ENERGY EFFICIENT RESOURCE SCHEDULING METHODOLOGIES FOR CLUSTER AND CLOUD COMPUTING

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Ahmad Fayyaz

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department:
Electrical and Computer Engineering

July 2015

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Energy Efficient Resource Scheduling Methodologies for Cluster and Cloud Computing

By

Ahmad Fayyaz

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Samee U. Khan
Chair

Jacob S. Glower

Sudarshan K. Srinivasan

Ying Huang

Approved:

07/07/2015
Date

Scott C. Smith
Department Chair

ABSTRACT

One of the major challenges in the High Performance Computing (HPC) clusters, Data Centers, and Cloud Computing paradigms is intelligent power management to improve energy efficiency. The key contribution of the presented work is the modeling of a Power Aware Job Scheduler (PAJS) for HPC clusters, such that the: **(a)** threshold voltage is adjusted judiciously to achieve energy efficiency and **(b)** response time is minimized by scaling the supply voltage. The key novelty in our work is utilization of the Dynamic Threshold-Voltage Scaling (DTVS) for the reduction of cumulative power utilized by each node in the cluster. Moreover, to enhance the performance of the resource scheduling strategies in first part of the work, independent tasks within a job are scheduled to most suitable Computing Nodes (CNs). First, our research analyzes and compares eight scheduling techniques in terms of energy consumption and makespan. Primarily, the most suitable Dynamic Voltage Scaling (DVS) level adhering to the deadline is identified for each of the CNs by the scheduling heuristics. Afterwards, the DTVS is employed to scale down the static, as well as dynamic power by regulating the supply and bias voltages. Finally, the per node threshold scaling is used attain power saving. Our simulation results affirm that the proposed methodology significantly reduces the energy consumption using the DTVS.

The work is further extended and the effect of task consolidation is studied and analyzed. By consolidating the tasks on a fewer number of servers the overall power consumed can be significantly reduced. The tasks are first allocated to suitable servers until all the tasks are exhausted. The idle servers are then turned off by using DTVS. The Virtual Machine (VM) monitor checks for under-utilized, partially filled, over-utilized, and empty servers. The VM monitor then migrates the tasks to suitable servers for execution if a set of conditions is met. By this way, many servers those were under-utilized get free and are turned off by using DTVS to

save power. Simulations results confirm our study and a substantial reduction in the overall power consumption of the cloud data center is observed.

ACKNOWLEDGEMENTS

First and foremost thanks to ALLAH ALLMIGHTY Who has helped me throughout the course of my studies. All of my knowledge, strength, courage, health, and abilities are His blessings upon me and there is no way to fulfill His right to thank Him.

Special thanks to Dr. Samee U. Khan, my advisor, for his help, guidance and innovative ideas. I offer my sincere and deep hearted gratitude to my advisor who always encouraged me, and persistently conveyed the spirit and guidance required for the research. Without his kind guidance and continuous efforts, this disquisition would not have been possible.

Special thanks to my committee members, Dr. Jacob S. Glower, Dr. Sudarshan K. Srinivasan, and Dr. Ying Huang for their support, guidance and helpful recommendations. Thanks to the Electrical and Computer Engineering staff members Jeffrey Erickson, Laura D. Dallman, and Priscilla Schlenker for all the unconditional help and favor.

I would like to thank my family. Their continuous support is always a source of motivation and encouragement for me. I especially like to thank my father and mother, who are the only and every reason for whatever I am today and whatever I achieved in my life. I also would like to thank my loving wife, Sabahat, and my sons, Qazi Abdullah and Qazi Suleman, for their patience, time, and support. They both have shown extreme patience and cooperation that resulted in successful completion of this disquisition.

Finally, I owe my heartiest thanks to all my friends and colleagues here in the US and Pakistan, who always helped me in the time of need.

DEDICATION

I would like to dedicate this thesis to my family, especially to my parents, my wife, and my sons for all the inexplicable love, support, and motivation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

In this chapter, the introduction to the research that we have performed during Ph.D. is discussed. Our research focus was on the reduction in energy consumption in clusters, data centers, and Cloud computing facilities. In our research studies, we focused on the resource scheduling techniques that can be used to minimize the energy consumption in computing paradigms. We also investigated Dynamic Power management (DPM) techniques that can be incorporated in the scheduling algorithms such that the energy consumed by these giant computing facilities is further reduced. In the first case, we incorporated Dynamic Threshold Voltage Scaling (DTVS) and Dynamic Voltage Scaling (DVS) in the scheduling algorithms and in the second study we studied the effect of task consolidation on the overall energy consumption of the cloud data center.

1.1. Energy Efficiency in Cluster and Cloud Computing using Voltage Scaling

With the enormous progression in computer technology, the need for power awareness has rapidly increased. Whether it's supercomputing center, cluster computing, or a large-scale data center, the minimization of energy consumption is a serious concern [1.1]. Implementation of energy efficient workstations is an indispensable need of the day due to the rising energy cost and environmental impacts. The demand for the reduction in energy consumption is even higher in large clusters, because the annual power budget of the cluster is approximately equal to the cost of a new server [1.2].

Cluster computing can be defined as a single system image of multiple computing resources combined together through networks, software, and hardware to handle complex computations [1.1]. The High Performance Computing (HPC) clusters are widely used to render

remarkable computing capabilities for scientific, as well as commercial applications [1.3]. Rigorous and complex research problems, such as complex image generation, molecular level design, and weather modeling can be solved using clusters, super computers, distributed computers, cloud, and grids [1.4]. The need for efficient processing of the aforementioned tasks has escalated the demand of cluster deployment to a significant level. However, despite the benefits cluster computing offers, a key challenge is the reduction in energy consumption.

Energy consumption of data centers, grids, and computer clusters is getting doubled almost every five years (since last 15 years). It is estimated that almost 50% of the operational expenses within a data center accounts for the energy cost [1.1, 1.5].

This paper presents an empirical approach to reduce response time and energy consumption while maintaining high performance using resource scheduling algorithms. The Dynamic Threshold-Voltage Scaling (DTVS) and Dynamic Voltage Scaling (DVS) are the two major techniques employed to minimize the power budget of a large scale cluster [1.6]. In the DVS, the supply voltage, V_{DD}, is scaled down to a discrete number of voltage levels without violating the task's deadline for energy saving. Alternatively, the DTVS manages both the leakage and dynamic power by adjusting the V_{DD} and bias voltage (V_{BS}) simultaneously, to improve power saving at increased activity levels. The operational voltage of the Computing Nodes (CNs) can be decreased by employing DTVS.

Previous works [1.7, 1.8] focuses solely on exploiting DVS to achieve energy savings. However, the work presented here hinges on introducing DTVS along with DVS to further minimize the energy consumption. The incorporation of DTVS and DVS produces significant improvement in energy saving regardless of the scheduling heuristic and workload. Simulation results affirm that our scheme produces better results than the current state of the art

methodologies in terms of energy consumption. Moreover, the performance of the heuristics is also improved without violating targeted deadlines of the tasks.

In the recent years, numerous techniques have been presented to minimize the power consumption in clusters. Kriourov *et al*. [1.9] and Lang *et al.* [1.10] introduced slack in the execution time of jobs to achieve the aforementioned goal. Nevertheless, the proposed technique increases the overall execution time (makespan) of jobs giving rise to a tradeoff between performance and energy consumption. The outcome of the above mentioned approach is undesirable especially in a real time job scheduling environment, where meeting the deadline is critical. A major research challenge is to focus on both the goals: **(a)** meeting deadline constraint and **(b)** minimizing the energy/power consumption of computing cluster.

This work hinges on the parametric model of Estimated Energy Dissipation (EED) to find an appropriate DVS level. Nevertheless, the deadline constraint is not compromised that makes the scheduler efficient in terms of energy consumption. To circumvent the aforementioned problems, the PAJS perform the following three steps to accomplish Energy Minimization Procedure (3EMP):

1. **Resource Allocation:** Identify the jobs (set of tasks) to be allocated to CNs and a set of task-to-node assignment is then formed**.**
2. **Resource Matching:** The algorithms dictate a CN-task pair that can meet the user defined deadline constraint.
3. **Resource Scheduling:** The algorithms determine the order of execution of tasks and the respective DTVS and DVS levels for each CN-task pair.

The key contribution of the work presented in this paper is the modeling and implementation of the PAJS, a task scheduler equipped with the DTVS and DVS modules. The notable contributions are apportioned as:

- We focus on minimizing the energy consumption using four defined DVS levels while maintaining the targeted deadlines.
- Eight heuristic based job scheduling algorithms are used to achieve energy efficient allocation of tasks to CNs.
- We have shown the adaptability of DTVS with the energy aware scheduling techniques. The results affirm that DTVS, when incorporated with DVS, further lowers the energy consumption as compared to the non-DTVS compliant version of the scheduling algorithms.

The analytical model proposed in this paper is implemented using MATLAB and analyzes eight probative task scheduling algorithms on the basis of: **(a)** makespan and **(b)** energy consumption. The PAJS methodology proposed here employs the set of eight heuristics based on greedy, recursive, and genetic algorithms. To maintain a fair comparison among the considered algorithms, the system parameters, such as the number of tasks and their respective deadlines, and execution time of tasks are kept similar. The eight heuristics performed are G-Min, G-Max, MinMax, G-Deadline, UtyFunc, ObjFunc, and two naturally motivated genetic heuristics, namely GenAlgo and GenAlgo-DVS. To investigate our simulations, variance in CNs and tasks heterogeneity is considered. To extend the scope of our analysis and evaluate the best solution on different data sets, the workload is varied from small sized workload (100 tasks) to large sized workload (100,000 tasks).

1.2. Energy Efficiency using Virtual Machine/Task Consolidation

The dual influence of increasing cloud computing data center energy consumption and increasing energy costs has raised the significance of cloud computing data center efficiency as a policy to decrease costs, accomplish size and indorse environmental responsibility. The data centers are the most integral part for most of Information Technology (IT) organizations. Many renowned organizations, such as Google, Microsoft, IMB, and Amazon have big data centers that contain thousands of computing servers around the world to provide fast and efficient cloud computing services to the customers [1.11]. The past decade has witnessed a phenomenal increase in the number of data centers, and the size of the existing data centers. The aforementioned situation have increased the word-wide power consumption that drive many research communities to carry out research on the data center energy consumption, energy efficient techniques for computing units, and power consumption prediction of the data centers [1.12-1.16]. In a study conducted by the Environmental Protection Agency (EPA) in 2006 [1.12] stated that the data centers are consuming more than 61 Tera Watt hour (TWh) of electricity per year that was 1.5% of the total power consumption of the whole US for the same year. The report also stated that the data center power consumption will have an annual growth of 16% over the next 10 years. Figure 1.1 shows the Emerson Network Power modeled energy consumption for a typical data center and evaluated how energy is used within the data center. The power usage is classified as either "demand-side" or "supply-side." Demand-side systems are the servers, storage, communications and other IT systems that support the data center business. The supply-side systems support the demand side.

The propagation of Cloud computing has stemmed in establishing large-scale data centers. The American Society of Heating, Refrigerating and Air-Conditioning Engineers

(ASHRAE) [1.17], has published a trend that by 2014, the energy and infrastructure costs of the data center will contribute about 75% in the total data center cost, while IT will contribute the remaining 25% in the overall operating cost of the data center [1.18].

The computing resources quantity and hardware power inefficiency are not the only factors that result in tremendously energy consumption in the data centers. The inefficient use of data center resources, such as CPUs and memory play a big part in the increase of energy consumption.

In [1.19], the authors collected a data from more than 5000 computing servers in a data center over a period of six-months and reported that the data center servers are usually not idle but the server utilization is rarely 100%. More than 90% of the servers were running at 10-50% utilization of their total 100% capacity. This phenomenon results in extra expenses on over provisioning that directly increase the total power consumption cost of the data center [1.19]. Moreover, handling and preserving over-provisioned data center's resources result in increased Cost of Ownership (TCO). In another study [1.20], authors reported that if the data center servers are completely idle even then the power consumption is 70% of their total peak power consumption. Therefore, it is a known conclusion that the underutilization of the data centers servers is extremely inefficient with respect to the energy consumption.

In [1.21], a comprehensive study is conducted that monitor energy consumption of Grid'5000 infrastructure. The authors reported significant opportunities for energy saving in the data center via techniques, such as switching servers on and off with respect to utilization or run the servers on low power mode. The data center's energy consumption can be reduce by switching idle servers to low-power modes, such as hibernation or sleeping, this process will reduce idle power consumption.

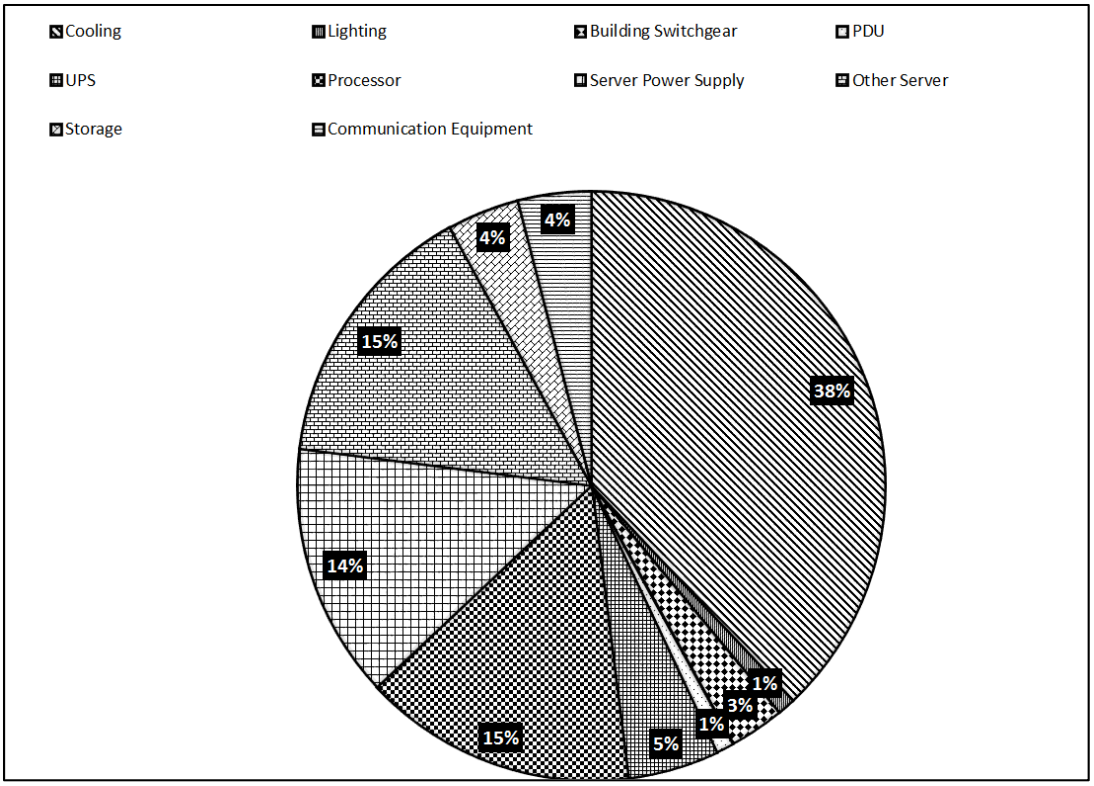

Fig. 1.1. Data Center Energy Consumption Partition

There are some other critical issues that arise from high energy consumption of computing resources, such as the power required by the cooling system operation of the data center. A report states that for each power Watt consumed by a computing entity or a server, data center has to consume another 0.5-1 Watt for the cooling system to keep the computing entity cool [1.22]. Moreover, higher the energy consumption by the data center's infrastructure, higher is the carbon dioxide (CO2) emissions that contribute to the greenhouse effect [1.23]. One simple solution for the energy inefficiency in the data centers is to involve virtualization technology [1.24]. The virtualization technology provides opportunities to the Cloud providers to create several Virtual Machines (VMs) on a single physical computing server. This virtualization phenomenon improves the resource utilization and also increases the Return On Investment

(ROI). Moreover, the use of live migration [1.25], we can dynamically consolidate the VMs to the minimal number of physical servers according to their current resource requirements.

1.3. Research Goals and Objectives

The key contribution of the work presented in this thesis is the modeling and implementation of the PAJS, a task scheduler equipped with the DTVS and DVS modules. The notable contributions are apportioned as:

- We focus on minimizing the energy consumption using four defined DVS levels while maintaining the targeted deadlines.
- Eight heuristic based job scheduling algorithms are used to achieve energy efficient allocation of tasks to CNs.
- We have shown the adaptability of DTVS with the energy aware scheduling techniques. The results affirm that DTVS, when incorporated with DVS, further lowers the energy consumption as compared to the non-DTVS compliant version of the scheduling algorithms.
- Further, Virtual Machine/task consolidation for energy minimization while taking into account the CPU power and storage capacity of the individual server and the bandwidth of the links between servers is incorporated.

1.4. References

[1.1] L. Wang, S. U. Khan, D. Chen, J. Kołodziej, R. Ranjan, C. Z. Xu, and A. Zomaya., "Energy aware parallel task scheduling in a cluster," *Future Generation Computer Systems,* vol. 29, no.7, pp. 1661-1670, Mar. 2013.

[1.2] S. Huang and W. Feng, “Energy efficient cluster computing via accurate workload characterization,” *In proceeding of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the grid; CCGRID,* IEEE S, May 2009.

[1.3] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry, “An overview of energy efficiency techniques in cluster computing systems,” *Cluster Computing*, vol. 16, no. 1, pp. 3-15, Sept. 2011.

[1.4] Z. Zong, X. Qin, X. Ruan, K. Bellam, M. Nijim, and M. Alghamdi, “Energy-efficient scheduling for parallel applications running on heterogeneous clusters,” *International Conference on Parallel Processing (ICPP 2007),* pp. 19-26, Sep. 2007.

[1.5] A. Abbas, M. Ali, A. Fayyaz, A. Ghosh, A. Kalra, S. U. Khan, M. U. S. Khan, T. D. Menezes, S. Pattanayak, A. Sanyal, and S. Usman, "A Survey on Energy-Efficient Methodologies and Architectures of Network-on-Chip," Computers and Electrical Engineering. DOI: 10.1016/j.compeleceng.2014.07.012.

[1.6] N. Mehta and B. Amrutur, “Dynamic supply and threshold voltage scaling for CMOS digital circuits using in-situ power monitor,” *IEEE Transactions on VLSI Systems*, vol. 20, no. 5, pp. 892-901, May 2012.

[1.7] S. U. Khan and N. Min-Allah, "A Goal Programming Based Energy Efficient Resource Allocation in Data Centers," Journal of Supercomputing, vol. 61, no. 3, pp. 502-519, 2012.

[1.8] N. Min-Allah, H. Hussain, S. U. Khan, and A. Y. Zomaya, "Power Efficient Rate Monotonic Scheduling for Multi-core Systems," Journal of Parallel and Distributed Computing, vol. 72, no. 1, pp. 48-57, 2012.

[1.9] A. Krioukov, C. Goebel, S. Alspaugh, Y. Chen, D.E. Culler, and R.H. Katz, “Integrating renewable energy using data analytics systems: Challenges and opportunities,” *Bulletin of the IEEE Computer Society Technical Committee.*, vol. 34, no. 1, pp. 3-11, 2011.

[1.10] W. Lang and J. M. Patel, “Energy management for map reduce cluster,” in *Proceedings of the VLDB*, vol. 3, no. 1-2, pp. 129–139, 2010.

[1.11] E. R. Masanet, R. E. Brown, A. Shehabi, J. G. Koomey, and B. Nordman. "Estimating the energy use and efficiency potential of US data centers." *Proceedings of the IEEE* 99, no. 8 (2011): 1440-1453.

[1.12] R. Brown, "Report to congress on server and data center energy efficiency: Public law 109-431." *Lawrence Berkeley National Laboratory* (2008).

[1.13] R. Bolla, F. Davoli, R. Bruschi, K. Christensen, F. Cucchietti, and S. Singh. "The potential impact of green technologies in next-generation wireline networks: Is there room for energy saving optimization?" *Communications Magazine, IEEE* 49, no. 8 (2011): 80-86.

[1.14] M. Webb, "SMART 2020: enabling the low carbon economy in the information age, a report by The Climate Group on behalf of the Global eSustainability Initiative (GeSI)." *Global eSustainability Initiative (GeSI), Tech. Rep* (2008).

[1.15] J. G. Koomey, "Worldwide electricity used in data centers. "*Environmental Research Letters* 3, no. 3 (2008): 034008.

[1.16] L. Wang, and S. U. Khan. "Review of performance metrics for green data centers: a taxonomy study." *The Journal of Supercomputing* 63, no. 3 (2013): 639-656.

[1.17] Trends, Datacom Equipment Power. "Cooling Applications." *ASHRAE, http://www.ashrae.org* (2005).

[1.18] C. L. Belady, "In the data center, power and cooling costs more than the it equipment it supports." Electronics cooling 13, no. 1 (2007): 24.

[1.19] L. A. Barroso, and U. Hölzle. "The case for energy-proportional computing." Computer 12 (2007): 33-37.

[1.20] X. Fan, W. D. Weber, and L. A. Barroso. "Power provisioning for a warehouse-sized computer." In ACM SIGARCH Computer Architecture News, vol. 35, no. 2, pp. 13-23. ACM, 2007.

[1.21] M. D. De Assuncao, J. P. Gelas, L. Lefevre, and A. C. Orgerie. "The Green Grid'5000: Instrumenting and using a Grid with energy sensors." In Remote Instrumentation for eScience and Related Aspects, pp. 25-42. Springer New York, 2012.

[1.22] . P. Ranganathan, P. Leech, D. Irwin, and J. Chase. "Ensemble-level power management for dense blade servers." In ACM SIGARCH Computer Architecture News, vol. 34, no. 2, pp. 66-77. IEEE Computer Society, 2006.

[1.23] The green grid consortium 2011. URL http://www.thegreengrid.org. (Accessed June 2015)

[1.24] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, Al. Ho, R. Neugebauer, I. Pratt, and A. Warfield. "Xen and the art of virtualization." ACM SIGOPS Operating Systems Review 37, no. 5 (2003): 164-177.

[1.25] C. Clark, K. Fraser, S. Hand, Jacob G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. "Live migration of virtual machines." In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, pp. 273-286. USENIX Association, 2005.

2. RELATED WORK

In this chapter we discuss some of the work that is related to the research we have performed during Ph.D.

2.1. Power Aware Resource Allocation in Computing Clusters

In this section some of the works related to the energy efficient resource allocation schemes in data centers and computing clusters are presented. The power management mechanisms in cluster computing can be categorized as: **(a)** Dynamic Power Management (DPM), and **(b)** Static Power Management (SPM) [2.1]. The SPM technique employs a flash storage and a pair of low power embedded CPUs to limit the peak power consumption. Lang *et al*. [2.2] exploited Fast Array of Wimpy Nodes (FAWN) for achieving the benefits offered by the SPM. The FAWN methodology promises high efficiency when tested on nodes working at lower frequency. However, while solving non-parallelizable problems and working on indivisible data set size, the FAWN is inefficient [2.2].

In the recent years, significant attempts have been made to conserve energy consumption of clusters using the DPM. The technique proposed by Chaparro-Baqueero *et al*. [2.3] speculates the resource utilization and uses software along with power scalable modules to reduce the power consumption of the system [2.4, 2.5]. For brevity, we focus on the DPM that can be categorized into: **(a)** Dynamic Voltage Scaling (DVS) [2.6, 2.7], **(b)** Dynamic Frequency Scaling (DFS), and **(c)** Dynamic Voltage and Frequency Scaling (DVFS) [2.5, 2.8]. Kolodziej *et al.* [2.9] employed DVS scaling for intelligent power management. It is noteworthy that the DVS approach scales down the voltage level, V_{DD} according to the need of the node. As soon as V_{DD} is decreased, a net decrease in the energy consumption is observed. Though the aforementioned

method yields promising results in soft real time tasks, the energy optimization may not be significant in hard real time job scheduling, where the completion time of tasks is of foremost importance [2.9]. However, our proposed scheme achieves energy efficiency without violating the targeted deadlines of the jobs.

In the DVFS approach, a node needs to be furnished with a DVFS unit to exploit the benefits of frequency and voltage scaling. A node can be categorized in either of the two modes: **(a)** active mode or **(b)** idle/sleep mode. The node with minimum activity is assigned the sleep mode, whereas the node with high activity is assigned the active mode. In case the node in active mode is overloaded, it is assigned more voltage lines and clock speed [2.10]. A recent work by Beloglazov *et al.* [2.11] and Kliazovich *et al.* [2.12] exploited sleep mode for attaining the energy saving benefit in data centers for cloud computing.

A similar approach is modeled in [2.9] and [2.13] for energy optimization using the Dynamic Voltage and Frequency Scaling (DVFS). Although the above mentioned approaches attempt to enhance energy efficiency in clusters, the DVFS is inherently limited. For each node the minimum voltage level is defined, below which the nodes operate incorrectly. The DVFS is an energy efficient technique employed by network designers to reduce the energy dissipation [2.7]. The DVFS adjust the clock frequency in real-time based on the operational voltage of a processor [2.14]. The key idea is to provide the circuit just enough of speed and voltage that is required to process the assigned workload [2.15]. The DVFS technique performs a transition from a high-power state (of the processor) to a low-power state, when the workload reduces [2.10]. The aim of the DVFS is to reduce the dynamic power [2.16]. However, the PAJS employs the DTVS technique to attain power conservation irrespective of the mode (active/idle) of the node. Therefore, our proposed work minimizes both the static and dynamic power losses.

A related approach to the DVFS is DFS. The DFS needs to be incorporated within the system utilizing it, for example speed step by Intel and VIA Technologies product called as Long Haul [2.13]. A dynamic change is made in frequency of the system and energy efficiency is achieved by scaling the clock speed dynamically. However, energy benefits of the DFS are limited, due to the fact that a reduction in the frequency reduces the speed of the system. Therefore, the overall performance is degraded. The majority of the modern day scientific work focuses on the DVS for intelligent power management [2.17].

