PBA approach is always higher than that of other approaches due to joint optimization effect. On an average, PBA achieves 5.6% higher value than that of ValOpt. The joint optimization also leads to higher energy consumption when jobs arrival rate is not high. *3)* The energy consumption by NBA-FCTS and NB-FTPS is close to each other and lower than that of ValOpt. On an average, NBA-FCTS and PBA reduce energy consumption by 15.8% and 5.8%, respectively, when compared to ValOpt. Therefore, for the sake of both value and energy optimization, PBA is recommended to be employed.

C. Value and energy consumption with varying number of nodes and varying number of cores in each node

Figure 5 shows the influence of the number of nodes (servers) on the overall value and energy consumption. At each node, a total of 8 cores are considered. The shown results are for high arrival rate of the jobs. The value and energy results are normalized w.r.t. the value and energy by ValOpt approach at 2 nodes. It can be observed that the overall value by all the approaches increases with the number of nodes due to increased processing capability leading to completion of higher number of jobs before their value becomes zero. It can also be observed that PBA achieves higher overall value than other approaches. Further, on an average, PBA performs better than other approaches if both the value and energy metrics are jointly evaluated as value divided by energy.

We also have computed overall value and energy consumption when number of cores at each node is varied for a fixed number of nodes. It has been observed that the value by all the approaches increases with the number of cores due to

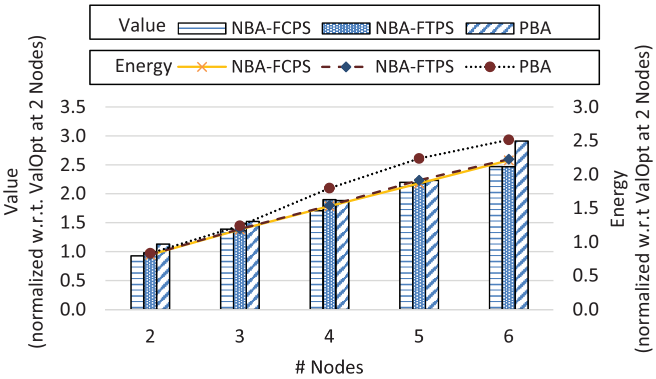

Fig. 5. Value and Energy with varying number of nodes.

TABLE I. PERCENTAGE OF REJECTED JOBS AT DIFFERENT ARRIVAL RATES

	ValOpt	NBA-FCPS	NBA-FTPS	PBA
High	49.0%	49.4%	48.8%	46.8%
Medium	29.2%	30.8%	30.0%	22.0%
Low	13.0%	12.8%	12.2%	00.0%
Average	30.4%	31.0%	30.3%	22.9%

increased processing capability. Further, PBA achieves higher overall value than other approaches and better results when both the value and energy need to be considered. However, in case advance profiling of jobs is not possible, NBA-FTPS can be employed to achieve a better trade-off between value and energy.

D. Percentage of rejected jobs

Table I shows the rejected jobs (%) at different arrival rates when various approaches are employed. The average over different arrival rates is also shown for all the approaches. The tabulated results have been computed by considering 3 nodes, where each node contains 8 cores. It can be observed that, on an average, our proposed approaches NBA-FTPS and PBA reject lesser number of jobs as compared to baseline approaches. The PBA has the lowest rejection of jobs as each job is allocated on the exact number of cores exploiting all the potential parallelism with the help of design-time profiled results. This result in cores availability for higher number of jobs before their value become zero and thus lowers rejections. It should be noted that rejection rate by PBA for low arrival rate is not always zero and varies with number of cores/nodes.

VI. Conclusions and Future Work

We have proposed value and energy optimizing resource allocation approaches for HPC data centers. We show that the approaches combine identification of efficient allocation and appropriate voltage/frequency levels to jointly optimize value and energy consumption for executing jobs containing dependent tasks. It has been shown that our approaches significantly reduce energy consumption and improve value. In future, we plan to extend our approaches to heterogeneous HPC data centers, where servers may contain different types of processing cores.

Acknowledgment

This work is funded by EU FP7 DreamCloud project under grant agreement no. 611411.

References

[1] I. Rodero, J. Jaramillo, A. Quiroz, M. Parashar, F. Guim, and S. Poole, "Energy-efficient application-aware online provisioning for virtualized clouds and data centers," in *International Green Computing Conference (IGCC)*, 2010, pp. 31–45.

[2] J. Koomey, "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, 2011.