Alfonso *et al.* [2.18] proposed the CLUES (Cluster Energy Saving System) tool, an energy management strategy for HPC clusters. The aforementioned methodology adjusts the number of working nodes by extracting information about the activity of nodes. A check on the inactivity time of the node is observed and if the check is successful, the particular node is turned off. Although the aforementioned approach reduces power consumption by minimizing the count of active nodes, it is prone to configuration issues. Moreover, such an approach might prolong the job wait time because of two reasons: **(a)** the delay incurred due to the check on inactivity of nodes and **(b)** addition of new nodes in the scheduler. Contrary to this approach, the benefit of the work presented in this paper is that it does not need to put a check on the activity level to maximize energy saving.

Valentini *et al*. [2.1] presents an extensive survey of the power management endeavors using the aforementioned methodologies. Surveys performed by Shuja *et al*. [2.14] and Usman *et al*. [2.10] present a collection of interesting examples of the recently developed Power management schemes. Nevertheless, this line of work is sound in comparison to the existing state of the art as the PAJS combines the benefits of DVS with DTVS to meet the sine qua non for high performance and energy-efficient cluster.

2.2. Energy Efficiency through VM Consolidation

A number of literature works have been published that proposed solutions to reduce carbon dioxide emission amount of the Cloud data centers. One group of researchers focused on the reduction of energy consumption in a single data center or by only considering the hardware aspects of the data centers [2.21], [2.22]. The data centers are benefited from some renowned technologies, such as virtualization [2.23]. Among the virtualization techniques, there are VMs migration [2.26] and consolidation [2.28]. However, the main issue in the VM migration or consolidation is its complexity. Moreover, the VMs resumption and suspension causes system overloading [2.24]. Furthermore, these methodologies are more of a reactive methods rather than proactive and preventive. Therefore, preventive methods are more important and effective. As stated in the introduction that an idle server consumes almost half of the power compared to the power it consumes at peak load [2.20]. The authors of [2.27] introduce a dynamic right-sizing on-line algorithm that predicts how many servers will be required to execute the arriving workload of the data center. The experimental results of [2.27] stated that dynamic right-sizing achieves significant energy savings, but the technique requires different power levels of the servers and servers should be able to transit between different states. In a similar work [2.26], Green Open Cloud (GOC) architecture is proposed that has advance resource reservation for the users to increase the prediction of the arrived requests. The aforementioned technologies are implemented within a data center and aim to decrease the energy consumption, while the technologies do not specifically consider carbon emission. The reduction in the data center energy consumption will not unavoidably reduce the carbon footprint. The works presented in [2.19] and [2.25] reflect the availability of both non-polluting and polluting energy sources in a

single data center. The techniques use prediction-based scheduling algorithms to increase usage of green energy sources.

The Green Scheduler considers the servers to be in an order [2.29, 2.30]. It then starts scheduling the tasks to first server from the pool until that server can execute no more tasks and is overloaded. The scheduler then schedules the tasks to the next server and so on. Servers that come last from the pool of servers are idle most of the time because of the fact that the tasks are scheduled to servers that come earlier in the pool. Consolidation of tasks is achieved at the time of allocation of tasks. Our work is different from Green scheduler as it has variable sized workload as compared to the fixed sized workload of Green scheduler. In addition our technique consolidates the tasks even after the allocation phase is over i.e. we migrate the tasks from one server to the other to minimize the energy consumption even if the task is in execution phase.

The DENS [2.31] methodology selects the best-fir computing resources for the execution of tasks by considering the communication potential and load level of data center components. Its aim is to achieve balance between traffic demands, job performances, energy consumed by the data center, and the job QoS requirements.

Round Robin [2.32] scheduler equally distributes the communicational and computing loads among the switches and servers. As a result no server is overloaded and the network traffic is balanced. This scheduler is least efficient in terms of energy consumption because all the switches and servers are busy most of the time. In our work the workload is exponentially distributed to mimic the real time arrival of workload. We are also incorporating consolidation of tasks whereas Round Robin is not using any type of consolidation technique.

2.4. References

[2.1] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry, "An overview of energy efficiency techniques in cluster computing systems," *Cluster Computing*, vol. 16, no. 1, pp. 3-15, Sept. 2011.

[2.2] W. Lang, S. Harizopoulos, J.M. Patel, M.A. Shah, and D. Tsirogiannis, "Towards energy-efficient database cluster design," *In: Proceedings of the VLDB Endowment (PVLDB)*, vol.5, no. 11, pp. 1684-1695, Aug. 2012.

[2.3] G. A. Chaparro-Baqueero, Q. Zhou, C. Liu, J. Tang, and S. Liu, "Power-efficient schemes via workload characterization on the Intel's single chip cloud computer," *IEEE Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW)*, pp. 999-1006, May 2012.

[2.4] H. Al-Daud, I. Al-Azzonib, and D. G. Down, "Power aware linear programming based scheduling for heterogeneous computer clusters," *In Future Generation Computer System* vol. 24, no. 5, pp. 745-754, May 2012. [Special Section: Energy Efficiency in Large Scale Distributed System]

[2.5] L. Wang, S. U. Khan, D. Chen, J. Kołodziej, R. Ranjan, C. Z. Xu, and A. Zomaya., "Energy aware parallel task scheduling in a cluster," *Future Generation Computer Systems*, vol. 29, no.7, pp. 1661-1670, Mar. 2013.

[2.6] J. Kolodziej, S. U. Khan, L. Wang, A. Byrski, N. Min-Allah, and S. A. Madani, "Hierarichal genetic based grid scheduling with energy optimization," *Journal of Cluster Computing*, vol. 16, no. 3, pp. 591-609, Sep. 2013.

[2.7] J. S. Yuan, "A novel energy efficient algorithm for cloud resource management," *International Journal of Knowledge and Language Processing,* vol. 4, no. 2, pp. 12-22, Apr. 2013.

[2.8] S. Sha, J. Zhou, C. Liu, and G. Quan, "Power and energy analysis on Intel single-chip cloud computer system," *In South Easton, Proceedings of IEEE,* pp. 1- 6, Mar. 2012.

[2.9] J. Kolodziej, S. U. Khan, L. Wang, M. Kisiel-Dorohinicki, S. A. Madani, E. Niewiadomska-Szynkiewicz, A. Y. Zomaya, and C. Z. Xu, "Security, energy and performance aware resource allocation mechanisms for computational grids," *Future Generation Computer Systems,* Oct. 2012.

[2.10] S. Usman, S. U. Khan, and S. Khan, "A comparative study of voltage/frequency scaling in NOC," *2013 IEEE International Conference on Electro/Information Technology,* May 2013.

[2.11] A. Beloglazov, J. Abawaj, and R. Buyya, "Energy aware Resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems,* vol. 28, no. 5, pp. 755-768, May 2012.

[2.12] D. Kliazovich, S. T. Arzo, F. Granelli, P. Bouvry, and S. U. Khan, "Accounting for load variation in energy efficient data centers," in *IEEE International Conference on Communications (ICC),* pp. 1154-1159, Jun. 2013.

[2.13] H. Castro, M. Villamizar, G. Sotelo, C. O. Diaz, J. E. Pecero, and P. Bouvry, "Green flexible opportunistic computing with task consolidation and virtualization," *Journal of Cluster Computing,* vol. 16, no. 3, pp. 545-557, Sep. 2013.

[2.14] J. Shuja, S. A. Madani, K. Bilal, K. Hayat, S. U. Khan, and S. Sarwar, "Energy efficient data centers," *Computing,* vol. 94, no. 12, pp. 973-994, Sep. 2012.

[2.15] C. O. Diaz, M. Guzek, J. E. Pecero, P. Bouvry, and S. U. Khan, “Scalable and energy-efficient scheduling techniques for large-scale systems,” International Conference on Computer and Information Technology (CIT ‘11), Sept. 2011, pp. 641-647.

[2.16] M. A. Aziz, S. U. Khan, T. Loukopoulos, P. Bouvry, H. Li, and J. Li, "An Overview of Achieving Energy Efficiency in On-chip Networks," International Journal of Communication Networks and Distributed Systems, vol. 5, no. 4, pp. 444-458, 2010.

[2.17] O. V. Maiuri and W. R. Moore., “Implications of voltage and dimension scaling on CMOS Testing: the multidimensional testing paradigm,” *16th IEEE Symposium on VLSI Test*, Apr. 1998.

[2.18] C. D. Alfonso, M. Caballer, F. Avarruiz, and V. Hernandez, “An energy management system for cluster infrastructures,” *Journal of Computer and Electrical Engineering*, vol. 39, no. 8, June 2013.

[2.19] B. Aksanli, J. Venkatesh, L. Zhang, T. Rosing: Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. In: Proc. of the 4th Workshop on Power-Aware Computing and Systems, pp. 5:1–5:5. ACM (2011).

[2.20] L. Barroso, U. Holzle: “The case for energy-proportional computing.” IEEE Computer 40(12), 33–37 (2007).

[2.21] A. Beloglazov, R. Buyya, Y. Lee, and A. Zomaya: A taxonomy and survey of energy efficient data centers and cloud computing systems. Advances in Computers 82(2), 47–111 (2011).

[2.22] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Dang, K. Pentikousis: Energy-efficient cloud computing. The Computer Journal 53(7), 1045–1051 (2010).

[2.23] T. Brey, L. Lamers: Using virtualization to improve data center efficiency. The Green Grid, Whitepaper 19 (2009).

[2.24] C. Clark, K. Fraser, S. Hand, Jacob G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. "Live migration of virtual machines." In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pp. 273-286. USENIX Association, 2005.

[2.25] I. Goiri, R. Beauchea, K. Le, T. Nguyen, M. Haque, J. Guitart, J. Torres, R. Bianchini: Greenslot: scheduling energy consumption in green datacenters. In: Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 20:1–20:11. ACM (2011).

[2.26] E. Harney, S. Goasguen, J. Martin, M. Murphy, M. Westall: The efficacy of live virtual machine migrations over the internet. In: Proc. of the 2nd International Workshop on Virtualization Technology in Distributed Computing. ACM (2007) 16. Lef`evre, L., Orgerie, A.: Designing and evaluating an energy efficient cloud. The Journal of Supercomputing 51(3), 352–373 (2010).

[2.27] M. Lin, A. Wierman, L. Andrew, E. Thereska: Dynamic right-sizing for powerproportional data centers. In: Proc. of the IEEE INFOCOM, pp. 1098–1106. IEEE (2011).

[2.28] S. Srikantaiah, A. Kansal, F. Zhao: Energy aware consolidation for cloud computing. In: Proc. of the 2008 Conference on Power Aware Computing and Systems, p. 10. USENIX Association (2008).

[2.29] D. Kliazovich, P. Bouvry, and S. U. Khan. "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers." *The Journal of Supercomputing* 62, no. 3 (2012): 1263-1283.

[2.30] D. Kliazovich, S. T. Arzo, F. Granelli, P. Bouvry, and S. U. Khan. "e-STAB: energy-efficient scheduling for cloud computing applications with traffic load balancing." In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pp. 7-13. IEEE, 2013.

[2.31] D. Kliazovich, P. Bouvry, and S. U. Khan. "DENS: data center energy-efficient network-aware scheduling." *Cluster computing* 16, no. 1 (2013): 65-75.

[2.32] B. P. Rimal, E. Choi, and I. Lumb. "A taxonomy and survey of cloud computing systems." In *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pp. 44-51. Ieee, 2009.

3. POWER-AWARE RESOURCE ALLOCATION IN COMPUTER CLUSTERS USING DYNAMIC THRESHOLD VOLTAGE SCALING AND DYNAMIC VOLTAGE SCALING: COMPARISON AND ANALYSIS

This paper[1] has been published in *Journal of Cluster Computing in Springer in 2015*. The authors of the paper are Ahmad Fayyaz, Kashif Bilal, Saeeda Usman, and Samee U. Khan.

3.1. Introduction

With the enormous progression in computer technology, the need for power awareness has rapidly increased. Whether it's supercomputing center, cluster computing, or a large-scale data center, the minimization of energy consumption is a serious concern [3.33]. Implementation of energy efficient workstations is an indispensable need of the day due to the rising energy cost and environmental impacts. The demand for the reduction in energy consumption is even higher in large clusters, because the annual power budget of the cluster is approximately equal to the cost of a new server [3.13].

Cluster computing can be defined as a single system image of multiple computing resources combined together through networks, software, and hardware to handle complex computations [3.33]. The High Performance Computing (HPC) clusters are widely used to render remarkable computing capabilities for scientific, as well as commercial applications [3.32].

[1] The material in this chapter was co-authored by Ahmad Fayyaz, Kashif Bilal, Saeeda Usman, and Samee U. Khan. Ahmad Fayyaz had primary responsibility for designing the system model, conducting experiments and collecting results. Ahmad Fayyaz also drafted and revised all versions of this chapter.

Rigorous and complex research problems, such as complex image generation, molecular level design, and weather modeling can be solved using clusters, super computers, distributed computers, cloud, and grids [3.35]. The need for efficient processing of the aforementioned tasks has escalated the demand of cluster deployment to a significant level. However, despite the benefits cluster computing offers, a key challenge is the reduction in energy consumption. Energy consumption of data centers, grids, and computer clusters is getting doubled almost every five years (since last 15 years). It is estimated that almost 50% of the operational expenses within a data center accounts for the energy cost [3.1, 3.33].

This paper presents an empirical approach to reduce response time and energy consumption while maintaining high performance using resource scheduling algorithms. The Dynamic Threshold-Voltage Scaling (DTVS) and Dynamic Voltage Scaling (DVS) are the two major techniques employed to minimize the power budget of a large scale cluster [3.27]. In the DVS, the supply voltage, V_{DD}, is scaled down to a discrete number of voltage levels without violating the task's deadline for energy saving. Alternatively, the DTVS manages both the leakage and dynamic power by adjusting the V_{DD} and bias voltage (V_{BS}) simultaneously, to improve power saving at increased activity levels. The operational voltage of the Computing Nodes (CNs) can be decreased by employing DTVS.

Previous works [3.17, 3.28] focuses solely on exploiting DVS to achieve energy savings. However, the work presented here hinges on introducing DTVS along with DVS to further minimize the energy consumption. The incorporation of DTVS and DVS produces significant improvement in energy saving regardless of the scheduling heuristic and workload. Simulation results affirm that our scheme produces better results than the current state of the art

methodologies in terms of energy consumption. Moreover, the performance of the heuristics is also improved without violating targeted deadlines of the tasks.

In the recent years, numerous techniques have been presented to minimize the power consumption in clusters. Kriourov *et al*. [3.21] and Lang *et al.* [3.23] introduced slack in the execution time of jobs to achieve the aforementioned goal. Nevertheless, the proposed technique increases the overall execution time (makespan) of jobs giving rise to a tradeoff between performance and energy consumption. The outcome of the above mentioned approach is undesirable especially in a real time job scheduling environment, where meeting the deadline is critical. A major research challenge is to focus on both the goals: **(a)** meeting deadline constraint and **(b)** minimizing the energy/power consumption of computing cluster.

This work hinges on the parametric model of Estimated Energy Dissipation (EED) to find an appropriate DVS level. Nevertheless, the deadline constraint is not compromised that makes the scheduler efficient in terms of energy consumption. To circumvent the aforementioned problems, the PAJS perform the following three steps to accomplish Energy Minimization Procedure (3EMP):

1. **Resource Allocation:** Identify the jobs (set of tasks) to be allocated to CNs and a set of task-to-node assignment is then formed**.**
2. **Resource Matching:** The algorithms dictate a CN-task pair that can meet the user defined deadline constraint.
3. **Resource Scheduling:** The algorithms determine the order of execution of tasks and the respective DTVS and DVS levels for each CN-task pair.

The key contribution of the work presented in this paper is the modeling and implementation of the PAJS, a task scheduler equipped with the DTVS and DVS modules. The notable contributions are apportioned as:

- We focus on minimizing the energy consumption using four defined DVS levels while maintaining the targeted deadlines.
- Eight heuristic based job scheduling algorithms are used to achieve energy efficient allocation of tasks to CNs.
- We have shown the adaptability of DTVS with the energy aware scheduling techniques. The results affirm that DTVS, when incorporated with DVS, further lowers the energy consumption as compared to the non-DTVS compliant version of the scheduling algorithms.

The analytical model proposed in this paper is implemented using MATLAB and analyzes eight probative task scheduling algorithms on the basis of: **(a)** makespan and **(b)** energy consumption. The PAJS methodology proposed here employs the set of eight heuristics based on greedy, recursive, and genetic algorithms. To maintain a fair comparison among the considered algorithms, the system parameters, such as the number of tasks and their respective deadlines, and execution time of tasks are kept similar. The eight heuristics performed are G-Min, G-Max, MinMax, G-Deadline, UtyFunc, ObjFunc, and two naturally motivated genetic heuristics, namely GenAlgo and GenAlgo-DVS. To investigate our simulations, variance in CNs and tasks heterogeneity is considered. To extend the scope of our analysis and evaluate the best solution on different data sets, the workload is varied from small sized workload (100 tasks) to large sized workload (100,000 tasks).

The remainder of the paper is arranged as follows. Related work is elaborated in Section 3.2. The problem formulation, system model, and implementation details of the PAJS are presented in Section 3.3. Discussion related to all of the heuristics is detailed in Section 3.4. Section 3.5 presents the simulation results and their comparison, while in Section 3.6 concluding remarks are presented.

3.2. Related Work

The power management mechanisms in cluster computing can be categorized as: **(a)** Dynamic Power Management (DPM), and **(b)** Static Power Management (SPM) [3.32]. The SPM technique employs a flash storage and a pair of low power embedded CPUs to limit the peak power consumption. Lang *et al*. [3.22] exploited Fast Array of Wimpy Nodes (FAWN) for achieving the benefits offered by the SPM. The FAWN methodology promises high efficiency when tested on nodes working at lower frequency. However, while solving non-parallelizable problems and working on indivisible data set size, the FAWN is inefficient [3.22].

In the recent years, significant attempts have been made to conserve energy consumption of clusters using the DPM. The technique proposed by Chaparro-Baqueero *et al*. [3.10] speculates the resource utilization and uses software along with power scalable modules to reduce the power consumption of the system [3.11, 3.33]. For brevity, we focus on the DPM that can be categorized into: **(a)** Dynamic Voltage Scaling (DVS) [3.19, 3.34], **(b)** Dynamic Frequency Scaling (DFS), and **(c)** Dynamic Voltage and Frequency Scaling (DVFS) [3.30, 3.33]. Kolodziej *et al.* [3.20] employed DVS scaling for intelligent power management. It is noteworthy that the DVS approach scales down the voltage level, V_{DD} according to the need of the node. As soon as V_{DD} is decreased, a net decrease in the energy consumption is observed. Though the aforementioned method yields promising results in soft real time tasks, the energy

optimization may not be significant in hard real time job scheduling, where the completion time of tasks is of foremost importance [3.20]. However, our proposed scheme achieves energy efficiency without violating the targeted deadlines of the jobs.

In the DVFS approach, a node needs to be furnished with a DVFS unit to exploit the benefits of frequency and voltage scaling. A node can be categorized in either of the two modes: **(a)** active mode or **(b)** idle/sleep mode. The node with minimum activity is assigned the sleep mode, whereas the node with high activity is assigned the active mode. In case the node in active mode is overloaded, it is assigned more voltage lines and clock speed [3.31]. A recent work by Beloglazov *et al.* [3.8] and Kliazovich *et al.* [3.18] exploited sleep mode for attaining the energy saving benefit in data centers for cloud computing.

A similar approach is modeled in [3.20] and [3.9] for energy optimization that uses the Dynamic Voltage and Frequency Scaling (DVFS). Although the above mentioned approaches attempt to enhance energy efficiency in clusters, the DVFS is inherently limited. For each node the minimum voltage level is defined, below which the nodes operate incorrectly. The DVFS is an energy efficient technique employed by network designers to reduce the energy dissipation [3.19]. The DVFS adjust the clock frequency in real-time based on the operational voltage of a processor [3.29]. The key idea is to provide the circuit just enough of speed and voltage that is required to process the assigned workload [3.12]. The DVFS technique performs a transition from a high-power state (of the processor) to a low-power state, when the workload reduces [3.31]. The aim of the DVFS is to reduce the dynamic power [3.7]. However, the PAJS employs the DTVS technique to attain power conservation irrespective of the mode (active/idle) of the node. Therefore, our proposed work minimizes both the static and dynamic power losses.

A related approach to the DVFS is DFS. The DFS needs to be incorporated within the system utilizing it, for example speed step by Intel and VIA Technologies product called as Long Haul [3.9]. A dynamic change is made in frequency of the system and energy efficiency is achieved by scaling the clock speed dynamically. However, energy benefits of the DFS are limited, due to the fact that a reduction in the frequency reduces the speed of the system. Therefore, the overall performance is degraded. The majority of the modern day scientific work focuses on the DVS for intelligent power management [3.26].

Alfonso *et al.* [3.3] proposed the CLUES (Cluster Energy Saving System) tool, an energy management strategy for HPC clusters. The aforementioned methodology adjusts the number of working nodes by extracting information about the activity of nodes. A check on the inactivity time of the node is observed and if the check is successful, the particular node is turned off. Although the aforementioned approach reduces power consumption by minimizing the count of active nodes, it is prone to configuration issues. Moreover, such an approach might prolong the job wait time because of two reasons: **(a)** the delay incurred due to the check on inactivity of nodes and **(b)** addition of new nodes in the scheduler. Contrary to this approach, the benefit of the work presented in this paper is that it does not need to put a check on the activity level to maximize energy saving.

Valentini *et al*. [3.32] presents an extensive survey of the power management endeavors using the aforementioned methodologies. Surveys performed by Shuja *et al*. [3.29] and Usman *et al*. [3.31] present a collection of interesting examples of the recently developed Power management schemes. Nevertheless, this line of work is sound in comparison to the existing state of the art as the PAJS combines the benefits of DVS with DTVS to meet the sine qua non for high performance and energy-efficient cluster.

3.3. Problem Formulation

In this section, we discuss the basic parameters that affect the energy consumption of a cluster. For a CN two main modules are required to function: (**a)** voltage line and (**b)** clock speed, referred as frequency [3.24]. To address energy consumption issue, we want the system to be designed in a method that the above stated components of the framework are controlled judiciously. Both voltage and frequency share a linear relation (Eq. 3.1), such as, decreasing the voltage decreases frequency. As a result, fewer computation cycles per unit time increase the execution time of the tasks. The time taken by the cluster to execute all of the assigned tasks is referred to as makespan in the paper.

$$f \propto V \quad . \tag{3.1}$$

The total power consumption P_t can be scripted as:

$$P_t = P_d + P_s \ , \tag{3.2}$$

where P_d is the dynamic power and P_s is the static power.

The dynamic power in Eq. (3.3) signifies that power is a quadratic function of voltage. This implies that a reduction in the supply voltage of the system will reduce the power consumption in a quadratic function

$$P_d = A \times V_{DD}^2 \times f_{CLK} \times C_{EFF} \ , \tag{3.3}$$

where, f_{CLK} is the clock frequency, V_{DD} is the supply voltage, A represents the activity factor, and C_{EFF} is the effective switched capacitance.

The energy consumption in a cluster is calculated using Eq. (3.4). To reduce the energy consumption, either the instantaneous power should be reduced or the computation time ought to

be decreased. Therefore, power factor becomes a significant metric in the design of energy efficient clusters. Consequently, the focus should be on the power management.

$$Energy = Power \times Computation\ Time. \tag{3.4}$$

The DTVS procedure is illustrated with the help of a flow chart shown in Fig. 3.1. The motivation of using a power tracking mechanism is to monitor power consumption of each CN in the cluster. The power tracker/activity indicator module ascertains that the total power consumption level is within the tolerable bounds. When the total power consumed, P_{total} crosses its acceptable bounds, the DTVS module asserts the V_{TH} and V_{DD} scaling modules simultaneously, so as to regulate the total power P_t of the cluster.

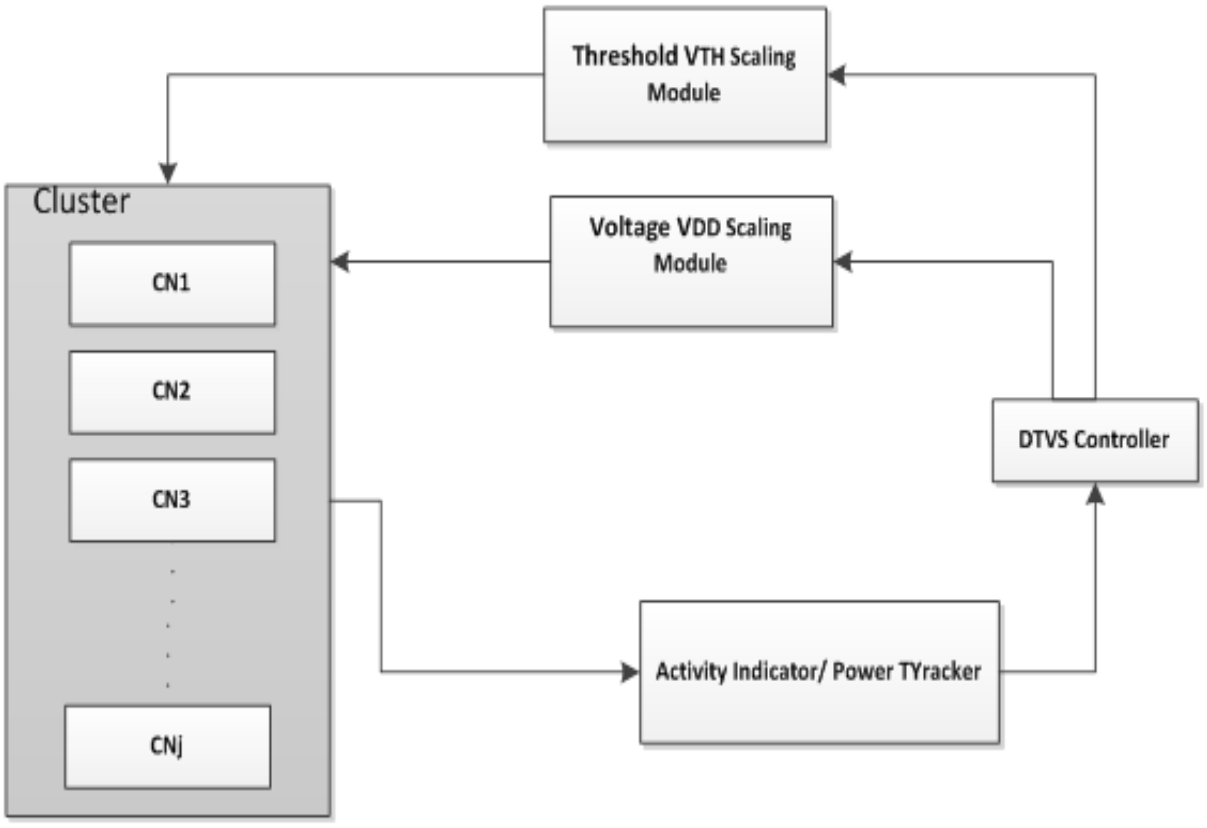

Fig. 3.1. DTVS Mechanism

Table 3.1. Notation/ Acronyms and their Meanings

Symbols	**Meaning**	**Symbols**	**Meaning**
DPM	Dynamic Power Management	G-Min	Greedy Heuristic that schedule shortest task first
DVS	Dynamic Voltage Scaling		
PAJS	Power-Aware Job Scheduler	G-Max	Greedy Heuristic that schedule longest task first
4EMP	4-Step Energy Minimization Procedure	G-Deadline	The tasks with the shortest deadlines are scheduled to the CNs first in this Greedy Heuristic
t_{ij}	Run time of task t_i on CN_j		
CN	Computing Node	MinMax	Greedy Heuristic that schedule tasks to the least efficient CNs
V_{DD}	CN supply Voltage		
M	Makespan	ObjFunc	Greedy Heuristic in which objective functions are employed to govern task allocation
J	Set of all t_i		
t_i	i^{th} task $\in J$	UtyFunc	Greedy Heuristic that schedule tasks based on a utility function
CVB	Coefficient of variation based		
$dead_i$	Deadline of t_i	GenAlgo	Genetic Algorithm
m_{ti}	Memory requirement of t_i	GenAlgo-DVS	Genetic Algorithm that utilizes DVS
m_{CNj}	Memory Available to CN_j	C_i	Computational Cycles required by t_i
DVS_k	k^{th} DVS level	$I\text{-}ETE$	Index for ETE Matrix
EED	Estimated Energy Dissipation	t_{ijk}	Run time of task t_i on CN_j at DVS_k
ETE	Estimated Time of Execution	m_j	Run time of computing node CN_j
CN	Set of CNs in CN allocation	$Energy_{idle}$	Energy Consumed when CN is idle
CN_p	Set of CNs in CN pool	$Energy_j$	Energy Consumed by CNj
CN_j	j^{th}CN $\in CN$	$DTVS$	Dynamic Threshold and Voltage Scaling (DTVS)
p_{ij}	Power Consumed by CN_j to execute t_i	μ_{task}	Average task execution time
E_{sol}	The best solution	V_{task}	Variance in execution time of tasks
C_j	Power Consumption of CN_j	NIS	Naturally Inspired Solutions
k_{idle}	Power scalar for idle CN	V_{CN}	Variance in computing node heterogeneity

The power tracker/activity indicator also monitors if any CN in the cluster is idle. V_{TH} of the idle CNs is increased and V_{DD} is scaled down to a level that the CN is turned off, resulting in less power consumption. A list of acronyms and their meanings is summarized in Table 3.1.

3.3.1. The System Model

We present a holistic model of a high performance cluster in this section. The aforementioned is comprised of a collection of CNs and works on a set of tasks, referred as a job.

Computing Nodes: The set of CNs in the cluster is denoted as, CN= {CN_1, CN_2…CN_m}. It is assumed that each of the CNs is endowed with a DTVS as well as a DVS module. For the problem catered in this paper, we assume a constant and negligible transition time between successive levels of DVS as considered in [3.24].

Each CN is described by:

- The instantaneous power dissipation of the CN, C_j. The C_j may alter between C_j^{min} to C_j^{max}, depending on the DVS level of the CN. The range of C_j is thereof defined as $0 < C_j^{min} < C_j^{max}$.
- The specific computing node memory is abbreviated as m_{CNj}.

Tasks: Consider a job, J, which is a collection of tasks. The set J= {t_1, t_2, ……, t_n}, where i_{th} task is represented by t_i. Each task in the job is characterized by:

- The deadline, $dead_i$ that needs to be fulfilled by the task.
- The computational cycles, C_i that a task needs to accomplish.
- The memory requirement of the task, m_{ti}.

Preliminaries: We are given a set of CNs and a metaset of tasks, J. Each member of the set J has to be allocated to a CN such that the deadline constraint is not violated. A realistic task to node allocation is accomplished when:

1. The runtime m_j, of CN_j, is less than or equal to $dead_i$.
2. Each task, $t_i \in J$ is mapped to at least one CN_j if all the related constraints of each task are fulfilled.
3. For a successful mapping, $m_{CNj} > m_{ti}$ is satisfied. In case the aforementioned inequality is not satisfied, t_i cannot be assigned to CN_j.

3.3.2. Modeling the Energy-Makespan Minimization Problem

Given a metaset of tasks J, and a pool of CNs, the problem statement such that:

- The net power utilized by the CNs is minimized.
- A minimization in makespan, M, of the metaset of tasks, J is achieved.

The mathematical model of the problem statement can be expressed as:

$$min \sum_{i=1}^{n} C_{ij} x_{ij} \quad \text{such that} \quad minmax\ 1 \le j \le m \sum_{i=1}^{n} t_{ij} x_{ij}$$

Obliged to the subsequent constraints from 3.5 through 3.9:

$$x_{ij} \in \{0,1\}, \text{ where } i = 1,2,\ldots m \text{ and } \quad j = 1,2,\ldots. n \tag{3.5}$$

$$If\ t_i \rightarrow m_j, \forall_i, \forall_j : \text{such that } m_{CNj} > m_{ti}, \text{ then } x_{ij}, \tag{3.6}$$

$$(t_{ij}x_{ij} \le dead_i) \in \{0,1\}, \tag{3.7}$$

$$t_{ij}x_{ij} \le dead_i, \forall_i, \forall_j, x_{ij} = 1, \tag{3.8}$$

$$\prod_{i=1}^{n} \left(t_{ij} x_{ij} \le\ dead_i\right) = 1, \forall_i, \forall_j, x_{ij} = 1. \tag{3.9}$$

A task, t_i, from a job is assigned to CN_j when the mapping constraint (3.5) equals 1.This can be expressed as when x_{ij} =1. Constraint (3.6) juxtaposes an additional constraint on the mapping procedure. The aforementioned mapping can take place only when the CN_j satisfy the memory requirement of *ti*. The Boolean relationship between the deadline of the task and its actual time of completion is depicted in constraint (3.7). Constraint (3.8) is adherence to the individual task deadline. The deadline constraint of the metaset is shown in (3.9) and is logically true if for each $t_i \in J$, the deadline, $dead_i$, is satisfied. The main aim of PAJS is energy minimization in computational clusters. In this paper, we discuss how to reduce makespan and minimize the system's overall power consumption at the same time. The aforementioned parameters make the PAJS problem formulation a multi objective and multi constrained optimization problem. The formulation is similar to the Power Aware Task Allocation (PATA) and Energy-Aware Task Allocation (EATA), except for the additional constraints of the DTVS. The major difference between the PAJS and the mentioned techniques is that the domain of PAJS is wider in terms of capacity of resources and computations performed.

3.4. Heuristics Implementation and Evaluation

This section presents the execution behavior of the eight algorithms analyzed and implemented in the implementation of the PAJS. Fig. 3.2 depicts the control of the eight scheduling heuristics employed for the analysis process. The Estimated Time of Execution (ETE) matrix gives the task completion time on a specific node. The number of rows in the ETE depends on the tasks count in J, and the number of entries in CN_p indicates the column length of the ETE. Each entry (i,j) of the ETE matrix corresponds to the estimated execution time of task *i* on node *j*. In the ETE matrix, the entries in a row signify the estimated completion time of a particular task on different nodes whereas, elements along a column depict the estimated

completion time of various tasks on a specified node. The Coefficient of Variance Based (CVB) method is employed for the generation of the ETE matrix [3.4]. The three parameters used to introduce heterogeneity in generation of the ETE matrix are:

- The variance in Computing Nodes heterogeneity, V_{CN}.
- The variance in execution time of tasks, V_{task}.
- The average completion time of each $t_i \in J$, μ_{task}.

The above listed heterogeneity parameters are incorporated in the generation of ETE matrix to imitate a workload that is supported in previous studies and is derived from real world applications [3.2, 3.4, 3.5, 3.16, and 3.24]. The probability distribution function used by the CVB is a gamma distribution, so it is necessary to define the shape parameter, α, and scale parameter, β. The mean of the gamma distribution is $\mu = \beta\alpha$ and the variance is, $V = 1/\sqrt{\alpha}$.

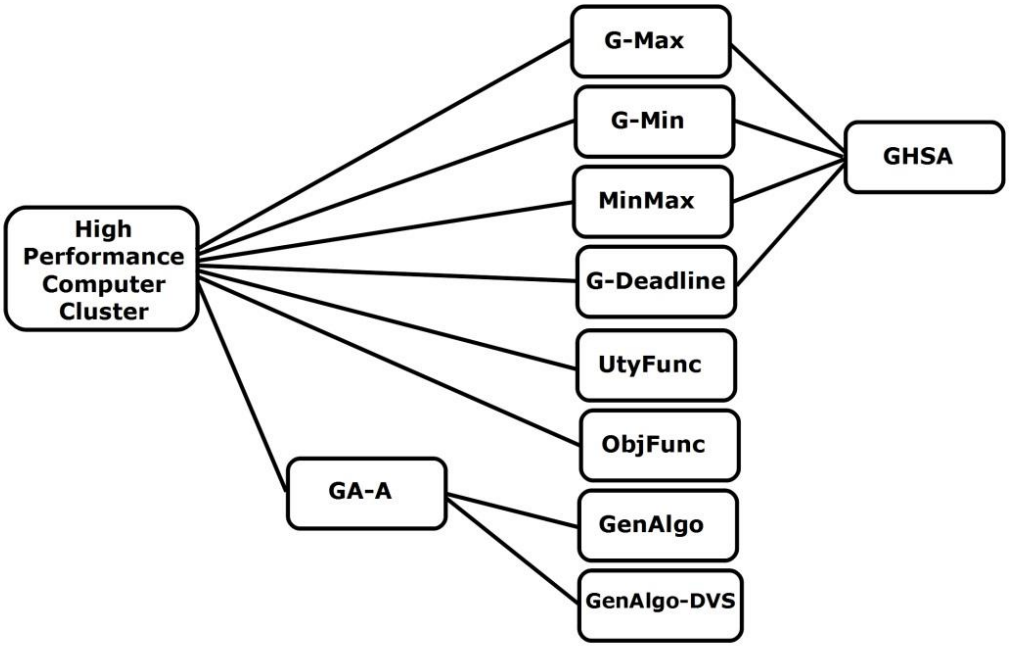

Fig. 3.2. Scheduling Heuristics used by the PAJS

However, the parameters α_{task}, $\alpha_{CN}, \beta_{task},$ and β_{CN} in the gamma distribution are interpreted as $\mu_{task}, V_{task},$ and V_{CN} in this paper.

$$\alpha_{task} = 1/V_{task}^2, \tag{3.10}$$

$$\alpha_{CN} = 1/V_{CN}^2, \tag{3.11}$$

$$\beta_{task} = \mu_{task}/\alpha_{task}, \tag{3.12}$$

$$\beta_{CN} = G(\alpha_{task}, \beta_{task})/\alpha_{CN}, \tag{3.13}$$

where $G(\alpha_{task}, \beta_{task})$ is the gamma distribution's sample number.

The deadline, $dead_i$ for each task within a job $t_i \in J$ is derived from the ETE matrix and is given as

$$dead_i = \frac{|t_i|}{|CN|} \times arg_j\, max\big(ETE(i,j)\big) \times k_d, \tag{3.14}$$

where parameter k_d is used to tighten the deadline $dead_i$ [3.24].

3.4.1. Greedy Heuristics

3.4.1.1. Greedy Heuristic Scheduling Algorithm (GHSA)

The GHSA accomplishes resource scheduling for the algorithms discussed in the subsequent sections, i.e. Section 3.4.1.2 to 3.4.1.5. The methodology used by each of the above-mentioned heuristics for scheduling J to CN varies. The pseudo-code of GHSA is presented in Algorithm 3.1. The inputs of the GHSA encompass $ETE, CNp,\ dead_i\ \forall\ t_i\ \in J,$ and CN. GHSA produces a *J* to *CN* mapping, M, and E_{sol} subject to the inputs defined.

The GHSA initiates the mapping procedure by assigning the first entry (task) of the ETE matrix to the best suitable *CN*. To ensure a feasible mapping the $dead_i$ constraint, stated in line 4 should be satisfied. The mapping of t_i to CN_j is to adhere to the minimum possible level of DVS

scaling, DVS_K (Table 3.2). Starting from DVS_1, the DVS_K is increased when $dead_i$ is not satisfied. The DVS_K at which $dead_i$ constraint agrees, no further increments are applied to DVS_K. In case the $dead_i$ is not met by the task even at the extreme DVS level (DVS_4), then the particular t_i is allocated to the next CN in the ETE matrix by GHSA. Line 10 illustrates that if the deadline is not compiled by GHSA in scheduling t_i on any of the CNs, then a flag, d_{flag} is activated. The d_{flag} indicates the absence of a feasible solution for a specific t_i. After the successful assignment of t_i to a *CN*, both the execution time of t_i and the energy consumption of CN_j are recorded. Line 6 corresponds to the addition of ETE (i,j) to m_j.

Table 3.2. Power Scales and Operational Speeds for Different DVS Levels

DVS Level	Power Scale	Operational Speed
1	0.3430	70%
2	0.5120	80%
3	0.7290	90%
4	1.0000	100%

Line 7 signifies the addition of EED (i,j) to E_{sol}. For any feasible solution obtained, t_i to *CN* mapping is processed and the energy consumed is computed. The idle time energy in Line 16 is calculated by equation (3.15) that is:

$$Energy_{idle} = CN_j \times time_{idle} \times k_{idle} . \quad (3.15)$$

Where k_{idle} is the DVS_K level of an idle *CN* and $time_{idle}$ is given by the equation (3.16) that is:

$$time_{idle} = M - m_j . \quad (3.16)$$

From line 18 to 23, the run time t_{ijk}, for each task $t_i \in J$ on a particular CN at a specific DVS_k level, is compared with the run time, m_j, of that CN and the overall makespan, M. If t_{ijk} is greater than m_j but less than M, this indicates that the CN_j is idle. To conserve energy at this idle time of the CN_j, the DTVS mechanism is employed to the node.

3.4.1.2. G-Min

The name of the algorithm 3.2 suggests that the tasks are greedily scheduled based on the shortest task first and is named as G-Min. The goal of executing the tasks in aforementioned manner is to introduce a slack in scheduling of the tasks. The main benefit of using a slack is that it allows the scheduler to schedule the subsequent tasks with longer run-times without violating their individual deadlines. Input parameters of G-Min comprise of an ETE matrix. The output is thereof a rearranged ETE matrix, an EED matrix, and I-ETE. In Algorithm 3.2, R signifies a row in ETE matrix, whereas, C_i stands for the i^{th} column in ETE matrix.

It is interesting to note that MinMax G-Deadline, G-Min, and G-Max share identical input and output parameters. Once the elements of the ETE matrix are rearranged, the GHSA is applied to the ETE matrix for evaluation purposes. The minimum power consumption is achieved by computing ETE for different DVS levels using the DVS methodology.

3.4.1.3. G-Max

The greedy heuristic that prefers to execute the longest task first is termed as G-Max and the scheduling mechanism is depicted in Algorithm 3.3. Therefore, the leftover tasks to be scheduled are the ones with the shorter execution times. First, each row is rearranged in ascending order. Then the rows are swapped in such a manner that the first column of the resulting ETE matrix is arranged in descending order. The crux behind this scheduling scheme is that the tasks with shorter execution times are more easily scheduled by GHSA without contravening the deadline constraint.

Algorithm 3.1: Greedy Heuristic Scheduling Algorithm (GHSA)

Input: $ETE, CN_p, dead_i \forall t_i \in j, CN$

Output: $J\ to\ CN\ mapping, E_{sol}, M$

foreach $t_i \in j$ **do**
  $CN_j \in CN$ **do**
  for $DVS_k = 1\ to\ 4$ **do**
    if $t_{ijk} + m_j \leq dead_i$ **then**
      Assign $t_i\ to\ CN_j\ at\ DVS_k$;
      $m_j \leftarrow m_j + ETE(ij)$;
      $E_{sol} \leftarrow E_{sol} + EED(ij)$;
    end
  end
  if t_i not assigned **then**
    $d_{flag} \leftarrow 1$;
    EXIT;
  end
end
foreach $CN_j \in CN$ **do**
  $E_{sol} \leftarrow E_{sol} + Energy_{idle}$;
end
foreach t_{ijk}
  if $t_{ijk} > m_j$ and $t_{ijk} < M$
    DTVS = 0;
  else
    DTVS = 1;
end

Algorithm 3.2: G-Min

Input: *ETE*

Output: *ETE, EED, I-ETE*

1: **foreach** row**,** $R \in ETE$ **do**

2: Sort *R* and corresponding row in I-ETE in ascending order;

3: **end**

4: $\forall R \in ETE$, interchange *R's* such that C_1 is in ascending order;

5: Make similar changes in *I-ETE* with respect to step 4*;*

6: INVOKE GHSA;

Algorithm 3.3: G-Max

Input: *ETE*

Output: *ETE, EED, I-ETE*

1: **foreach** row**,** $R \in ETE$ **do**

2: Sort *R* and corresponding row in I-ETE in ascending order;

3: **end**

4: $\forall R \in ETE$, interchange *R's* such that C_1 is in descending order;

5: Make similar changes in *I-ETE* with respect to step 4*;*

6: INVOKE GHSA;

Algorithm 3.4: G-Deadline

Input: *ETE*

Output: ETE, EED, I-ETE

1: **foreach** row, $R \in ETE$ **do**

2: Sort *R* and corresponding row in I-ETE in ascending order according to each t_i's $dead_i$;

3: **end**

4: $\forall R \in ETE$, interchange *R's* such that C_1 is in ascending order on the basis of vector $dead_i$;

5: Make similar changes in *I-ETE* with respect to step 4 *;*

6: INVOKE GHSA;

3.4.1.4. G-Deadline

In algorithm 3.4 the tasks are scheduled based on the criterion of their respective deadlines and is named as G-Deadline. The tasks scheduled first are the ones with the shortest deadlines. The task with less sensitive deadline constraint can be lingered on and scheduled later in the scheduling order. The working of the algorithm demands a two-step reordering procedure of the ETE matrix: **(a)** Initially, the ETE matrix is arranged such that the individual rows are in ascending order and **(b)** then a swapping of rows is done such that the elements in the first column are assembled in ascending order based on the task's deadline. The GHSA is invoked once the ETE matrix is re-arranged, by the G-deadline scheduling routine.

3.4.1.5. MinMax

Scheduling of tasks in algorithm 3.5 is achieved on the basis of efficiency of CNs and is named as MinMax. The initial phase of MinMax allocated the task to the least efficient CNs. The main goal is to introduce a slack in the scheduling process. The next phase schedules the subsequent tasks on the most efficient CNs. Firstly, the rows in the ETE matrix are ordered in descending order. Finally, the swapping of rows is performed such that the first column is aligned in descending order based on task completion times.

Algorithm 3.5: MinMax

Input: *ETD*

Output: ETD, EED, I-ETE

1: **foreach** row, $R \in ETE$ **do**
2: Sort *R* and corresponding row in I-ETE in ascending order;
3: **end**
4: $\forall R \in ETE$, interchange *R's* such that $C1$ is in ascending order;
5: Make similar changes in *I-ETE* with respect to step 4*;*
6: INVOKE GHSA;

3.4.1.6. ObjFunc

To ascertain a power efficient task to CN mapping the greedy heuristic objective function is employed, abbreviated as ObjFunc in the paper. This heuristic utilizes the following two objective functions for the above stated purpose: **(a)** task selection and **(b)** node selection. Algorithm 3.6 depicts the pseudo-code for ObjFunc. The inputs to ObjFunc comprises of $ETE, CNp,\ dead_i\ \forall\ t_i\ \in J,$ and CN. The output renders $J\ to\ CN$ mapping, M and the most optimized energy solution, E_{sol}. The algorithm starts with the selection of a task for which a Select Task array (ST) is generated. The ST array has an entry for every t_i that is to be assigned to one of the CN for the above said purpose. The following objective function is used for selection:

$$ST = \alpha_1\left(T_{2,i} - T_{1,i}\right) + \alpha_2\left(N_{2,k} - N_{1,k}\right) + \alpha_3 \frac{T_{1,i}+T_{2,i}}{\sum_{j=1}\left(T_{1,j}+T_{2,j}\right)} + \alpha_4 + \alpha_5 + \alpha_6\ , \quad (3.17)$$

where $T_{1,i}$ and $T_{2,i}$ represent the minimal and second minimal estimated execution time of task, t_i, respectively. $N_{1,k}$ is the most power-efficient and $N_{2,k}$ is the second most power-efficient CN for the execution of task t_i, respectively.

The components α_1, α_2, and α_3 are the weight parameters and components α_4, α_5, and α_6 are scalars added to ST array in case the following conditions are satisfied:

- α_4 is added to the ST array if the most power-efficient CN is also the one with shortest t_{ij}.
- α_5 is added to the ST array if the second most power-efficient CN is also the one with shortest t_{ij}, or *vice-versa*.
- The scalar α_6 is added to the ST array if the second most power-efficient CN and the CN with second shortest t_{ij} are same.

Line 4 depicts the sorting of ST in descending order to ensure that ObjFunc schedules the most appropriate task first. In order to determine the most suitable CN for each task, Line 7 is coded and the result is recorded in CN select array, SN.

For every task the selection function assigns a value to each CN using the objective function given by the following equation.

$$SN = \beta_1 T_{1,CN_{k,ti}} + \beta_2 N_{1,CN_{k,ti}} + \beta_3 load(CN_k)\,, \tag{3.18}$$

where $T_{1,CN_{k,ti}}$ corresponds to the completion time of task t_i on node CN_k, $N_{1,CN_{k,ti}}$ corresponds to the instantaneous power consumption when task t_i is executed on node CN_k and *load* (CN_k) is a value dependent on certain conditions. If t_i satisfies $dead_i$, the value of load (CN_k) equals zero otherwise, it is equal to m_j-$dead_i$. The intention behind this is to assign t_i to the CN having the lowest SN value. The lowest level DVS_k that CN_j can be assigned is determined by ObjFunc at line 12. Once t_i is scheduled on CN_j, the energy consumption of CN_j and the completion time of t_i must be recoded. Next, line 13-14 adds *ETE (ij)* to m_j and *EED (ij)* to E_{sol}, respectively. If the condition at Line 10 is not satisfied even when CN_j runs at the highest DVS_k, then this indicates that a workable solution does not exit, and a flag is set.

The net energy consumption of the solution is evaluated in case a feasible solution is achieved. The values of parameters α_1 to α_6 and β_1 to β_3 are presented in Table 3.3. In lines 25 to 30, DTVS is employed to ensure power aware scheduling. The status of each CN_k is evaluated, if the check result in an idle status for the CN_k, then DTVS is set to zero. Otherwise the value of DTVS is set to one.

Algorithm 3.6: ObjFunc

Input: $ETE, CNp, dead_i \ \forall \ t_i \ \in J$, and CN

Output: J to CN mapping E_{sol}, M

```
foreach ti ∈ J do
    Calculate STi
end
Sort ST in descending order;
foreach ti ∈ ST do
    foreach CNj ∈ CN do
        Calculate SNij;
    end
    j ← arg_j min(SNij)
    for DVS_k = 1 to 4 do;
        if t_ijk + m_j ≤ dead_i then
            Assign t_i to CN_j at DVS_k;
            m_j ← m_j + ETE(ij);
            E_sol ← E_sol + EED(ij);
        end
    end
    if t_i not assigned then
        d_flag ← 1;
        EXIT
    end
end
foreach CNj ∈ CN do
    E_j ← E_j + E_idle;
end
foreach t_ijk
    if t_ijk > m_j and t_ijk < M
        DTVS = 0;
    else
        DTVS = 1;
    end
end
```

3.4.1.7. UtyFunc

For the task to CN assignment the heuristic utilizes a utility function and is abbreviated as UtyFunc. The purpose of using the utility function is that it determines the advantage obtained from each task to CN allocation. Algorithm 3.7 depicts the pseudo-code of UtyFunc. The inputs to UtyFunc are $dead_i \; \forall \; t_i \in J$, ETE matrix, CN, and instantaneous power for each node. The output of UtyFunc is the J to CN mapping, $Energy_{sol}$, and makespan of the most energy efficient node. In Line 4, the UtyFunc iterates to compute the utility of every task for each of the CN and the corresponding DVS level. To correlate speed and execution time, the utility function is employed. The speed and utility can be depicted as:

$$Speed\ (V) = \frac{K_1(V-V_t)^2}{V}, \tag{3.19}$$

$$Utility\ =\ (Speed)^p\ \times (Time)^q, \tag{3.20}$$

In eq. 3.20, p ascertains the comparative significance of the speed whereas q represents the relative importance of the completion time of the task. Based on the highest utility yielded by the CN and DVS level, the process of task assignment is concluded. However, it is noteworthy to mention that the assignment of t_i to the highest utility CN_j does not ensure the fulfillment of the deadline constraint. As depicted in Line 15, the utility of the particular t_i-CN_K pair is set to zero if the deadline is violated. The UtyFunc in such a scenario recognizes the CN-DVS level pair for the particular task that guarantees the next highest utility and allocates the task to the identified CN-DVS level pair. For a successful task allocation the power consumption of CN_j and execution time of t_i are added to E_{sol} and m_j, respectively. A d_{flag} is set when a task is unable to be assigned to any of the CNs even though the deadline is satisfied. The main point behind enabling the d_{flag} is to indicate an absence of a feasible solution.