[3] *Intel Core i7 Processor Series Datasheet, Vol. 1*, Intel Corporation, 2010, http://www.intel.com/.

[4] R. N. Calheiros and R. Buyya, "Energy-Efficient Scheduling of Urgent Bag-of-Tasks Applications in Clouds Through DVFS," in *Proceedings of IEEE International Conference on Cloud Computing Technology and Science (CLOUDCOM)*, 2014, pp. 342–349.

[5] B. Khemka, R. Friese, S. Pasricha, A. A. Maciejewski, H. J. Siegel, G. A. Koenig, S. Powers, M. Hilton, R. Rambharos, and S. Poole, "Utility maximizing dynamic resource management in an oversubscribed energy-constrained heterogeneous computing system," *Sustainable Computing: Informatics and Systems*, vol. 5, pp. 14–30, 2015.

[6] A. K. Singh, P. Dziurzanski, and L. S. Indrusiak, "Market-inspired Dynamic Resource Allocation in Many-core High Performance Computing Systems," in *IEEE International Conference on High Performance Computing & Simulation (HPCS)*, 2015, pp. 413–420.

[7] C. S. Yeo and R. Buyya, "A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing," *Softw. Pract. Exper.*, vol. 36, no. 13, pp. 1381–1419, 2006.

[8] T. Theocharides, M. K. Michael, M. Polycarpou, and A. Dingankar, "Hardware-enabled Dynamic Resource Allocation for Manycore Systems Using Bidding-based System Feedback," *EURASIP J. Embedded Syst.*, vol. 2010, pp. 3:1–3:21, 2010.

[9] C. D. Locke, "Best-effort Decision-making for Real-time Scheduling," Ph.D. dissertation, Pittsburgh, PA, USA, 1986, aAI8702895.

[10] A. M. Burkimsher, "Fair, responsive scheduling of engineering workflows on computing grids," Ph.D. dissertation, UK, 2014.

[11] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proceedings of Conference on Power Aware Computing and Systems (HotPower)*, 2008, pp. 10–10.

[12] A. Beloglazov and R. Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers," *Concurr. Comput. : Pract. Exper.*, vol. 24, no. 13, pp. 1397–1420, 2012.

[13] G. von Laszewski, L. Wang, A. Younge, and X. He, "Power-aware scheduling of virtual machines in DVFS-enabled clusters," in *Proceedings of IEEE International Conference on Cluster Computing and Workshops (CLUSTER)*, 2009, pp. 1–10.

[14] X. Ruan, X. Qin, Z. Zong, K. Bellam, and M. Nijim, "An Energy-Efficient Scheduling Algorithm Using Dynamic Voltage Scaling for Parallel Applications on Clusters," in *Proceedings of International Conference on Computer Communications and Networks (ICCCN)*, 2007, pp. 735–740.

[15] L. Wang, G. von Laszewski, J. Dayal, and F. Wang, "Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS," in *Proceedings of IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID)*, 2010, pp. 368–377.

[16] Y. Tian, C. Lin, Z. Chen, J. Wan, and X. Peng, "Performance evaluation and dynamic optimization of speed scaling on web servers in cloud computing," *Tsinghua Science and Technology*, pp. 298–307, 2013.

[17] A. K. Singh, A. Das, and A. Kumar, "Energy Optimization by Exploiting Execution Slacks in Streaming Applications on Multiprocessor Systems," in *Proceedings of ACM Design Automation Conference (DAC)*, 2013, pp. 115:1–115:7.

[18] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer, "A multi-objective approach for workflow scheduling in heterogeneous environments," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2012, pp. 300–309.

[19] I. Pietri, M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, and R. Sakellariou, "Energy-constrained provisioning for scientific workflow ensembles," in *IEEE International Conference on Cloud and Green Computing (CGC)*, 2013, pp. 34–41.

[20] D. E. Irwin, L. E. Grit, and J. S. Chase, "Balancing Risk and Reward in a Market-Based Task Service," in *IEEE International Symposium on High Performance Distributed Computing (HPDC)*, 2004, pp. 160–169.

[21] S. Eyerman and L. Eeckhout, "Fine-grained DVFS Using On-chip Regulators," *ACM Trans. Archit. Code Optim.*, vol. 8, no. 1, pp. 1:1–1:24, 2011.

[22] M. Sayuti and L. Indrusiak, "Real-time low-power task mapping in Networks-on-Chip," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2013, pp. 14–19.