Algorithm 3.7: UtyFunc

Input: $ETE, CNp, dead_i \; \forall \; t_i \in J$, and CN

Output: $J \; to \; CN, E_{SOL}, M$

```
foreach t_i ∈ J do
    foreach CN_j ∈ CN do
        for DVS_k = 1 to 4 do
            Calculate Ut_ijk;
        end
    end
end
foreach t_i ∈ J do
    j, k ← arg j, k max(Ut_ijk);
    if t_ijk + m_j ≤ dead_i then
        Assign t_i to CN_j at DVS_k;
        m_j ← m_j + ETE_(ij);
        E_sol ← E_sol + EED_(ij);
    else
        Ut_ij_k ← 0;
    end
    if t_i not assigned then
        d_flag ← 1
        EXIT
    end
end
foreach CN_j ∈ CN do
    E_sol ← E_sol + E_idle;
end
foreach t_ijk
    if t_ijk > m_j and t_ijk < M
        DTVS = 0;
    else
        DTVS = 1;
    end
end
```

Table 3.3. Parameters used in Task Select Array and CN Select Array

Parameters	Value
β_1	0.0970764
$\beta 2$	0.4008180
$\beta 3$	0.7734070
$\alpha 1$	0.5206560
$\alpha 2$	0.3819580
$\alpha 3$	0.0431519
$\alpha 4$	0.1605830
$\alpha 5$	0.5223390
$\alpha 6$	0.6965640

In case an acceptable solution is found the UtyFunc algorithm, then determines the net energy consumption in a similar manner as GHSA and ObjFunc. The application of DTVS is in the same way as discussed for GHSA and ObjFunc.

3.4.2. Genetic Algorithms

A class of evolutionary algorithm that employs searching and optimization techniques is known as Genetic algorithms. The main objective is to find the most suitable solution among a randomly generated population. In order to achieve the above stated objective, we first randomly generate an initial population of solutions using the genetic algorithm. The genetic algorithm performs the following four steps recursively until it encounters a halting condition.

1. Solution Ranking: The solutions ranking is based on their Distance from Origin (DFO).
2. Solution Grading: The solutions with the same DFO are then checked for their Distance from Line (DFL).
3. Reproduction: The solutions with the highest ranking are improved through *mutation* and *crossover.*

4. Substitution: The solutions with lowest rank are discarded and substituted by the newly produced solutions.

The traversal and completion of the above stated steps, completes a generation. The halting condition is verified soon after the appraisal step. In genetic algorithm, each solution produced by the algorithm is characterized by a chromosome. Fig. 3.3 represents a chromosome.

The chromosome depicts information about each task and its mapping on a specific CN. A CN may be assigned more than one task. The scheduling order of the task recalls the Step 3 of EMP and is attained by invoking GenAlgo or GenAlgo-DVS. Both of these are explained in the subsequent text.

The *PAJS* framework utilizes distance from origin (DFO) approach to rank the solutions produced. One particular solution dominates the other solution if its DFO value is less as compared to others. The PAJS framework is multi-Objective and examines two metrics: **a)** makespan, and **b)** energy consumption.

A generalized multi-objective optimization function can expressed as follows:

$$min\, F(y) = (f_1(y), f_2(y), \dots\dots\dots, f_n(y))^T \quad | \, y \in S$$

$$y = (y_1, y_2, \dots\dots\dots, y_n)^T, \tag{3.21}$$

where $(y_1, y_2, \dots\dots\dots, y_n)$ are the optimization parameters, $f_1(y), f_2(y), \dots\dots, f_n(y)$ are the objective functions, and *S* is the parameter or solution space. The solution space *S* is mapped onto *Y*, by *F*. Fig. 3.4 shows the dominated solutions. The solutions with the same DFO are compared by taking DFL, which is a 45˚ angle line, to ensure an equal importance of each metric. Moreover, PAJS is also capable of producing diversity in its objectives weights by modifying the slope of the straight line employed to determine DFL.

The solutions are ranked based on the Pareto optimality that uses non-dominated sorting for the ranking purpose [3.14]. First, the non-dominated solutions are figured out. These solutions are given the highest rank i.e. they are represented by one and removed from the population. Now, in the reduced set of population the same procedure is repeated to identify the non-dominated solutions. The result obtained by doing so is given a rank two, and the solutions with this particular rank are removed from the population pool. This process is repetitive and it goes on until the entire population of solutions is ranked as shown in Fig. 3.4.

To maintain genetic diversity the genetic algorithm employs two genetic operators: **a)** Crossover and **b)** Mutation. The genetic operator, crossover, is a used to exchange information between two parent chromosomes. GA-Algo performs a two cut-point crossover technique. The same points are selected in both the parent chromosomes to be the cut points.

Swapping of information is performed amongst the two cut points to create child chromosomes as illustrated in Fig. 3.5. It is evident from the child chromosomes that the swapping of tasks has been performed between the two cut points in the parent chromosome. Mutation is a genetic operator that ensures diversity in solution along the generations. In order to figure out the best solution, the tasks in a solution are reassigned randomly to a different CN. This strategy incorporates heterogeneity and explores the most energy saving mapping of a task onto a CN.

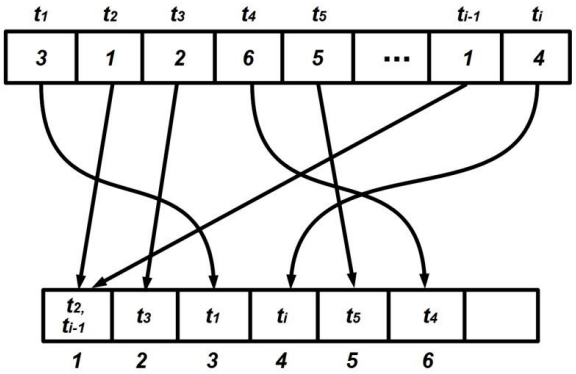

Fig. 3.3. Chromosome Representation

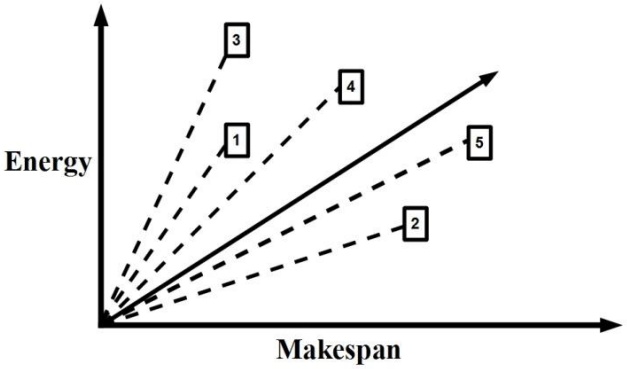

Fig. 3.4. Representing Pareto Ranking in the PAJS Framework

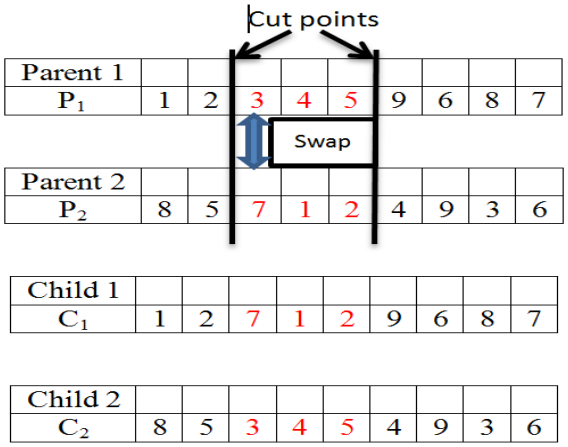

Fig. 3.5. Crossover

3.4.2.1. GA-Algo

To maintain diversity in the generated population, algorithm 3.8 is employed and is termed as GA-Algo. In order to provide effective guidance in choosing the best solution cluster of CNs are created. Algorithm 3.8 illustrates the procedure of GA-Algo.

- *Initializations*

GA-A algorithm comprises of inputs such as ETE matrix, $dead_i \; \forall \; t_i \; \in J$, CN_p, and instantaneous power. Initial solutions are randomly generated by GA-A (Line2) and then GenAlgo or GenAlgo-DVS is invoked for the evaluation of the initial solutions generated.

- *Solution Evaluation (Line 3-5)*

Either of the GenAlgo or GenAlgo-DVS is invoked, the initial solutions are evaluated until the halting condition, and the two energy minima's are calculated: **a)** the local energy minima, E_{min}, and **b)** the global energy minima, $\mathcal{E}_{min}$ for every generation.

- *Aggregating diversity (Line 10-14)*

The solutions with the most dominating rank are chosen for reproduction using genetic operator's crossover or mutation. However, in each cluster $Clus_m$, the solutions with lowest rank are discarded and replaced by the newly reproduced high-ranked solutions. As previously described GA-A involves GenAlgo or GenAlgo-DVS for the evaluation of the solution. In case a solution with energy efficiency better than the previous solutions is found, zero is assigned to k_2 to terminate the inner while loop and $Energy_{min}$ is updated. Otherwise the value of k_2 is incremented.

Algorithm 3.8: GA-Algo

Input: $ETE,\ CNp,\ dead_i \forall\ t_i\ \in\ J$

Output: $J\ to\ CN\ mapping,\ \mathcal{E}_{min},\ M$

```
1:  Generate Clus;
2:  Initial Solutions Generation;
3:  INVOKE GenAlgo / GenAlgo-DVS;
4:  while k1 ≤ k1,max do
5:      while k2 ≤ k2,max do
6:          Ranking of Solutions;
7:          Solution Reproduction carried out;
8:          INVOKE GenAlgo/GenAlgo-DVS;
9:          if any Sn ∈ Clusm Improved then
10:             k2 ← 0;
11:         else
12:             INCREMENT k2
13:         end
14:     end
15:     if Any Clusm ∈ Clus Improved then
16:         k1 ← 0;
17:     else
18:         INCREMENT k1;
19:     end
20:     Rank Clusters;
21:     Cluster Reproduction carried out;
22:     INVOKE GenAlgo/GenAlgo-DVS;
23: end
```

- *Cluster Ranking (Lines 15-23)*

In any of the cluster if the local energy minima become less than the global minima, k_1 is assigned zero. Otherwise the value k_1 is incremented. Cluster ranking based on the solution

quality is performed. Considering the cost of nodes making the clusters, cluster domination, $Clus_{dom}$ is used to rank the clusters. Mathematically, C_{dom} can be expressed as:

$$Clus_{dom}(a,b) = max\big(x \in NIS(x)\big) \sum_{y \in NIS(b)} DOM(x,y)\,, \quad (3.22)$$

$$Clus_{rank}[a] = \sum_{b \in C, \forall a \neq b} Clus_{dom}(a,b). \quad (3.23)$$

The above expressions reveal that the sum of $Clus_{dom}$ value for each cluster can be used to rank the clusters. Once the clusters are ranked, the reproduction of cluster is performed in a similar manner to reproduction of solutions, to establish diversity in out experimental setup, the reproduced clusters are improved using mutation and crossover to report the best chromosome as the final solution.

3.4.2.2. GenAlgo

To cater the task scheduling and evaluation of solutions produced by GA–Algo (step 3-4 of EMP) GenAlgo is employed. Algorithm 3.9 depicts the pseudo-code for GenAlgo. The input consists of an *ETE matrix, dead*$_i$ $\forall$ $t_i \in J$, *Clus*, CN_p and *CN*. The output are E_{S_n} and *M*.

- *Solution Evaluation (Lines 1-3)*

Before scheduling the task on the nodes available in each cluster *Clus*, every solution in *Clus* is evaluated iteratively. Because the task scheduling is not carried out yet, so the energy consumption of $Solu_n$ is set as zero.

- *Task Scheduling (Line 4-8)*

Iterations are performed through $Solu_n$ to figure out the most appropriate CN. It is observable that feasibility of CN is satisfied only if the deadline constraint is not violated. When a successful allocation takes place, the energy dissipation and runtime of t_i is added to CN_j*'s* total energy consumption E_{S_n} and runtime (m_j) respectively.

- *Invalid Solutions: (Line 9-10)*

Taking into consideration the deadline constraint, a solution is termed as invalid if the aforementioned condition is not satisfied.

- *DTVS Mechanism: (Line 15-20)*

It can be gathered from the above description of GenAlgo that it does not consider CN's DVS level. The DTVS mechanism is employed to conserve the energy of the nodes that are in idle mode.

3.4.2.3. GenAlgo-DVS

In Algorithm 10 the CN's DVS module is exploited and is termed as GenAlgo-DVS due to the fact that DVS levels are used in the algorithm. The technique used for task evaluation and scheduling is similar to GenAlgo, with the exception that the mode power level is now governed by its DVS level.

The main intuition behind GenAlgo-DVS is that it will enable the programmer to analyze the energy consumption (using DVS module) and compare it to GenAlgo. Though the input and output parameters of both GenAlgo and GenAlgo-DVS are the same, except that the later uses two additional variables. The two variables introduced are t_a and k. The variable k determines the DVS level and t_a determines whether the task has been allocated or not.

If a task cannot be assigned at any of the DVS level due to the deadline constraint, then it is said to be as invalid. A zero value of t_a signifies that the task has not been scheduled to any CN, however a 1 determines a successful assignment.

From line 23 to 28, the run time t_{ijk}, for each task $t_i \in J$ on a particular CN at a specific DVSk level, is compared with the run time, mj, of that CN and the overall makespan, M. If t_{ijk} is

greater than m*j* but less than M, this indicates that the CN*j* is idle. To conserve energy at this idle time of the CN*j* DTVS mechanism is employed on the node.

Algorithm 3.9: GenAlgo

Input: $ETE, EED,\ deadi\ \forall\ ti\ \in J, Clus,$ and J

Output: $E_{S_n}\ \forall\ S_n \in Clus$ and M

```
1:  foreach Clus_m ∈ Clus do
2:      foreach Solu_N ∈ C_m do
3:          Energy_{S_n} ← 0;
4:          foreach t_i ∈ J do
5:              if ETE(ij) + m_j ≤ dead_i then
6:                  Assign t_i to CN_j;
7:                  m_j ← m_j + ETE(ij)
8:                  E_{S_n} ← E_{S_n} + EED(ij);
9:              else
10:                 Solu_n is invalid;
11:             end
12:         end
13:     end
14: end
15: foreach t_ijk
16:     if t_ijk > m_j and t_ijk < M
17:         DTVS = 0;
18:     else
19:         DTVS = 1;
20:     end
21: end
```

Algorithm 3.10: GenAlgo-DVS

Input: $ETE, EED,\ di\ \forall ti\ \in T, C,$ and T

Output: $E_{sol}\ \forall\ S_n \in Clus$ and M

```
foreach Clus_m ∈ Clus do
    foreach S_N ∈ Clus_m do
        E_{S_n} ← 0;
        foreach t_i ∈ T do
            t_a ← 0;
            k ← 1;
            while t_a = 0 & k ≤ 4 do
                if t_ijk + m_j ≤ d_i then
                    Assign t_i to CN_j at DVS_k;
                    t_a ← 1;
                    m_j ← m_j + t_ijk
                    E_{S_n} ← E_{S_n} + EED(ij);
                else
                    k ← k + 1;
                end
            end
            if t_a = 0 then
                Solu_n is invalid;
            end
        end
    end
end
foreach t_ijk
    if t_ijk > m_j and t_ijk < M
        DTVS = 0;
    else
        DTVS = 1;
    end
end
```

3.5. Simulation Results and Discussion

The MATLAB is very efficient at performing operations and solving large sized matrices [3.25]. Therefore, the heuristics discussed in this paper were implemented using MATLAB. The ETE matrices used in our simulations had dimensions as large as 100,000 tasks by 20 Machines. The simulations were performed on 3.4 GHz Core i7 processor with 8 GB of RAM.

The objectives for our simulation study are

- To measure the Energy consumption and Makespan for different sets of tasks on a pool of machines using the eight scheduling heuristics.
- To measure and compare the results of energy consumption of the discussed eight heuristics while employing DTVS and without employing DTVS.
- To study the effect of change in system parameters on the performance of all heuristics. The summary of system parameters is presented in Table 3.4.

Table 3.4. Summary of System Parameters

System Parameters		
Deadline scaling variable	k_d	(1, 1.3, 1.8)
Variance in task execution time	V_{task}	(0.1, 0.15, 0.35)
Variance in CN heterogeneity	V_{CN}	(0.1, 0.15, 0.35)
Number of tasks (workloads)	|T|	(100; 1000; 10,000; 100,000)
Average execution time	μ_{task}	10
Number of Computing Nodes	|CN|	20
Number of DVS levels employed	DVS_k	4
Number of DTVS levels employed	$DTVS_k$	2

3.5.1. Workload

The simulation experiments are divided in three categories on the basis of workload (the total number of tasks) namely small sized, medium sized, and large sized workloads. The simulations that were executed for 100 and 1000 tasks were placed in small sized workloads. The simulations executed for 10,000 tasks were placed in medium sized workloads, and for large sized workloads 100,000 tasks were considered in the simulations. The GenAlgo and GenAlgo-DVS were not computed for the large and medium sized workloads because of their large execution times [3.24].

The workload ETE matrix was generated using CVB ETE generation method as discussed in section 3.4. The variance in tasks, V_{task}, and variance in CNs, V_{CN}, ranged between 0.1 and 0.35 while a value of 10 was set for μ_{task}, the mean task execution time as in previous studies [3.24]. To incorporate the variance in the generation of ETE workload, the parametric values are chosen from real world applications as mentioned in [3.2, 3.4, 3.5, 3.16, 3.24]. As discussed in section 3, the deadline scaling parameter k_d is used to induce heterogeneity in deadlines $dead_{i,}$ and is ranged from 1 to 1.8 [3.24]. The number of computing nodes, CNs, was set to 20 for all workloads in our simulations [3.4]. The number of DTVS and DVS levels was chosen to be 2 and 4, respectively [3.24]. The number of DTVS levels can be increased but having 2 DTVS levels serves better due to the fact that the CNs are either busy in executing some task or they are idle. The reduction in energy consumption is the main goal when the Computing Nodes (CNs) are idle. In the active mode the major energy savings are achieved by using various DVS levels.

3.5.2. Results and Discussion

The simulations were carried out on small, medium, and large sized workloads by varying three parameters namely, V_{CN}, V_{task}, and k_d. The aforementioned parameters are varied for our simulations and their results are compared. The simulations were carried out fifteen times for each set of parameters (27 sets of parameters are obtained by varying V_{CN}, V_{task}, and k_d), making a total of 405 simulations for all of the workloads and parameters. Table 3.5 summarizes the workloads and their respective set of parameters used for simulations.

The graphs of the simulation results give a great deal of information. The bold black horizontal line in all of the following graphs represents the grand mean of all of the heuristics. The black box represents the mean of the individual heuristic. The ±1 and ±1.5 times the standard deviation is represented by the gray box and black whiskers, respectively. Whereas the hexagons represent the outliers and the asterisk is used to represent extremes.

Table 3.5. Summary of Workloads and Parameters Considered for Simulations

Category	**No. of tasks, No. of CNs**	**System parameters**
Small sized workloads	100 tasks, 20 CNs	$V_{CN} = 0.1$, $V_{task} = 0.1$ and $k_d = 1$
	100 tasks, 20 CNs	$V_{CN} = 0.1$, $V_{task} = 0.1$ and $k_d = (1, 1.3, 1.8)$
	1000 tasks, 20 CNs	$V_{CN} = 0.1$, $V_{task} = 0.1$ and $k_d = 1$
	1000 tasks, 20 CNs	$V_{CN} = 0.35$, $V_{task} = 0.35$ and $k_d = 1.8$
Medium sized workloads	10,000 tasks, 20 CNs	$V_{CN} = 0.1$, $V_{task} = 0.35$ and $k_d = 1$
	10,000 tasks, 20 CNs	$V_{CN} = 0.1$, $V_{task} = 0.1$ and $k_d = 1$
Large sized workloads	100,000 tasks, 20 CNs	$V_{CN} = 0.1$, $V_{task} = 0.1$ and $k_d = 1$

3.5.2.1. Small Sized Workloads

100 tasks*:*

We started the evaluation of the heuristics with the small sized workload. Fig. 3.6 shows the makespan comparison of all of the heuristics for 100 tasks. GenAlgo performed best for this workload while MinMax and UtyFunc were amongst the worse, when comparing makespan. The GenAlgo depicts the least makespan than any of the other heuristics, because GenAlgo does not employ DVS methodology. The G-Min, G-Max, and G-Deadline employ DVS scheme and exhibit almost similar results in terms of makespan. In terms of energy consumption without using the DTVS scheme on this workload, the comparison is depicted in Fig. 3.7. The G-Max, G-Deadline, and G-Min utilized the least amount of energy and produced similar results when scheduling 100 tasks. ObjFunc produced results with minimum standard deviation and having a lower amount of maximum and minimum energy consumption. MinMax, ObjFunc, UtyFunc, and GenAlgo all had average energy consumption that was more than the grand mean. It can be observed that all the heuristics produced results with minimum outliers due to the small sized workload.

Introducing the DTVS scheme in all of the heuristics rendered lower mean energy consumption. The DTVS scheme improves mean energy consumption for all of the heuristics by 31.09%. The results using the DTVS for 100 tasks are depicted in Fig. 3.8. The UtyFunc reported lower minimum and maximum energy consumption values and had highest standard deviation. These simulations were executed for a tighter deadline scaling parameter k_d, leaving less slack to be utilized for DTVS. The G-Min, G-Max, and G-Deadline have similar results with minimum mean energy consumption for scheduling 100 tasks. All heuristics using the DTVS had improved average energy consumption as compared to the results without using the DTVS. The

tasks having shortest execution times are allocated first when G-Min greedy heuristic is employed, while G-Max heuristic schedules those tasks first that have longest execution times. The tasks with lowest deadline d_i are scheduled first when G-Deadline is employed.

The results for energy consumption for these heuristics are compared and depicted in Fig. 3.9 and Fig. 3.10 with different values of deadline scaling parameter k_d. The lower the value of the deadline scaling parameter k_d, tighter are the deadlines for the tasks.

The mean energy consumption with and without using the DTVS decreases when deadline scaling parameter (k_i) is varied from 1 to 1.3, and mean energy consumption slightly increases when k_d is varied from 1.3 to 1.8. From aforementioned discussion, it can be inferred that when k_d is increased from 1 to 1.3, then mean energy consumption for G-Deadline, G-Min, and G-Max is decreased because of the fact that they make better use of the DVS and DTVS as compared to their counterparts. The MinMax heuristic reported large energy consumption because the tasks are scheduled to least efficient CNs first. On contrary, the mean energy consumption for these heuristics is slightly increased by varying k_d ($1.3 \leq k_d \leq 1.8$) because of the fact that the use of DVS for very loose deadlines results in very large makespan.

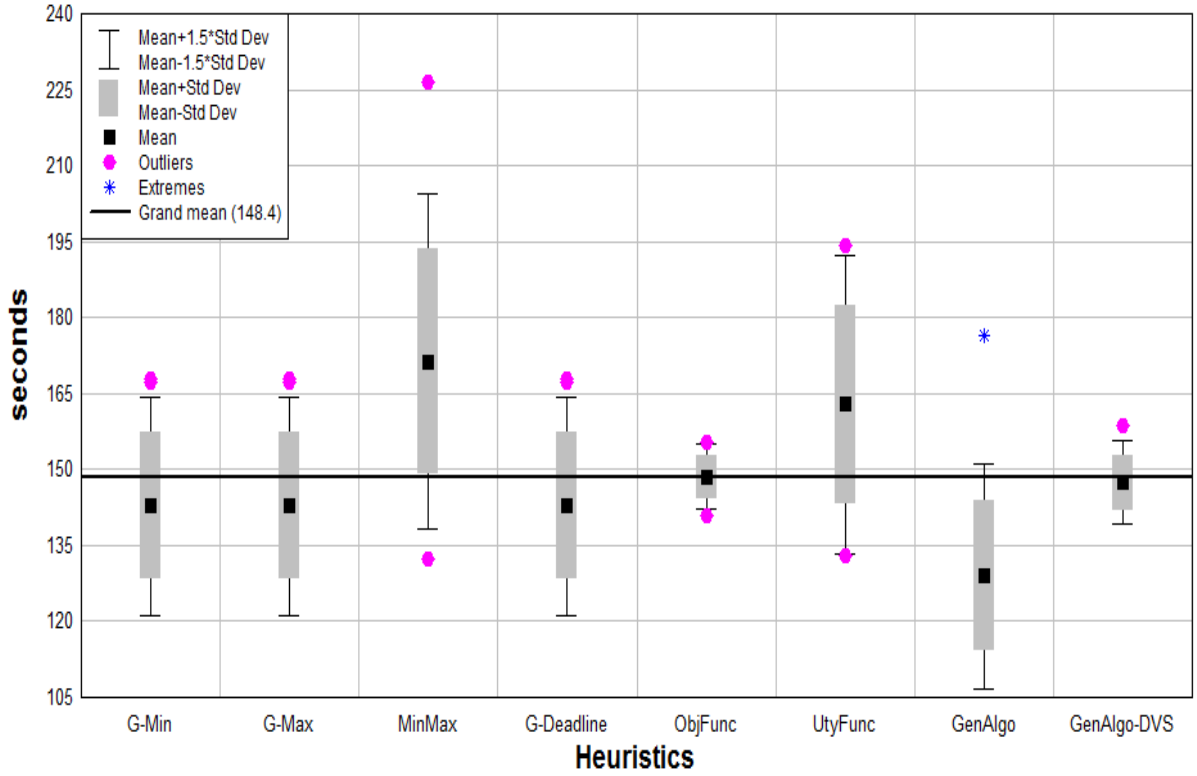

Fig. 3.6. Makespan for 100 Tasks (Vtask = 0.1, VCN = 0.1, kd = 1)

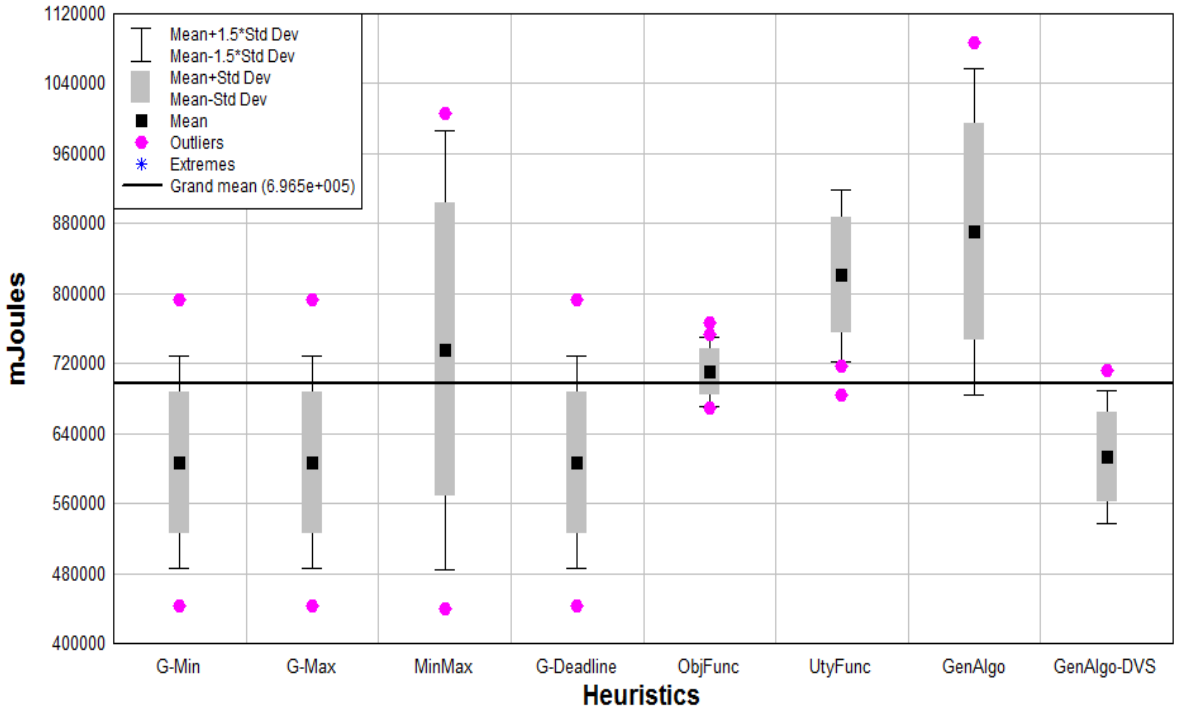

Fig. 3.7. Energy Consumption (without DTVS) for 100 Tasks (Vtask = 0.1, VCN = 0.1, kd = 1)

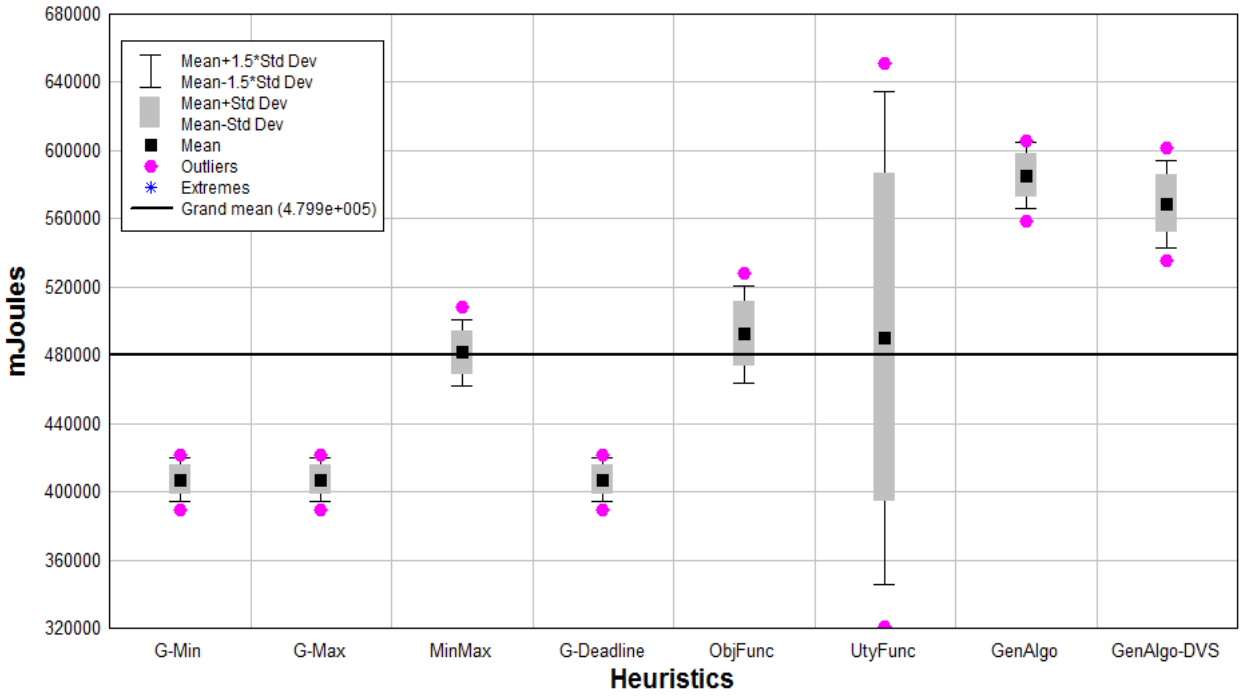

Fig. 3.8. Energy Consumption (with DTVS) for 100 Tasks (Vtask = 0.1, VCN = 0.1, kd = 1)

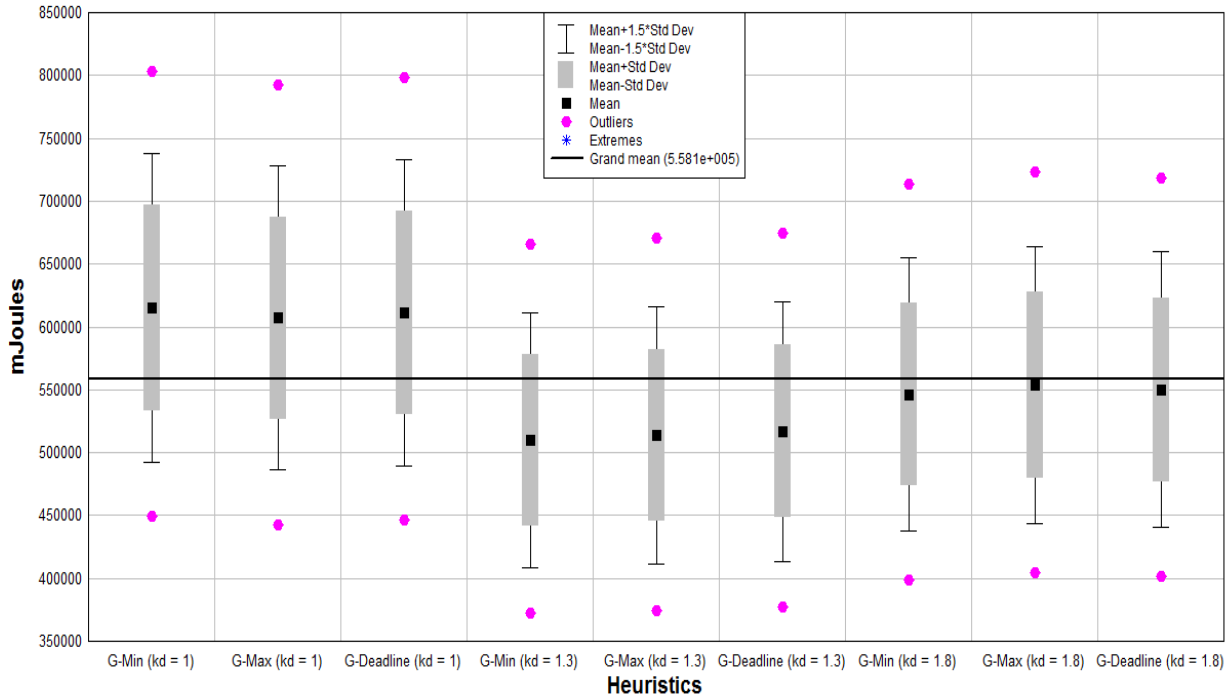

Fig. 3.9. Energy Consumption (without DTVS) for 100 Tasks with Various Values of kd

The energy consumed by each heuristic is calculated by taking product of makespan and the instantaneous power of the CN to which the task is scheduled. Therefore, the mean energy consumption with loose deadlines is increased.

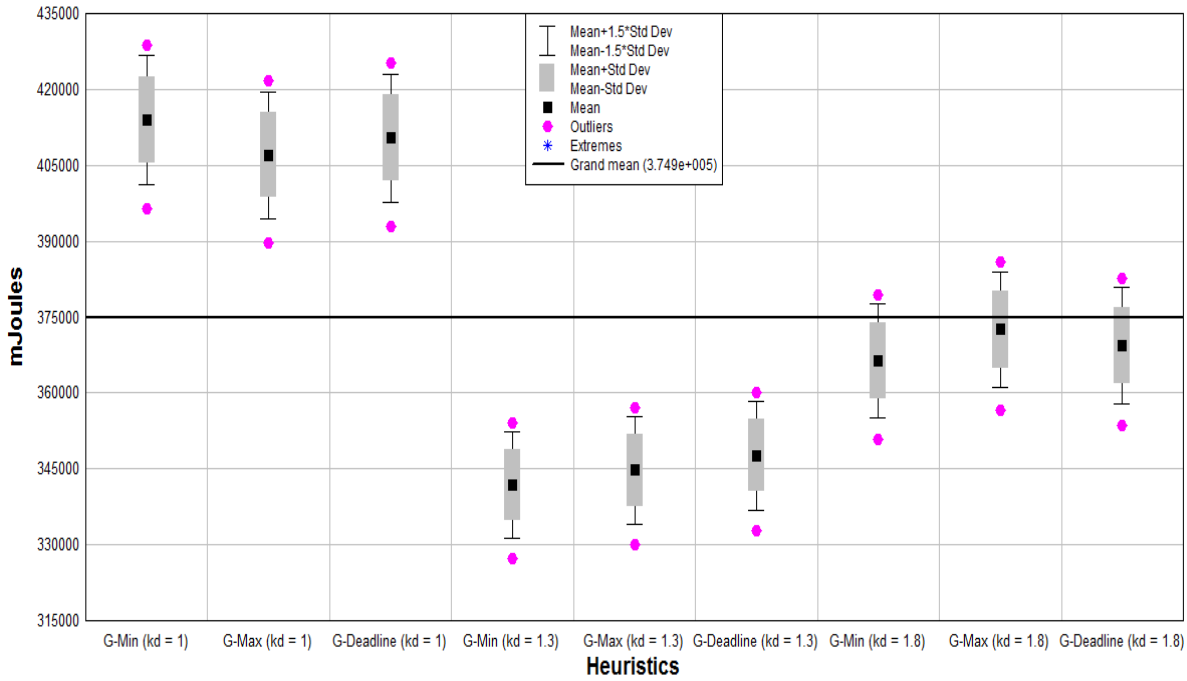

Fig. 3.10. Energy Consumption (with DTVS) for 100 Tasks with Various Values of kd

1000 tasks:

Fig. 3.11, 3.12, and 3.13 compare makespan, energy consumption without using DTVS, and energy consumption with using DTVS, for 1000 tasks, respectively. The GenAlgo outperformed all of the other heuristics in terms of makespan because it did not employ DVS and utilized the voltage at maximum available level. The MinMax schedules the tasks on least efficient CNs first. The MinMax and UtyFunc have large makespan while scheduling 1000 tasks. The ObjFunc and GenAlgo-DVS both have a mean makespan almost equal to the grand mean of all of the heuristics. The results for energy consumption for the G-Max, G-Deadline, and G-Min, are almost similar as they were for 100 tasks with G-Min having lowest value of mean energy consumption (1.6%) than the G-deadline and G-Max. The GenAlgo produced better results for

scheduling 1000 tasks as compared to scheduling 100 tasks due to its convergence for large sized workloads. When the DTVS is employed, MinMax revealed better performance as compared to the mean energy consumption without using the DTVS. The mean energy consumption with using the DTVS for all of the heuristics was improved by 17.65% when compared to mean energy consumption without using the DTVS. The performance of GenAlgo was degraded most when DTVS was employed amongst all the heuristics. This is due to the fact that the mean makespan for GenAlgo is lowest with low standard deviation resulting in less room for improvement.

The results for the G-Deadline, G-Min, and G-Max are compared without DTVS and with DTVS with high PE and task heterogeneity and loose deadline. The G-Deadline proved to the best amongst the three heuristics considered for the mean energy consumption without using the DTVS while scheduling 1000 tasks. This is due to the fact that the G-Deadline heuristic schedules the tasks with shortest deadlines first and making the deadlines larger has little effect on its performance.

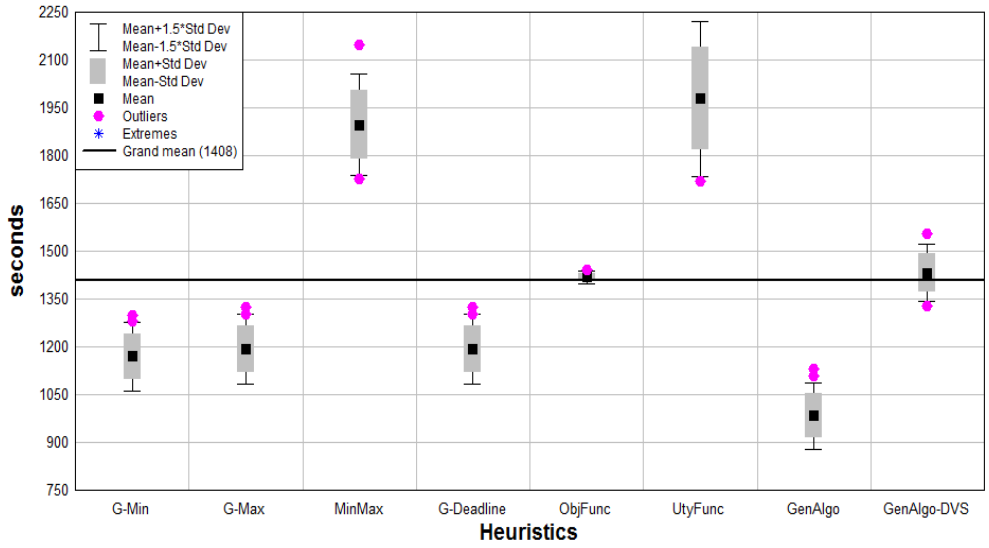

Fig. 3.11. Makespan for 1000 Tasks (Vtask = 0.1, VCN = 0.1, kd = 1)

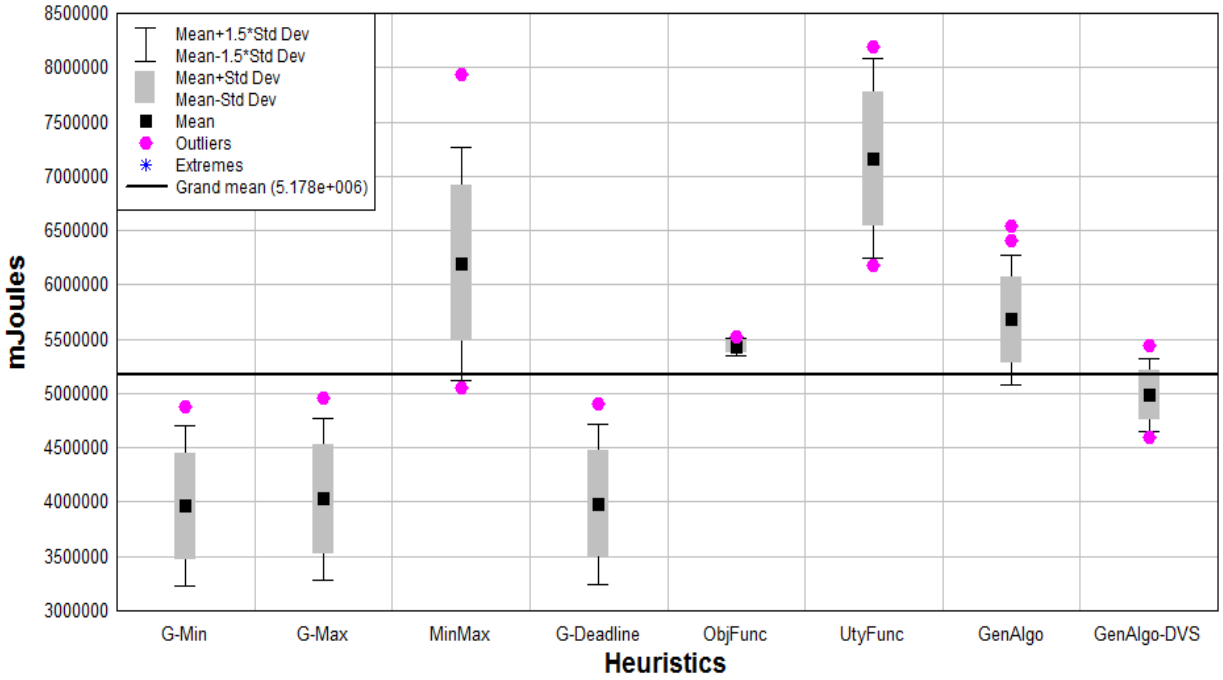

Fig. 3.12. Energy Consumption (without DTVS) for 1000 Tasks (Vtask = 0.1, VCN = 0.1, kd = 1)

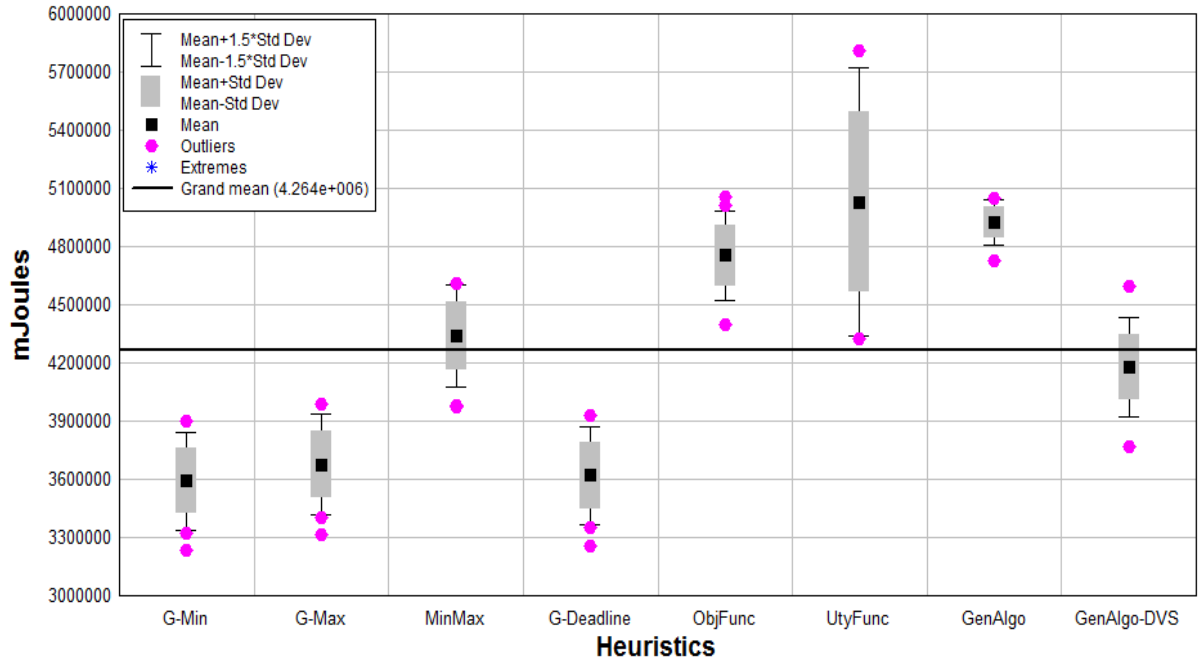

Fig. 3.13. Energy Consumption (with DTVS) for 1000 Tasks (Vtask = 0.1, VCN = 0.1, kd = 1)

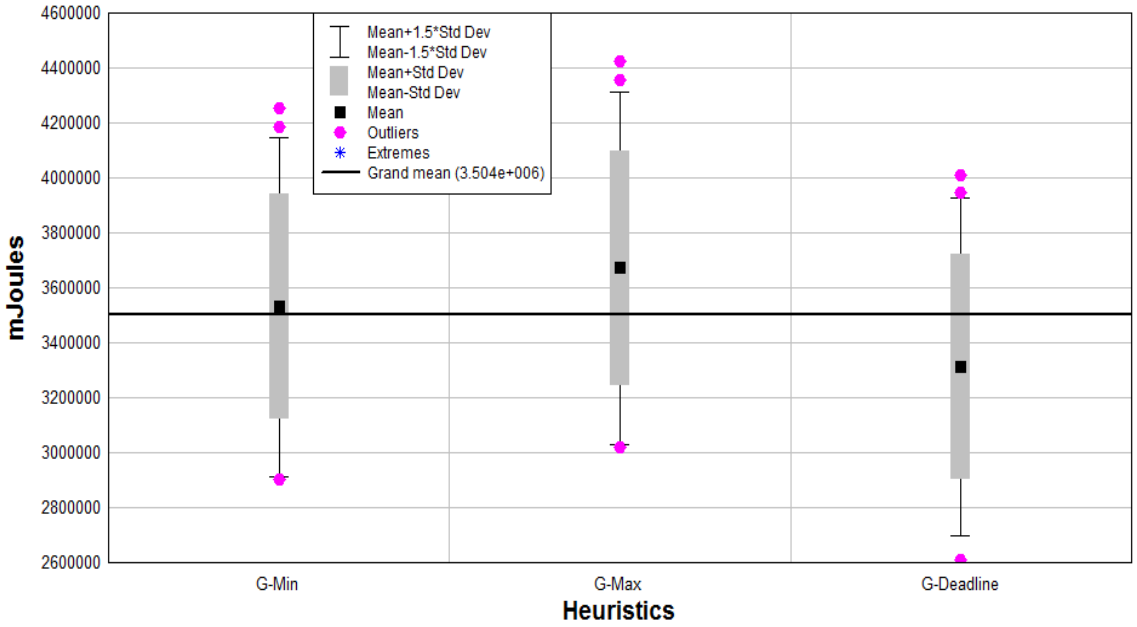

Fig. 3.14. Energy Consumption (without DTVS) for 1000 Tasks (Vtask = VCN = 0.35, kd = 1.8)

It can be observed from Fig. 3.14 that all of the three heuristics have lower minimum and larger maximum values for energy consumption that result in high standard deviation with high CN and task heterogeneity. There is an improvement of 46.60% in terms of energy consumption when DTVS is employed in these heuristics with same system parameters as depicted in Fig. 3.15. G-Max outperformed the other two heuristics and G-Deadline revealed worst performance. The heuristics performed well when DTVS is employed because of the diversity in CN speeds and task execution times as a result of increasing the task and CN heterogeneity.

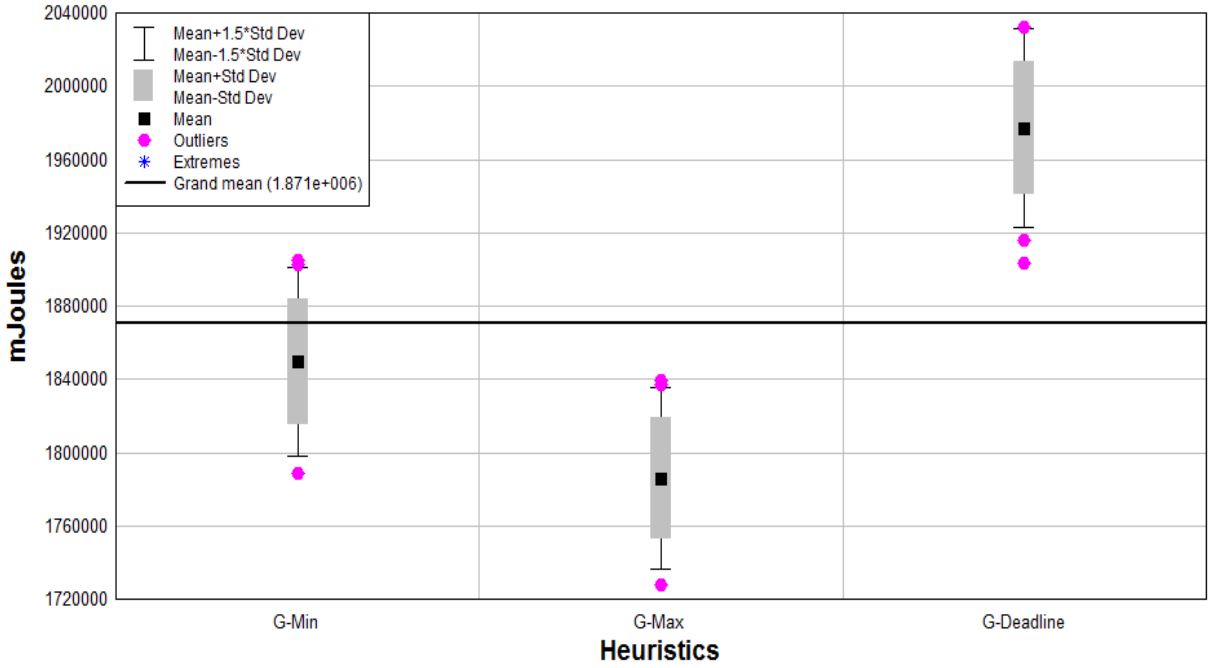

Fig. 3.15. Energy Consumption (with DTVS) for 1000 Tasks (Vtask = VCN = 0.35, kd = 1.8)

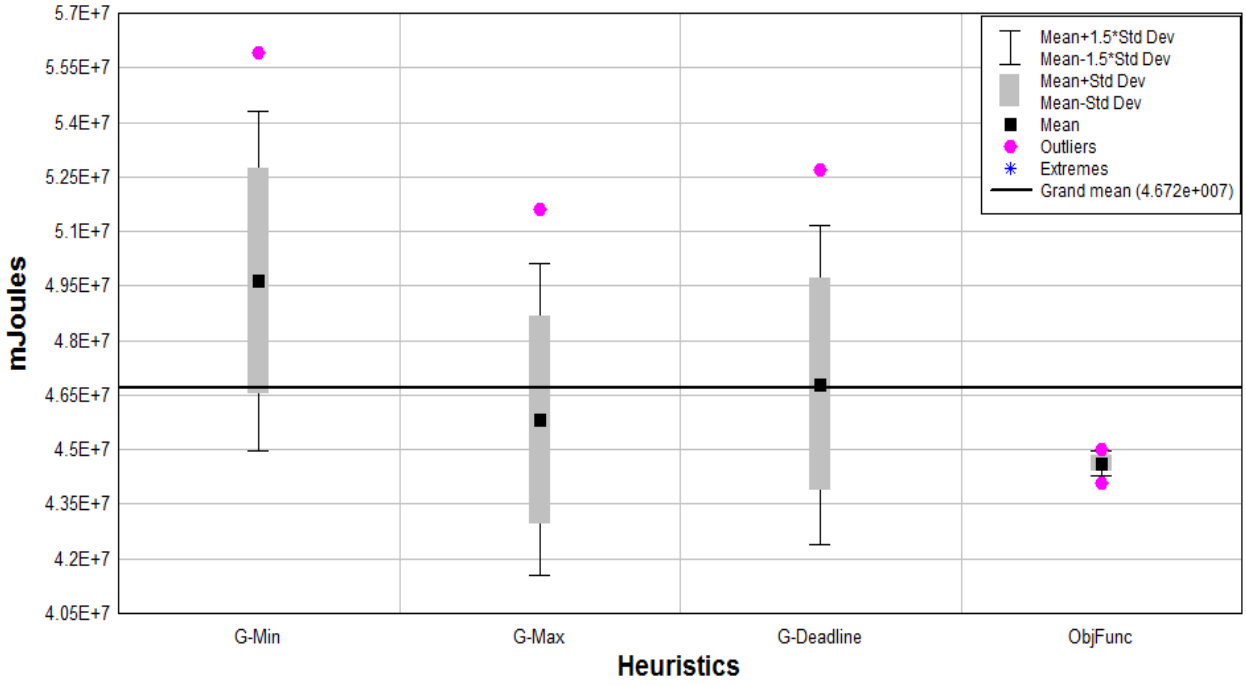

Fig. 3.16. Energy Consumption (without DTVS) for 10,000 Tasks (VCN = 0.1, kd = 1, Vtask = 0.35)

3.5.2.2. Medium Sized Workloads

10,000 tasks:

For medium sized workloads, we first compare the energy consumption of G-Deadline, G-Min, G-Max, and ObjFunc for high task heterogeneity with low CN heterogeneity and tight deadline. The comparisons of mean energy consumption of these heuristics without employing DTVS and using DTVS are plotted in Fig. 3.16 and Fig. 3.17, respectively.

Fig. 3.16 reveals that ObjFunc produces better results with lower standard deviation and mean energy consumption. The G-Deadline has mean energy consumption almost equal to the grand mean of the above listed heuristics. G-Min performed worst among these heuristics consuming 10.10% more energy than ObjFunc for scheduling 10,000 tasks.

The G-Min revealed significant improvement in energy consumption as compared to the other three heuristics, when DTVS was employed. There is a large slack in schedules of G-Min due to the fact that G-Min schedules the shortest tasks first.

The DTVS utilizes these slacks for improvement in the mean energy consumption. The G-Max schedules largest tasks first, leaving small tasks at the end to be scheduled. Therefore, small slack in schedules impose a limitation on performance of DTVS for G-Max. An improvement of 27.18% is observed in the mean energy consumption by using DTVS in these four heuristics.

When scheduling 10,000 tasks we compared the results of the G-Max, G-Min, MinMax, G-Deadline, ObjFunc, and UtyFunc for makespan and energy consumption with and without employing DTVS in Fig. 3.18, 3.19, and 3.20, respectively. The G-Max, G-Deadline, and G-Min have the lowest makespan and are similar to each other. The MinMax has slightly higher

makespan than the G-Max, G-Deadline, and G-Min. ObjFunc and UtyFunc produced worst results in terms of makespan (please see Fig. 3.18).

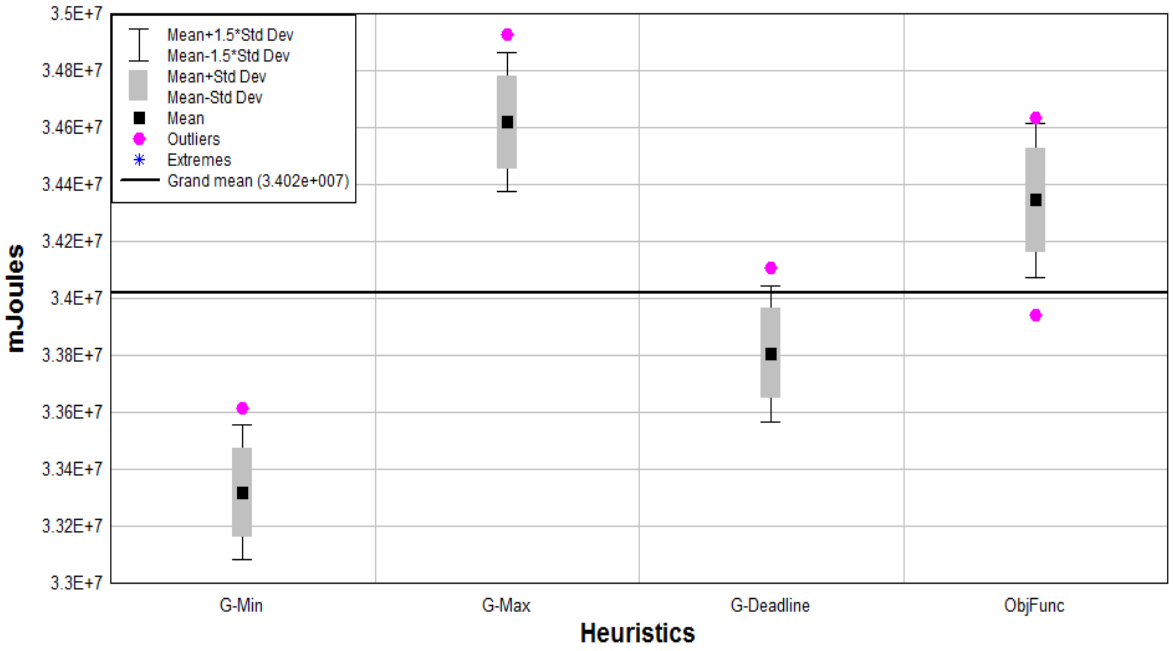

Fig. 3.17. Energy Consumption (with DTVS) for 10,000 Tasks (VCN = 0.1, kd = 1, Vtask = 0.35)

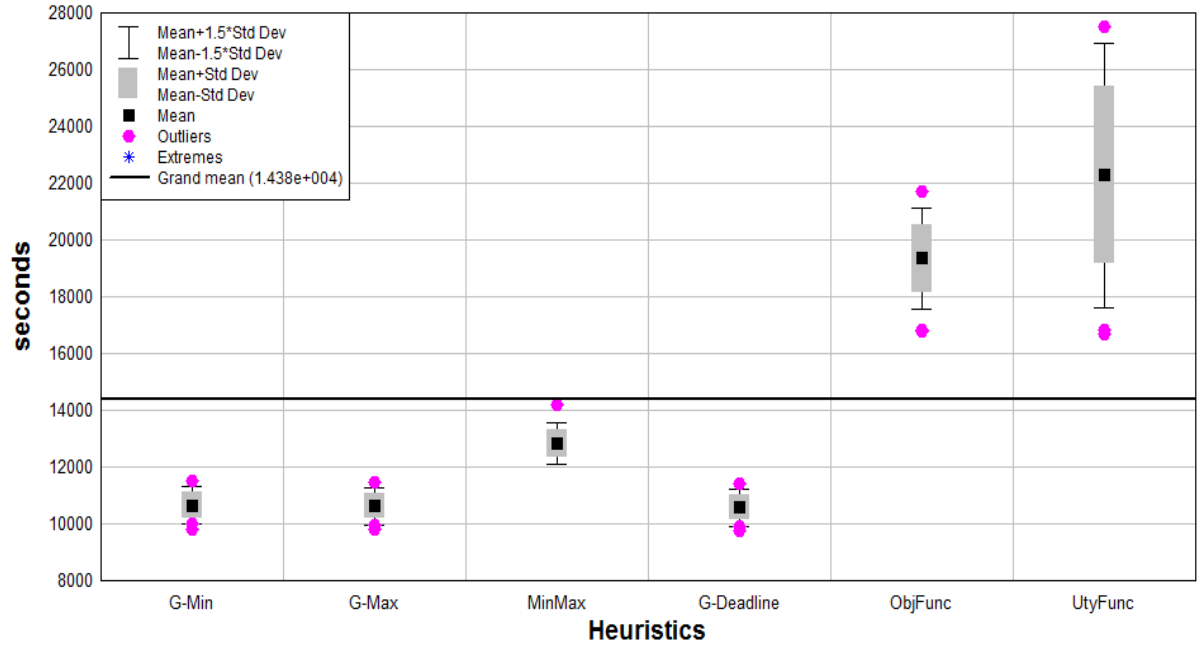

Fig. 3.18. Makespan for 10,000 Tasks (Vtask = VCN = 0.1, kd = 1)

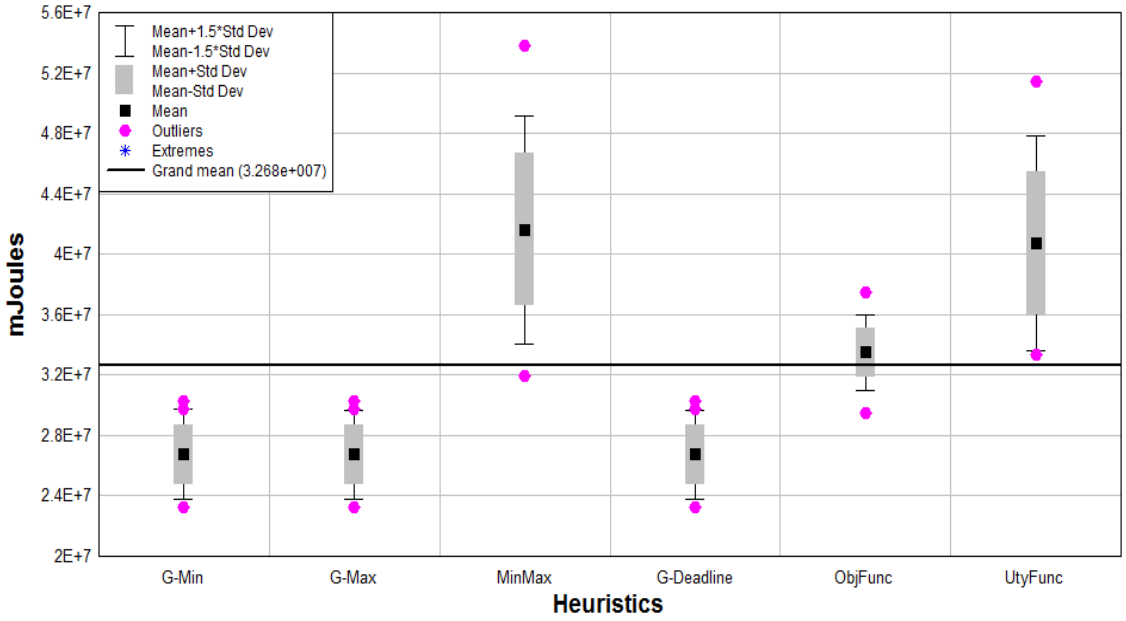

Fig. 3.19. Energy Consumption (without DTVS) for 10,000 Tasks (Vtask = VCN = 0.1, kd = 1)

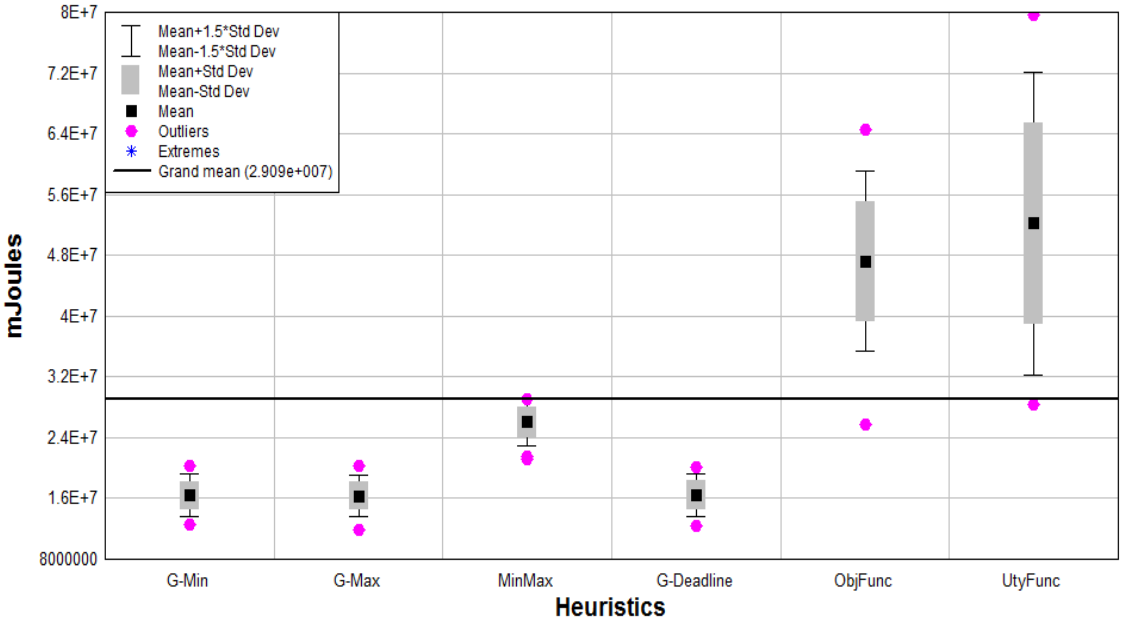

Fig. 3.20. Energy Consumption (with DTVS) for 10,000 Tasks (Vtask = VCN = 0.1, kd = 1)

The mean energy consumption for scheduling 10,000 tasks without deploying DTVS is depicted in Fig. 3.19. Fig. 3.20 shows the mean energy consumption with DTVS scheme. By comparing Fig. 3.19 and Fig. 3.20, it is revealed that a significant improvement in the mean

energy consumption of MinMax is achieved when DTVS is used. There is a decrease of 10.98% in the mean energy consumption of all of the heuristics with the deployment of DTVS. There is an improvement of 31.71% in mean energy consumption of the MinMax heuristic, which is significant when considering 10.98% improvement for all of the heuristics collectively.

3.5.2.3. Large Sized Workloads

100,000 tasks:

The ObjFunc produced best results in terms of mean energy consumption and mean makespan for large sized workloads, because ObjFunc considers multiple tasks and multiple CNs in its objective function when assigning the tasks to the CNs. The results for makespan and energy consumption (without DTVS) for large sized workloads are depicted in Fig. 3.21 and 3.22, respectively. UtyFunc produced results with large makespan and mean energy consumption because it considers the relative importance of speed and the execution time of the tasks, simultaneously. This results in a tradeoff between makespan and energy consumption, failing to produce better results. The G-Max, G-Deadline, and G-Min, all produced similar results with lower value of mean energy consumption and mean makespan than the grand mean.

UtyFunc revealed significant improvement in the mean energy consumption when DTVS was used. Overall 31.53% reduction in mean energy consumption of all of the heuristics is observed by using DTVS scheme. The results for mean energy consumption with DTVS are depicted in Fig. 3.23.

On the basis of the results obtained in this section we can safely draw a conclusion that irrespective of the workload size the heuristics G-Deadline, G-Min, and G-Max exhibits better results in terms of energy savings. If we categorize the workload size we observe that the

GenAlgo-DVS reported better results for small sized workload. Nevertheless, for large sized workload ObjFunc produces promising results.

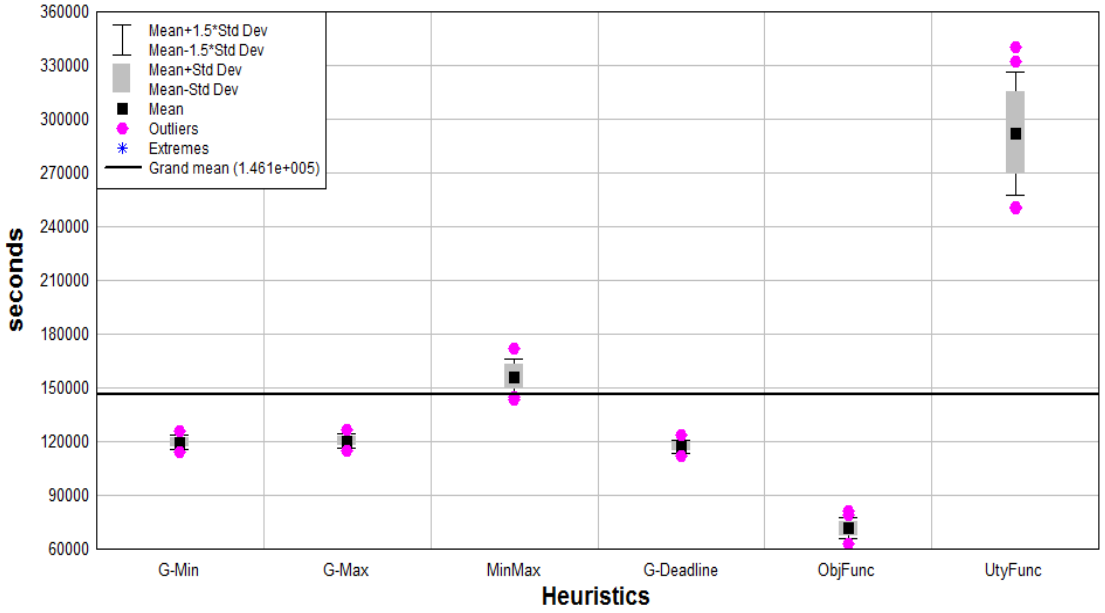

Fig. 3.21. Makespan for 100,000 Tasks (Vtask = VCN = 0.1, kd = 1)

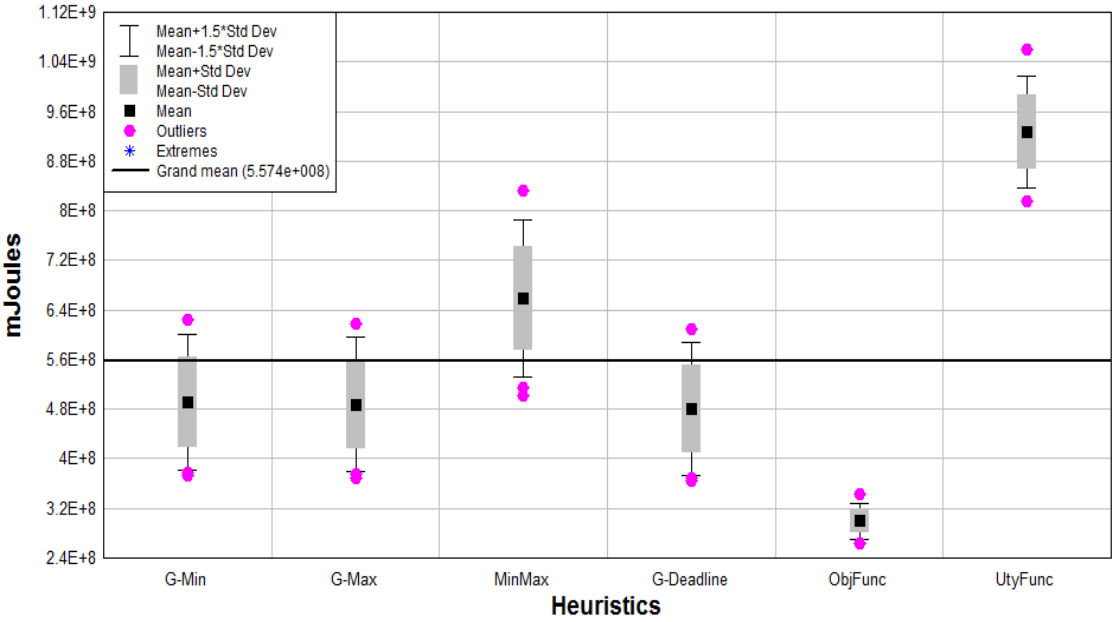

Fig. 3.22. Energy Consumption (without DTVS) for 100,000 Tasks (Vtask = VCN = 0.1, kd = 1)

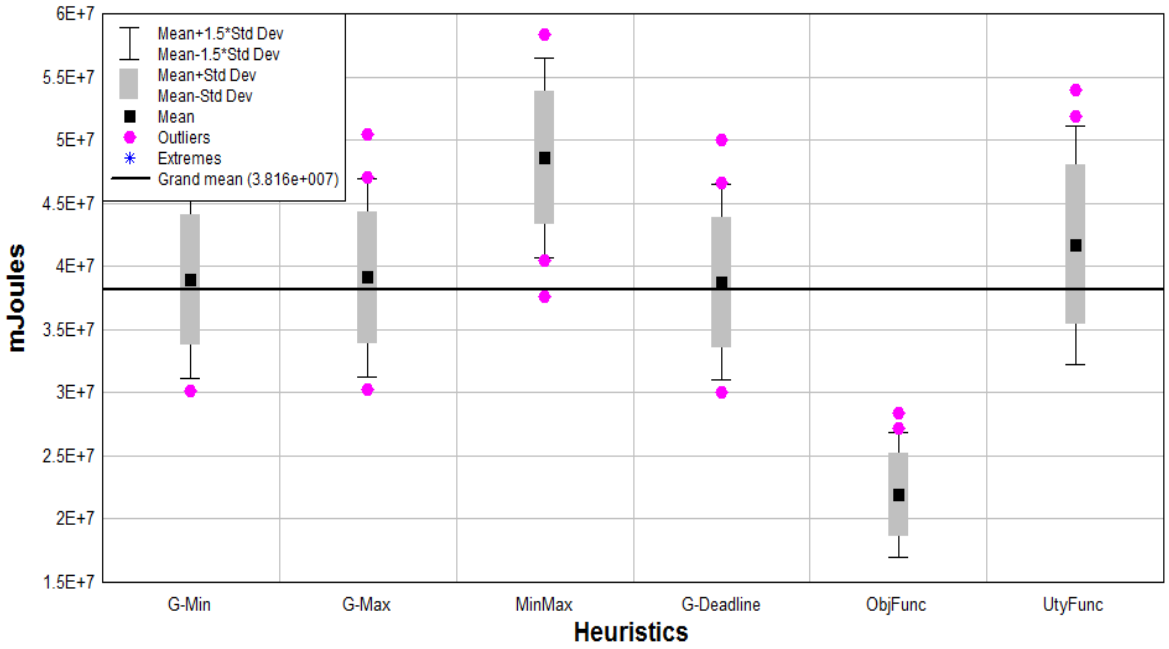

Fig. 3.23. Energy Consumption (with DTVS) for 100,000 Tasks (Vtask = VCN = 0.1, kd = 1)

3.6. Conclusions

With the growing demand of HPC for heavy computations, the power cost of the aforementioned is acting as a limiting factor and needs to be controlled and overcome using power-aware system solutions. This paper underlines the role of voltage level of nodes in cluster's power consumption and presents a methodology, termed as PAJS, that combines energy-efficient job scheduling with node awareness. We have analyzed and compared eight task scheduling techniques. The role of power optimization in modern HPC is emphasized and the proposed PAJS approach optimizes the tradeoff between energy efficient nodes (to reduce the amount of energy consumed) and performance aware patterns (to minimize the makespan). G-Max, G-Deadline and G-Min performed better for all sized workloads as compared to other heuristics. The minimum overall reduction in the energy consumption using DTVS was 10.98%. Nevertheless, the maximum energy savings using DTVS is 31.71% as compared to when DTVS

was not employed. For small-sized workloads GenAlgo-DVS also yielded better results in terms of mean makespan and mean energy consumption. However, for large-sized workloads ObjFunc performed well in addition to G-Max, G-Deadline, and G-Min.

The simulation results approve the superior performance of all the heuristics of the proposed methodology (PAJS) in terms of reduction in the makespan and energy consumption of the nodes comprising the HPC. The work presented here is not just restricted to clusters but can easily be adapted to grids, workstations, and datacenters. Both on the practical and theoretical point of view, we have introduced a coherent framework for the optimization of power in various resource scheduling strategies in HPC.

3.7. References

[3.1] A. Abbas, M. Ali, A. Fayyaz, A. Ghosh, A. Kalra, S. U. Khan, M. U. S. Khan, T. D. Menezes, S. Pattanayak, A. Sanyal, and S. Usman, "A Survey on Energy-Efficient Methodologies and Architectures of Network-on-Chip," Computers and Electrical Engineering. DOI: 10.1016/j.compeleceng.2014.07.012.

[3.2] I. Ahmad, S. Ranka, and S. U. Khan, "Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy," *22nd IEEE International Parallel and Distributed Processing Symposium*, pp. 1–6, Apr. 2008.

[3.3] C. D. Alfonso, M. Caballer, F. Avarruiz, and V. Hernandez, "An energy management system for cluster infrastructures," *Journal of Computer and Electrical Engineering*, vol. 39, no. 8, June 2013.

[3.4] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali, "Task execution time modeling for heterogeneous computing systems," *IEEE 9th Heterogeneous Computing Workshop*, pp. 185-199, May 2000.

[3.5] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali, “Representing task and machine heterogeneities for heterogeneous computing systems,” *Tamkang Journal of Science and Engineering.* vol. 3, no. 3, pp. 195-207, Nov. 2000.

[3.6] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasuderan, “FAWN: A fast array of wimpy nodes,” *22nd ACM symposium on Operating Systems Principles (SOSP 2009),* pp. 1-14, Oct. 2009.

[3.7] M. A. Aziz, S. U. Khan, T. Loukopoulos, P. Bouvry, H. Li, and J. Li, "An Overview of Achieving Energy Efficiency in On-chip Networks," International Journal of Communication Networks and Distributed Systems, vol. 5, no. 4, pp. 444-458, 2010.

[3.8] A. Beloglazov, J. Abawaj, and R. Buyya, “Energy aware Resource allocation heuristics for efficient management of data centers for cloud computing,” *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755-768, May 2012.

[3.9] H. Castro, M. Villamizar, G. Sotelo, C. O. Diaz, J. E. Pecero, and P. Bouvry, “Green flexible opportunistic computing with task consolidation and virtualization,” *Journal of Cluster Computing*, vol. 16, no. 3, pp. 545-557, Sep. 2013.

[3.10] G. A. Chaparro-Baqueero, Q. Zhou, C. Liu, J. Tang, and S. Liu, “Power-efficient schemes via workload characterization on the Intel’s single chip cloud computer,” *IEEE Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW)*, pp. 999-1006, May 2012.

[3.11] H. Al-Daud, I. Al-Azzonib, and D. G. Down, “Power aware linear programming based scheduling for heterogeneous computer clusters,” *In Future Generation Computer System* vol. 24, no. 5, pp. 745-754, May 2012. [Special Section: Energy Efficiency in Large Scale Distributed System]

[3.12] C. O. Diaz, M. Guzek, J. E. Pecero, P. Bouvry, and S. U. Khan, “Scalable and energy-efficient scheduling techniques for large-scale systems,” International Conference on Computer and Information Technology (CIT ‘11), Sept. 2011, pp. 641-647.

[3.13] S. Huang and W. Feng, “Energy efficient cluster computing via accurate workload characterization,” *In proceeding of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the grid; CCGRID,* IEEE S, May 2009.

[3.14] J. Andersson., “A survey of multiobjective optimization in engineering design,” *Technical Report, Department of Mechanical Engineering, Linköping University, Linköping, Sweden*, 2000.

[3.15] S. U. Khan and C. Ardil, “A game theoretical energy efficient resource allocation technique for Large distributed computing systems," *International Conference on Parallel and Distributed Processing, Techniques and Applications (PDPTA),* pp. 48-54, Jul. 2009.

[3.16] S. U. Khan and C. Ardil, "On the joint optimization of performance and power consumption in data centers," *International Conference on Distributed, High-performance and Grid Computing,* 2009, pp. 660–666.

[3.17] S. U. Khan and N. Min-Allah, "A Goal Programming Based Energy Efficient Resource Allocation in Data Centers," Journal of Supercomputing, vol. 61, no. 3, pp. 502-519, 2012.

[3.18] D. Kliazovich, S. T. Arzo, F. Granelli, P. Bouvry, and S. U. Khan, “Accounting for load variation in energy efficient data centers,” in *IEEE International Conference on Communications (ICC),* pp. 1154-1159, Jun. 2013.

[3.19] J. Kolodziej, S. U. Khan, L. Wang, A. Byrski, N. Min-Allah, and S. A. Madani, “Hierarichal genetic based grid scheduling with energy optimization,” *Journal of Cluster Computing,* vol. 16, no. 3, pp. 591-609, Sep. 2013.

[3.20] J. Kolodziej, S. U. Khan, L. Wang, M. Kisiel-Dorohinicki, S. A. Madani, E. Niewiadomska-Szynkiewicz, A. Y. Zomaya, and C. Z. Xu, “Security, energy and performance aware resource allocation mechanisms for computational grids,” *Future Generation Computer Systems*, Oct. 2012.

[3.21] A. Krioukov, C. Goebel, S. Alspaugh, Y. Chen, D.E. Culler, and R.H. Katz, “Integrating renewable energy using data analytics systems: Challenges and opportunities,” *Bulletin of the IEEE Computer Society Technical Committee.*, vol. 34, no. 1, pp. 3-11, 2011.

[3.22] W. Lang, S. Harizopoulos, J.M. Patel, M.A. Shah, and D. Tsirogiannis, “Towards energy-efficient database cluster design,” *In: Proceedings of the VLDB Endowment (PVLDB),* vol.5, no. 11, pp. 1684-1695, Aug. 2012.

[3.23] W. Lang and J. M. Patel, “Energy management for map reduce cluster,” in *Proceedings of the VLDB*, vol. 3, no. 1-2, pp. 129–139, 2010.

[3.24] P. Lindberg, J. Leingang, D. Lysaker, S. U. Khan, and J. Li, “Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems,” *Journal of Supercomputing*, vol. 59, no. 1, pp. 323-360, 2010.

[3.25] P. Lindberg, J. Leingang, D. Lysaker, K. Bilal, S. U. Khan, P. Bouvry, N. Ghani, N. Min-Allah, and J. Li, "Comparison and Analysis of Greedy Energy-Efficient Scheduling Algorithms for Computational Grids," in Energy Aware Distributed Computing Systems, A. Y. Zomaya and Y.-C. Lee, Eds., John Wiley & Sons, Hoboken, NJ, USA, 2012, ISBN 978-0-470-90875-4, Chapter 7.

[3.26] O. V. Maiuri and W. R. Moore., “Implications of voltage and dimension scaling on CMOS Testing: the multidimensional testing paradigm,” *16th IEEE Symposium on VLSI Test,* Apr. 1998.

[3.27] N. Mehta and B. Amrutur, “Dynamic supply and threshold voltage scaling for CMOS digital circuits using in-situ power monitor,” *IEEE Transactions on VLSI Systems,* vol. 20, no. 5, pp. 892-901, May 2012.

[3.28] N. Min-Allah, H. Hussain, S. U. Khan, and A. Y. Zomaya, "Power Efficient Rate Monotonic Scheduling for Multi-core Systems," Journal of Parallel and Distributed Computing, vol. 72, no. 1, pp. 48-57, 2012.

[3.29] J. Shuja, S. A. Madani, K. Bilal, K. Hayat, S. U. Khan, and S. Sarwar, “Energy efficient data centers,” *Computing*, vol. 94, no. 12, pp. 973-994, Sep. 2012.

[3.30] S. Sha, J. Zhou, C. Liu, and G. Quan, “Power and energy analysis on Intel single-chip cloud computer system,” *In South Easton, Proceedings of IEEE*, pp. 1- 6, Mar. 2012.

[3.31] S. Usman, S. U. Khan, and S. Khan, “A comparative study of voltage/frequency scaling in NOC,” *2013 IEEE International Conference on Electro/Information Technology,* May 2013.

[3.32] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry, “An overview of energy efficiency techniques in cluster computing systems,” *Cluster Computing*, vol. 16, no. 1, pp. 3-15, Sept. 2011.

[3.33] L. Wang, S. U. Khan, D. Chen, J. Kołodziej, R. Ranjan, C. Z. Xu, and A. Zomaya., “Energy aware parallel task scheduling in a cluster,” *Future Generation Computer Systems,* vol. 29, no.7, pp. 1661-1670, Mar. 2013.

[3.34] J. S. Yuan, “A novel energy efficient algorithm for cloud resource management,” *International Journal of Knowledge and Language Processing,* vol. 4, no. 2, pp. 12-22, Apr. 2013.

[3.35] Z. Zong, X. Qin, X. Ruan, K. Bellam, M. Nijim, and M. Alghamdi, “Energy-efficient scheduling for parallel applications running on heterogeneous clusters,” *International Conference on Parallel Processing (ICPP 2007),* pp. 19-26, Sep. 2007.

4. ENERGY EFFICIENT RESOURCE SCHEDULING THROUGH VM CONSOLIDATION IN CLOUD COMPUTING

This paper[1] is to be submitted in a reputed cloud computing journal. The authors of the paper are Ahmad Fayyaz, Muhammad Usman Shahid khan, and Samee U. Khan.

4.1. Introduction

The dual influence of increasing cloud computing data center energy consumption and increasing energy costs has raised the significance of cloud computing data center efficiency as a policy to decrease costs, accomplish size and indorse environmental responsibility. The data centers are the most integral part for most of Information Technology (IT) organizations. Many renowned organizations, such as Google, Microsoft, IMB, and Amazon have big data centers that contain thousands of computing servers around the world to provide fast and efficient cloud computing services to the customers [4.1]. The past decade has witnessed a phenomenal increase in the number of data centers, and the size of the existing data centers. The aforementioned situation have increased the word-wide power consumption that drive many research communities to carry out research on the data center energy consumption, energy efficient techniques for computing units, and power consumption prediction of the data centers [4.2-4.6]. In a study conducted by the Environmental Protection Agency (EPA) in 2006 [4.2] stated that the data centers are consuming more than 61 Tera Watt hour (TWh) of electricity per year that was

[1] The material in this chapter was co-authored by Ahmad Fayyaz, Muhammad Usman Shahid Khan, and Samee U. Khan. Ahmad Fayyaz had primary responsibility for designing the system model, conducting experiments and collecting results. Ahmad Fayyaz also drafted and revised all versions of this chapter.

1.5% of the total power consumption of the whole US for the same year. The report also stated that the data center power consumption will have an annual growth of 16% over the next 10 years. Figure 4.1 shows the Emerson Network Power modeled energy consumption for a typical data center and evaluated how energy is used within the data center. The power usage is classified as either "demand-side" or "supply-side." Demand-side systems are the servers, storage, communications and other IT systems that support the data center business. The supply-side systems support the demand side.

The propagation of Cloud computing has stemmed in establishing large-scale data centers. The American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) [4.7], has published a trend that by 2014, the energy and infrastructure costs of the data center will contribute about 75% in the total data center cost, while IT will contribute the remaining 25% in the overall operating cost of the data center [4.8].

The computing resources quantity and hardware power inefficiency are not the only factors that result in tremendously energy consumption in the data centers. The inefficient use of data center resources, such as CPUs and memory play a big part in the increase of energy consumption.

In [4.9], the authors collected a data from more than 5000 computing servers in a data center over a period of six-months and reported that the data center servers are usually not idle but the server utilization is rarely 100%. More than 90% of the servers were running at 10-50% utilization of their total 100% capacity. This phenomenon results in extra expenses on over provisioning that directly increase the total power consumption cost of the data center [4.9]. Moreover, handling and preserving over-provisioned data center's resources result in increased Cost of Ownership (TCO). In another study [4.10], authors reported that if the data center servers

are completely idle even then the power consumption is 70% of their total peak power consumption. Therefore, it is a known conclusion that the underutilization of the data centers servers is extremely inefficient with respect to the energy consumption.

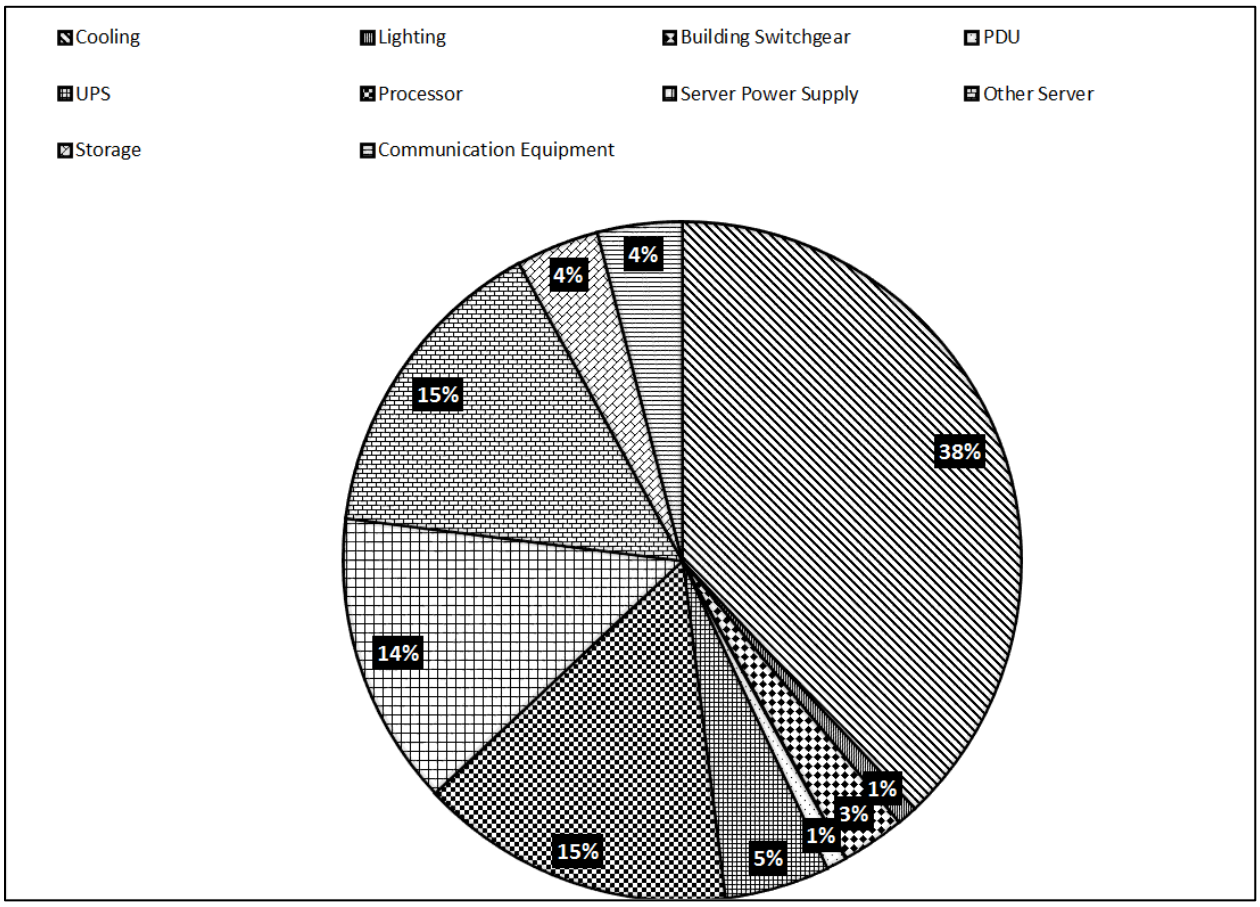

Fig. 4.1. Data Center Energy Consumption Partition

In [4.11], a comprehensive study is conducted that monitor energy consumption of Grid'5000 infrastructure. The authors reported significant opportunities for energy saving in the data center via techniques, such as switching servers on and off with respect to utilization or run the servers on low power mode. The data center's energy consumption can be reduce by switching idle servers to low-power modes, such as hibernation or sleeping, this process will reduce idle power consumption. There are some other critical issues that arise from high energy consumption of computing resources, such as the power required by the cooling system

operation of the data center. A report states that for each power Watt consumed by a computing entity or a server, data center has to consume another 0.5-1 Watt for the cooling system to keep the computing entity cool [4.12]. Moreover, higher the energy consumption by the data center's infrastructure, higher is the carbon dioxide (CO2) emissions that contribute to the greenhouse effect [4.13]. One simple solution for the energy inefficiency in the data centers is to involve virtualization technology [4.14]. The virtualization technology provides opportunities to the Cloud providers to create several Virtual Machines (VMs) on a single physical computing server. This virtualization phenomenon improves the resource utilization and also increases the Return On Investment (ROI). Moreover, the use of live migration [4.15], we can dynamically consolidate the VMs to the minimal number of physical servers according to their current resource requirements.

The rest of the paper is arranged as follows. Related work is elaborated in Section 4.2. The system model, problem formulation, and implementation details of the VM Consolidation model are presented in Section 4.3. Section 4.4 presents the simulation results and their comparison, while in Section 4.5 concluding remarks are presented.

4.2. Related Work

A number of literature works have been published that proposed solutions to reduce carbon dioxide emission amount of the Cloud data centers. One group of researchers focused on the reduction of energy consumption in a single data center or by only considering the hardware aspects of the data centers [4.19], [4.20]. The data centers are benefited from some renowned technologies, such as virtualization [4.21]. Among the virtualization techniques, there are VMs migration [4.23] and consolidation [4.24]. However, the main issue in the VM migration or consolidation is its complexity. Moreover, the VMs resumption and suspension causes system

overloading [4.15]. Furthermore, these methodologies are more of a reactive methods rather than proactive and preventive. Therefore, preventive methods are more important and effective. As stated in the introduction that an idle server consumes almost half of the power compared to the power it consumes at peak load [4.18]. The authors of [4.25] introduce a dynamic right-sizing on-line algorithm that predicts how many servers will be required to execute the arriving workload of the data center. The experimental results of [4.24] stated that dynamic right-sizing achieves significant energy savings, but the technique requires different power levels of the servers and servers should be able to transit between different states. In a similar work [4.23], Green Open Cloud (GOC) architecture is proposed that has advance resource reservation for the users to increase the prediction of the arrived requests. The aforementioned technologies are implemented within a data center and aim to decrease the energy consumption, while the technologies do not specifically consider carbon emission. The reduction in the data center energy consumption will not unavoidably reduce the carbon footprint. The works presented in [4.17] and [4.22] reflect the availability of both non-polluting and polluting energy sources in a single data center. The techniques use prediction-based scheduling algorithms to increase usage of green energy sources.

The Green Scheduler considers the servers to be in an order [4.30, 4.31]. It then starts scheduling the tasks to first server from the pool until that server can execute no more tasks and is overloaded. The scheduler then schedules the tasks to the next server and so on. Servers that come last from the pool of servers are idle most of the time because of the fact that the tasks are scheduled to servers that come earlier in the pool. Consolidation of tasks is achieved at the time of allocation of tasks. Our work is different from Green scheduler as it has variable sized workload as compared to the fixed sized workload of Green scheduler. In addition our technique

consolidates the tasks even after the allocation phase is over i.e. we migrate the tasks from one server to the other to minimize the energy consumption even if the task is in execution phase.

The DENS [4.32] methodology selects the best-fit computing resources for the execution of tasks by considering the communication potential and load level of data center components. Its aim is to achieve balance between traffic demands, job performances, energy consumed by the data center, and the job QoS requirements.

Round Robin [4.33] scheduler equally distributes the communicational and computing loads among the switches and servers. As a result no server is overloaded and the network traffic is balanced. This scheduler is least efficient in terms of energy consumption because all the switches and servers are busy most of the time. In our work the workload is exponentially distributed to mimic the real time arrival of workload. We are also incorporating consolidation of tasks whereas Round Robin is not using any type of consolidation technique.

4.3. VM/Task Consolidation Methodology

Underutilization of servers in a data center is a major cause of higher power consumption. Higher number of running servers results in higher power consumption. Therefore, optimal utilization of servers will result in lesser number of turned on servers and high power efficiency. There is a growing interest in reducing energy consumption of data centers. Cloud data centers use virtualization technology to host multiple virtual machines (VMs) on a single physical server.

Virtual Machine (VM) placement and scheduling are important characteristics of data center that consolidate the servers resulting in cutback of the amount of hardware usage. The general approach for handling the VM placement problem is to have a mathematical representation or a metric of resource utilization. Mapping a VM correctly to a PM is based on

the capacity of the PM and the resource requirements of the VM. VM placement is an important research domain in data centers where provisioning is performed manually. By applying efficient VM placement algorithms, Cloud providers are able to enhance energy efficiency.

Most of the methodologies proposed in the literature only considered CPU utilization (defined in MIPS) as a decision metric for the above stated approaches. Approaching the problem with a single varying parameter and keeping other parameters static solves the problem in the controlled environment. However, the assumption of non-varying parameters reduces the effectiveness of the proposed methodologies in real environment. Therefore, there is a need of research endeavors that are not based on a single parameter. In this regard we aim to carve a strategy that considers the following constraints, all at once.

- CPU constraints
- Memory constraints
- Bandwidth constraints

4.3.1. System Model

Power consumption $P(u)$ is defined as a function of CPU utilization [4.26].

$$P(u) = f.P_{max} + (1 - f).P_{max}.u, \tag{4.1}$$

where P_{max} is the power consumed by computing servers, the fraction of power used by idle server is given by f and CPU utilization is denoted by u.

Total energy consumption by a server is given by (Eq. 4.2). The CPU utilization may vary with respect to time due to variability in workloads

$$E = \int_t^{\cdot} P\big(u(t)\big)\, dt. \tag{4.2}$$

4.3.2. Problem Formulation

Consider a cloud consisting of *S* servers, each having its own memory. Let the i-th server be denoted by *Si* and let D_i denote the total memory capacity of Si. Suppose that the i-th server has a set of VMs, V = {v1, v2, v3, …., vM}, where M represents the total number of VMs at a particular server at a given time. VM_j^i denotes j-th VM on the i-th server. Suppose d_j^i is the memory consumed by the VM_j^i and AM_i is the available memory at *Si*, such that

$$AM_i = D_i - \sum_{j=1}^{M} d_j^i \tag{4.3}$$

$$\sum_{j=1}^{M} d_j^i = d_i \tag{4.4}$$

where d_i is the total memory being used by all the VMs at S_i at any time.

Each VM_j^i has some CPU requirement, i.e., the CPU utilization of VM_j^i is given by CPU_j^i, while the overall consumed CPU utilization of the server S_i is represented by cpu_i.

$$\sum_{j=1}^{M} CPU_j^i = cpu_i \tag{4.5}$$

$$CPU_{avail} = CPU_i - cpu_i \tag{4.6}$$

where CPU_i is total CPU power and CPU_{avail} is the available CPU power of the server S_i.

The Data Center Network (DCN) is the communicational backbone of the cloud computing [4.27]. We consider Fat-tree DCN architecture in our study because of its better performance in terms of throughput and average network delay [4.28]. The Fat-tree DCN architecture is switch-centric network topology consisting of *k* pods. There are *k* servers and *k* switches within each pod. The switches are ordered in two successive layers of *k/2* switches. The lower layer (edge layer) switches are linked to *k/2* servers and *k/2* upper layer (aggregation layer)

switches in each pod. Each aggregation layer switch in the pod is linked to *k/2* core level switches, out of the total of $(k/2)^2$ core level switches. The Fat-tree DCN architecture is shown in Figure 4.2. Let the power consumed at core level switches be denoted by P_{core}, power consumed by the aggregation level switches by P_{agg}, and at the edge level switches by P_{edge}, respectively. In addition the total power consumed by all the servers is given by P_{ser} where P_j^i is the power consumed by VM_j^i.

$$P_{ser} = \sum_{i=1}^{S} . \sum_{j=1}^{M} P_j^i \tag{4.7}$$

The bandwidth required by VM_j^i on server S_i is represented by BW_j^i and the bandwidth of the slowest link, *l*, between two servers is denoted by BW_l. Suppose that the utilized bandwidth of the link *l* at any given time is denoted by $BW_l^{utilized}$. The free bandwidth on the slowest link, BW_{free} is then given as

$$BW_{free} = BW_l - BW_l^{utilized} \tag{4.8}$$

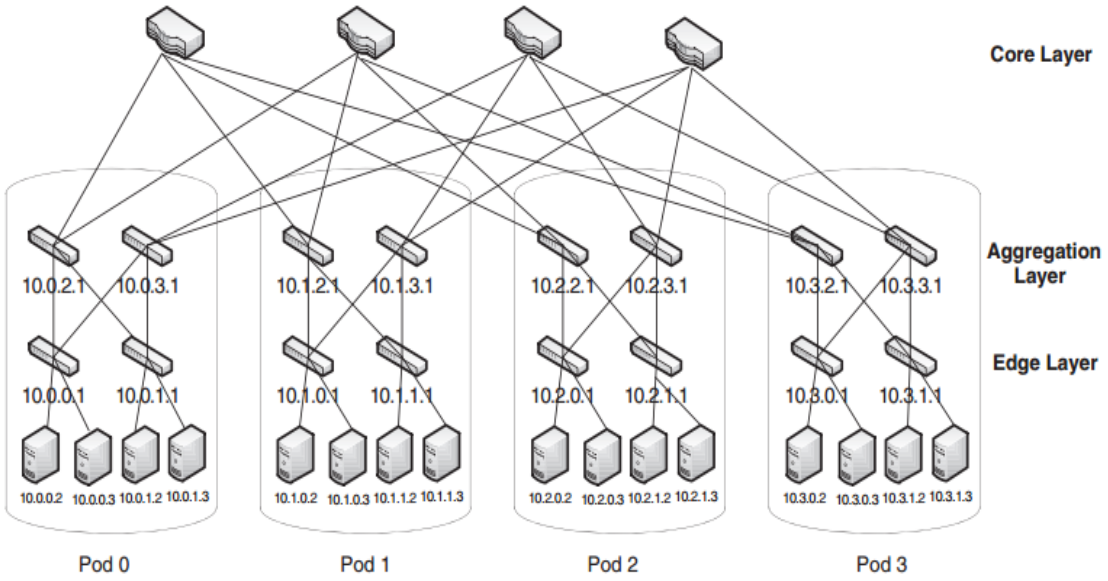

Fig. 4.2. Fat-Tree DCN Architecture [4.28]

Table 4.1. Notation/ Acronyms and their Meanings

Symbols	**Meaning**	**Symbols**	**Meaning**
DPM	Dynamic Power Management	DTVS	Dynamic Threshold Voltage Scaling
DVS	Dynamic Voltage Scaling	DVFS	Dynamic Voltage and frequency Scaling
S_i	*i*-th server	AM_i	Available memory at S_i
D_i	Total memory capacity of S_i	d_i	Total memory used by all the VMs at S_i
VM_j^i	*j*-th VM on the *i*-th server	CPU_j^i	CPU utilization of VM_j^i
d_j^i	Memory consumed by the VM_j^i	cpu_i	Overall CPU utilization of the server S_i
P_j^i	Power consumed by VM_j^i	CPU_i	Total CPU power of the server S_i
BW_j^i	Bandwidth required by VM_j^i on server S_i	CPU_{avail}	Available CPU power of the server S_i
P_{agg}	Consumed by the aggregation level switches	P_{ser}	Total power consumed by all the servers
BW_{free}	Free bandwidth on the slowest link	P_{edge}	Power consumed by edge level switches
$BW_l^{utilized}$	Utilized bandwidth of the link *l*	P_{core}	Power consumed by core level switches
BW_l	Bandwidth of the slowest link	DCN	Data Center Network

Our goal is to solve the multi-objective optimization problem with multiple constraints. The problem can be formulated as follows:

- The overall power consumption on both the servers and the switches in the DCN is minimized.

Mathematically

$$\min(P_{ser}) \quad + \quad min \sum (P_{core}, P_{agg}, P_{edge})$$

and

$$min \sum_{i=1}^{n} mt_j^i$$

Subject to the following constraints

$$BW_j^i \leq BW_{free} \tag{4.9}$$

$$CPU_j^i \leq CPU_{avail} \tag{4.10}$$

$$\sum_{j=1}^{M} d_j^i \leq D_i \tag{4.11}$$

$$size(VM_j^i) \leq AM_i \tag{4.12}$$

$$\sum_{j=1}^{M} CPU_j^i \leq CPU_i \tag{4.13}$$

Constraint (4.9) makes sure that the free BW is more than the BW required by the VM for allocation / migration. Similarly, constraint (4.10) is there to ensure that there is enough CPU power available for the incoming VM. The memory consumed by all the running VMs must be less than the total memory of the server and is depicted as constraint (4.11). In constraint (4.12), the size of the VM must be less than the available memory of server where the VM is being allocated and/or migrated. Constraint (4.13) guarantees that the CPU power consumed by all the running VMs must be less than the total CPU power of the server.

4.3.3. VM Consolidation Methodology

The task/VM scheduling is a two-step process. First the tasks arriving at the VM monitor (one server acts as VM monitor) are allocated to the servers in the cloud computing data centers. The tasks, as they arrive, are allocated to the servers starting from one side of the fat-tree network. The first server in the network is allocated tasks until anymore allocation of tasks

Algorithm 4.1: VM Consolidation

Definitions: *S= set of server machines, tag = status of server, Maxvm = maximum number of Virtual machines that can be hosted a machine, £ = list of over utilized servers, ¥= list of filled servers, μ = list of under-utilized servers, £ = list of not filled servers, Þ = list of OFF servers*

```
1.  for each s ∈ S do
2.      e ← getEnergyCosumption(s)
3.      vmc ← getHostedVMCount(s)
4.     if e > 0.9 then
5.        £ ← £ .append(s)
6.    else if e > 0.5 and vmc = Maxvm
7.        ¥← ¥ .append(s)
8.    else if e < 0.5 and vmc = Maxvm
9.        μ ← μ .append(s)
10.   else if  e > 0.1 and vmc < Maxvm
11.       Ω ← Ω .append(s)
12.   else
13.       Þ ← Þ .append(s)
14.    end if
15. end for
16. for each s ∈ £ do
17.     vm  ←getVM(s)
18.     oh ← getServerfromNotFilledOrOffLists(Ω, Þ)
19.     if oh != NULL then
20.        if  migrateVM(vm,oh) then
21.           £ ←£.remove(s)
22.           reemoveServerFromNotFilledOrOffLists(Ω, Þ,oh)
23.       else
24.          goto Line 18
25.      end if
26. end for
27. for each s ∈ Ω do
28.     while vm  ←getVM(s)
29.          oh ← getServerfromNotFilledLists (Ω,s)
30.          if oh != NULL then
31.              if !migrateVM(vm,oh) then
32.                  goto Line 29
33.              end if
34.          else
35.              Break
36.          end if
37.     end while
38.    if vm == NULL then
39.        Ω ← Ω.remove(s)
40.    end if
41. end for
```

overload the server. Further, the tasks are allocated to the next server and son on until all the tasks are exhausted. Virtual machines are created for the execution of each task allocated to the servers based on the CPU power and memory requirement of each task. The task continues to be executed on this VM until it is completed or migrated.

Second, when the pool of tasks to be allocated is exhausted, VM/task migration is initiated. In this step the VM monitor periodically checks al the servers for the under-utilized servers. The process for migration is depicted in Algorithm 4.1. A server that is loaded with tasks such that its 90% of its total capacity is being utilized in the execution of tasks is termed as overloaded. We consider 90% utilization of server by tasks as overloaded because we leave 10% of CPU power for the server's own operations. On the other hand we consider a server being under-utilized if 30% of the CPU capacity is used by the tasks execution. The servers those are assigned no tasks are turned off using DPM and DTVS to reduce the energy consumption in the idle mode.

When the VM monitor is done calculating the utilization if servers, it then migrates tasks from under-utilized servers to those running servers that can complete the tasks within the deadlines of the tasks. Here a delay is incurred due to the migration process and is dependent on the slowest link between the servers. Therefore, before migrating the task the estimated time to migrate and the time of execution at the target server is calculated so as to check whether the task's deadline can be met at the targeted server. If this condition is satisfied then the task is migrated and the server from which the task is migrated is turned off by using DTVS. The set of conditions that need to be satisfied for a migration to take place are listed and explained in the problem formulation. If the migrations are increased then the links and switches will experience congestion as all the traffic will be routed through the switches. Therefore the power consumed

by the communication links and the switches is increased. On the contrary, the power of servers is reduced by migrating tasks from under-utilized servers and turning them off. Migration of tasks even when they are in execution is termed as task consolidation.

4.4. Performance Evaluation

4.4.1. GreenCloud Simulator

We used the GreenCloud simulator to implement our proposed methodology for the purpose of performance evaluation. GreenCloud simulator was developed as an extension of the Network Simulator NS2 [4.29]. It captures the communication processes of the data center at the packet level. GreenCloud simulator provides users with a tool that monitors the energy consumed by servers, switches, and communication links within a cloud computing data center. Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) are the two energy efficient optimization techniques that are used by the simulator but in our proposed work we incorporate Dynamic voltage Scaling (DVS) and Dynamic Threshold Voltage Scaling (DTVS) in the GreenCloud simulator to further achieve reduction in energy consumption of the cloud data center.

The deadline based model is deployed for the execution of the workload in the simulator. The size of the tasks within the workload is kept variable so as to mimic the real time workloads.

4.4.2. Simulation Scenario

In our simulation setup we used the Fat-tree DCN architecture because of its better performance in terms of throughput and average network delay. The Fat-tree architecture is switch-centric network topology consisting of n pods as shown in Fig. 4.2 and explained in Section 4.3.2. To interconnect the servers inside the racks we used links with a bandwidth of

1Gigabit Ethernet (GE) while for the interconnection of the core, aggregation, and edge layer switches we used 10 GE links. To make the workload more realistic the GreenCloud workload is exponentially distributed.

4.4.3. Simulation Results

The simulation results are evaluated in terms of the power consumption of the cloud data center. Three scheduling techniques namely Green Scheduler, Round Robin, and Random scheduling are implemented. The results are then compared with the same scheduling techniques when VM/task consolidation is incorporated in them. It is important to note here that in the Green scheduler the consolidation of VMs/tasks is achieved at the time of allocation of tasks. It then starts scheduling the tasks to the first server from the pool until it becomes overloaded. In addition, the Green scheduler technique has fixed size workload but we considered a variable sized workload when we used task consolidation. The second technique, Round Robin scheduler, equally distributes the communicational and computing loads among the switches and servers. As a result no server is overloaded and the network traffic is balanced. Therefore, there is more room for performing VM consolidation in this technique as compared to the Green scheduler. Tasks are also scheduled randomly on different servers just for the purpose of comparison with our proposed work. The Random scheduler is included as a reference so that we can then calculate how much improvement in power consumption is achieved after VM migration is performed.

The results for the power consumption of the switches and servers are plotted individually against the changing load of the data center from 30% to 90% in figure 4.3 and figure 4.4. Figure 4.3 depicts power consumption of all the servers in the data center with the changing load of the cloud data center. It can be observed that the power consumption of the

servers has decreased significantly after performing VM migration. As the load in the data center increases, the number of task migrations is decreased because all of the servers then work near to their full capacity, leaving less or no space for VM migrations and as a result power consumption is slightly increased. In addition, it can also be noted that the scheduling techniques have higher power consumption when VM migration is not performed.

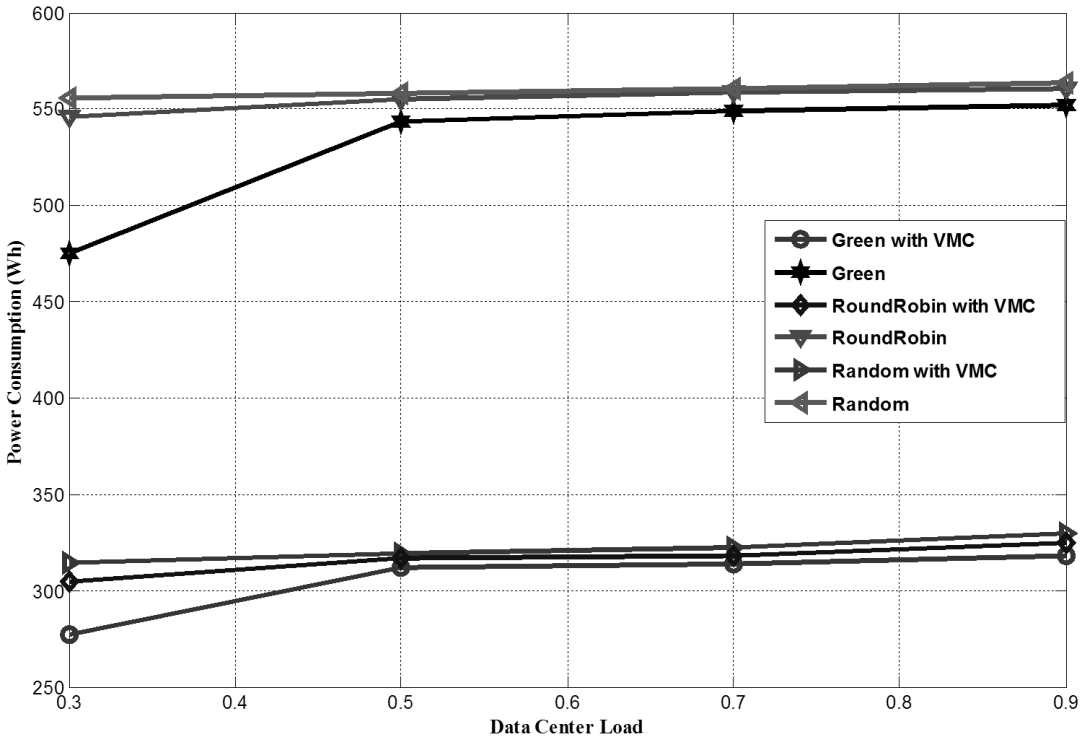

Fig. 4.3. Power Consumption of all Servers

In Figure 4.4, the results are plotted for the power consumption of all the switches in the cloud data center. When we perform VM migration in the data center, the network traffic increases due to the fact that many tasks are being transferred from one server to the other through the communication links and the switches. This causes congestion at the links and the switches. Due to the increased load, the communication links and switches consume more power than they would consume in the absence of VM migration. The results in Figure 4.4 affirm our intuition as it can be observed that the power consumption has increased when task migration is

performed. It can also be noted that as the load of the data center increases the power consumption in case of VM migration is decreased because with the increase in the load of the data center there is less room for VM migration. The power consumption at the switches and the communication links is almost uniform when VM migration is not performed in the aforementioned schedulers.

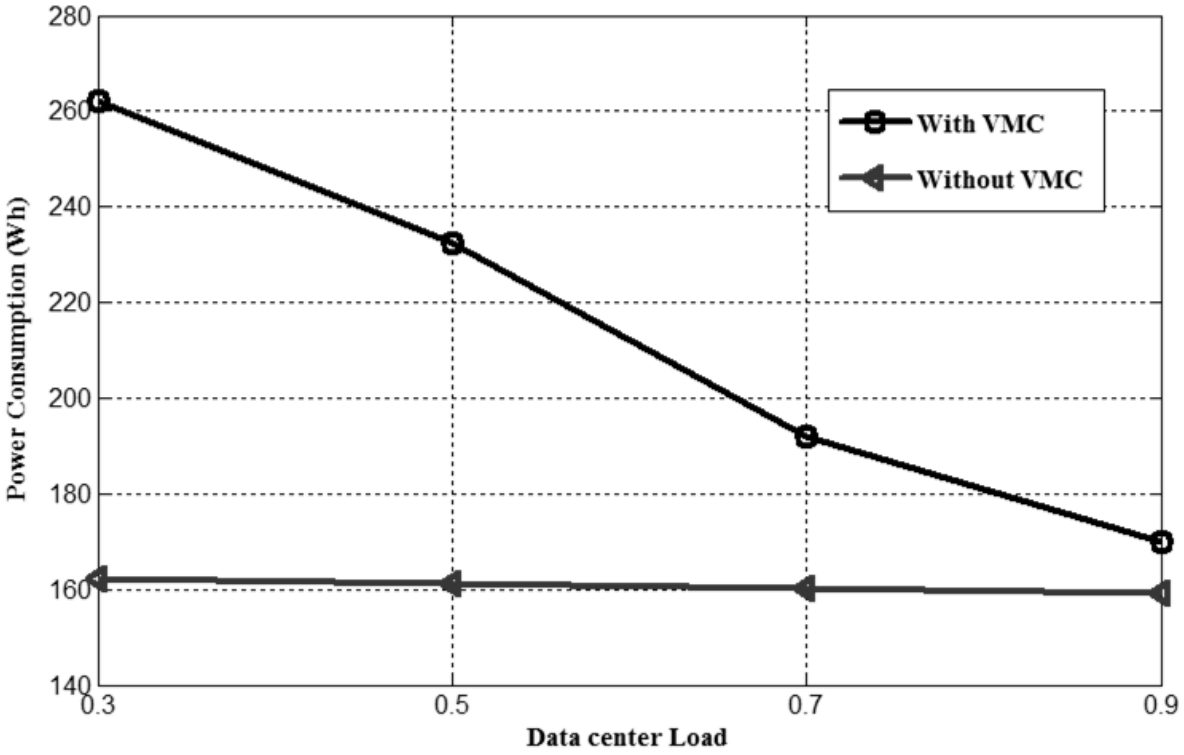

Fig. 4.4. Power Consumption of all Switches and Communication Links

Finally we plot the total power consumption of all the switches and servers (the net sum of power consumption of servers, switches and the communication links) in the cloud data center. Figure 4.5 depicts the results of the net power consumption of the data center both with and without task migration. By combining the effect of the power consumption of the servers and the switches we are still achieving power savings although the switches and the communication links were using more power when VM migrations increased. When the load of the data center is

increased the net power consumption decreases because at this point the power consumed by the switches and communication links is significantly decreased as depicted in fig 4.4.

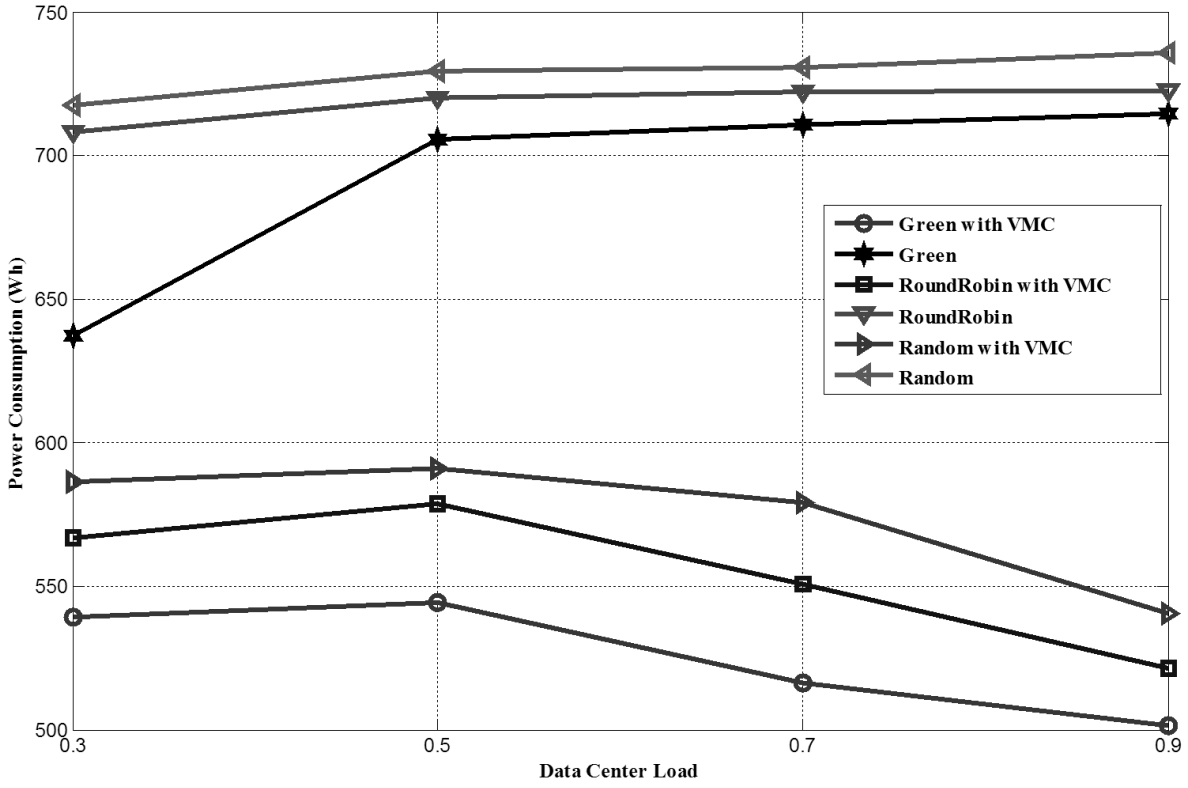

Fig. 4.5. Net Sum of Power Consumption of Switches and Servers in the Data Center

4.5. Conclusion

The work is further extended and the effect of task consolidation is studied and analyzed. By consolidating the tasks on a fewer number of servers the overall power consumed can be significantly reduced. The tasks are first allocated to suitable servers until all the tasks are exhausted. The idle servers are then turned off by using DTVS. The Virtual Machine (VM) monitor checks for under-utilized, partially filled, over-utilized, and empty servers. The VM monitor then migrates the tasks to suitable servers for execution if a set of conditions is met. By this way, many servers those were under-utilized get free and are turned off by using DTVS to

save power. Simulations results confirm our study and a substantial reduction in the overall power consumption of the cloud data center is observed.

4.6. References

[4.1] E. R. Masanet, R. E. Brown, A. Shehabi, J. G. Koomey, and B. Nordman. "Estimating the energy use and efficiency potential of US data centers." Proceedings of the IEEE 99, no. 8 (2011): 1440-1453.

[4.2] R. Brown, "Report to congress on server and data center energy efficiency: Public law 109-431." Lawrence Berkeley National Laboratory (2008).

[4.3] R. Bolla, F. Davoli, R. Bruschi, K. Christensen, F. Cucchietti, and S. Singh. "The potential impact of green technologies in next-generation wireline networks: Is there room for energy saving optimization?." Communications Magazine, IEEE 49, no. 8 (2011): 80-86.

[4.4] M. Webb, "SMART 2020: enabling the low carbon economy in the information age, a report by The Climate Group on behalf of the Global eSustainability Initiative (GeSI)." Global eSustainability Initiative (GeSI), Tech. Rep (2008).

[4.5] J. G. Koomey, "Worldwide electricity used in data centers. "Environmental Research Letters 3, no. 3 (2008): 034008.

[4.6] L. Wang, and S. U. Khan. "Review of performance metrics for green data centers: a taxonomy study." The Journal of Supercomputing 63, no. 3 (2013): 639-656.

[4.7] Trends, Datacom Equipment Power. "Cooling Applications." ASHRAE, http://www.ashrae.org (2005).

[4.8] C. L. Belady, "In the data center, power and cooling costs more than the it equipment it supports." Electronics cooling 13, no. 1 (2007): 24.

[4.9] L. A. Barroso, and U. Hölzle. "The case for energy-proportional computing." Computer 12 (2007): 33-37.

[4.10] X. Fan, W. D. Weber, and L. A. Barroso. "Power provisioning for a warehouse-sized computer." In ACM SIGARCH Computer Architecture News, vol. 35, no. 2, pp. 13-23. ACM, 2007.

[4.11] M. D. De Assuncao, J. P. Gelas, L. Lefevre, and A. C. Orgerie. "The Green Grid'5000: Instrumenting and using a Grid with energy sensors." In Remote Instrumentation for eScience and Related Aspects, pp. 25-42. Springer New York, 2012.

[4.12] . P. Ranganathan, P. Leech, D. Irwin, and J. Chase. "Ensemble-level power management for dense blade servers." In ACM SIGARCH Computer Architecture News, vol. 34, no. 2, pp. 66-77. IEEE Computer Society, 2006.

[4.13] The green grid consortium 2011. URL http://www.thegreengrid.org. (Accessed June 2015)

[4.14] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, Al. Ho, R. Neugebauer, I. Pratt, and A. Warfield. "Xen and the art of virtualization." ACM SIGOPS Operating Systems Review 37, no. 5 (2003): 164-177.

[4.15] C. Clark, K. Fraser, S. Hand, Jacob G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. "Live migration of virtual machines." In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2, pp. 273-286. USENIX Association, 2005.

[4.16] C. D. Alfonso, M. Caballer, F. Avarruiz, and V. Hernandez, "An energy management system for cluster infrastructures," Journal of Computer and Electrical Engineering, vol. 39, no. 8, June 2013.

[4.17] B. Aksanli, J. Venkatesh, L. Zhang, T. Rosing: Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. In: Proc. of the 4th Workshop on Power-Aware Computing and Systems, pp. 5:1–5:5. ACM (2011).

[4.18] L. Barroso, U. Holzle: "The case for energy-proportional computing." IEEE Computer 40(12), 33–37 (2007).

[4.19] A. Beloglazov, R. Buyya, Y. Lee, and A. Zomaya: A taxonomy and survey of energy efficient data centers and cloud computing systems. Advances in Computers 82(2), 47–111 (2011).

[4.20] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Dang, K. Pentikousis: Energy-efficient cloud computing. The Computer Journal 53(7), 1045–1051 (2010).

[4.21] T. Brey, L. Lamers: Using virtualization to improve data center efficiency. The Green Grid, Whitepaper 19 (2009).

[4.22] I. Goiri, R. Beauchea, K. Le, T. Nguyen, M. Haque, J. Guitart, J. Torres, R. Bianchini: Greenslot: scheduling energy consumption in green datacenters. In: Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 20:1–20:11. ACM (2011).

[4.23] E. Harney, S. Goasguen, J. Martin, M. Murphy, M. Westall: The efficacy of live virtual machine migrations over the internet. In: Proc. of the 2nd International Workshop on Virtualization Technology in Distributed Computing. ACM (2007) 16. Lef`evre, L., Orgerie, A.: Designing and evaluating an energy efficient cloud. The Journal of Supercomputing 51(3), 352–373 (2010).

[4.24] M. Lin, A. Wierman, L. Andrew, E. Thereska: Dynamic right-sizing for powerproportional data centers. In: Proc. of the IEEE INFOCOM, pp. 1098–1106. IEEE (2011).

[4.25] S. Srikantaiah, A. Kansal, F. Zhao: Energy aware consolidation for cloud computing. In: Proc. of the 2008 Conference on Power Aware Computing and Systems, p. 10. USENIX Association (2008).

[4.26] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing." Future generation computer systems 28, no. 5 (2012): 755-768.

[4.27] K. Bilal, M. Manzano, S. U. Khan, E. Calle, K. Li, and A. Zomaya, "On the characterization of the structural robustness of data center networks," IEEE Transactions on Cloud Computing, Vol. 1, No. 1, 2013, pp. 64-77.

[4.28] K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal, C.-Z. Xu, and A. Y. Zomaya, "Quantitative Comparisons of the State of the Art Data Center Architectures," Concurrency and Computation: Practice and Experience, vol. 25, no. 12, pp. 1771-1783, 2013.

[4.29] The Network Simulator NS2: available at http://www.isi.edu/nsnam/ns (accessed may 2015)

[4.30] D. Kliazovich, P. Bouvry, and S. U. Khan. "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers." The Journal of Supercomputing 62, no. 3 (2012): 1263-1283.

[4.31] D. Kliazovich, S. T. Arzo, F. Granelli, P. Bouvry, and S. U. Khan. "e-STAB: energy-efficient scheduling for cloud computing applications with traffic load balancing." In Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, pp. 7-13. IEEE, 2013.

[4.32] D. Kliazovich, P. Bouvry, and S. U. Khan. "DENS: data center energy-efficient network-aware scheduling." Cluster computing 16, no. 1 (2013): 65-75.

[4.33] B. P. Rimal, E. Choi, and I. Lumb. "A taxonomy and survey of cloud computing systems." In INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on, pp. 44-51. Ieee, 2009.

5. CONCLUSIONS

With the growing demand of HPC for heavy computations, the power cost of the aforementioned is acting as a limiting factor and needs to be controlled and overcome using power-aware system solutions. This paper underlines the role of voltage level of nodes in cluster's power consumption and presents a methodology, termed as PAJS, that combines energy-efficient job scheduling with node awareness. We have analyzed and compared eight task scheduling techniques. The role of power optimization in modern HPC is emphasized and the proposed PAJS approach optimizes the tradeoff between energy efficient nodes (to reduce the amount of energy consumed) and performance aware patterns (to minimize the makespan). G-Max, G-Deadline and G-Min performed better for all sized workloads as compared to other heuristics. The minimum overall reduction in the energy consumption using DTVS was 10.98%. Nevertheless, the maximum energy savings using DTVS is 31.71% as compared to when DTVS was not employed. For small-sized workloads GenAlgo-DVS also yielded better results in terms of mean makespan and mean energy consumption. However, for large-sized workloads ObjFunc performed well in addition to G-Max, G-Deadline, and G-Min.

The simulation results approve the superior performance of all the heuristics of the proposed methodology (PAJS) in terms of reduction in the makespan and energy consumption of the nodes comprising the HPC. The work presented here is not just restricted to clusters but can easily be adapted to grids, workstations, and datacenters. Both on the practical and theoretical point of view, we have introduced a coherent framework for the optimization of power in various resource scheduling strategies in HPC.

The work is further extended and the effect of task consolidation is studied and analyzed. By consolidating the tasks on a fewer number of servers the overall power consumed can be

significantly reduced. The tasks are first allocated to suitable servers until all the tasks are exhausted. The idle servers are then turned off by using DTVS. The Virtual Machine (VM) monitor checks for under-utilized, partially filled, over-utilized, and empty servers. The VM monitor then migrates the tasks from under-utilized servers to suitable servers for execution if a set of conditions is met. By this way, many servers those were under-utilized get free and are turned off by using DTVS to save power. Simulations results confirm our study and a substantial reduction in the overall power consumption of the cloud data center is observed.