Rack Aware Scheduling in HPC data centers

An energy conservation strategy

Vikas Ashok Patil
Department of Computer Science
University at Buffalo, State University of New York,
Buffalo, USA
vikasash@buffalo.edu

Dr. Vipin Chaudhary
Department of Computer Science
University at Buffalo, State University of New York,
Buffalo, USA
vipin@buffalo.edu

Abstract— **Energy consumption in high performance computing data centers has become a long standing issue. With rising costs of operating the data center, various techniques need to be employed to reduce the overall energy consumption. Currently, among others there are techniques that guarantee reduced energy consumption by powering on/off the idle nodes. However, most of them do not consider the energy consumed by other components in a rack. Our study addresses this aspect of the data center. We show that we can gain considerable energy savings by reducing the energy consumed by these rack components. In this regard, we propose a scheduling technique that will help schedule jobs with the above mentioned goal. We claim that by our scheduling technique we can reduce the energy consumption considerably without affecting other performance metrics of a job. We implement this technique as an enhancement to the well known Maui scheduler and present our results. We compare our technique with various currently available Maui scheduler configurations. We simulate a wide variety of workloads from real cluster deployments using the simulation mode of Maui. Our results consistently show about 7 to 14% savings over the currently available Maui scheduler configurations. We shall also see that our technique can be applied in tandem with most of the existing energy aware scheduling techniques to achieve enhanced energy savings.**

***Keywords-** Datacenter, Energy aware scheduling, Rack aware scheduling (key words)*

I. INTRODUCTION

The energy used by the current data centers is significant. The EPA report to the US Congress on “Server and Data Center Efficiency” [1] estimated a usage of 61 billion kilowatt hours (kWh) in 2006. The same report predicted its rise to 100 billion kWh in 2011. The carbon emission equivalent for this amount of energy consumption was about 846 million metric tons in 2006. The report highlights the magnitude of the energy use by the current data centers and the need for aggressive energy conservation strategies to be adopted by the operators of the data centers. Due to the ever increasing business and scientific computing needs, this problem is exacerbated and has resulted in a significant rise in operating costs.

Electricity cost is one of the major operation costs of a data center [2]. Reducing the energy consumption in a data center can significantly reduce the operating costs. This has sparked serious research interests in both the academic and commercial research groups. As a result, there have been considerable improvements at the software, hardware and infrastructure levels of the data center ever since.

This view of reducing energy consumption to cut down the operating costs in a data center was deemed contradictory in High Performance Computing (HPC) data centers, as achieving improvements in performance has always been the key focus. However, even HPC data-centers are plagued by the increasing costs due to energy consumption. The most power consuming super computer runs at 6.95 megawatts (MW) and IBM’s roadrunner, the No 2 in the TOP 500 list consumes 2.48 MW [3]. This has sparked significant research interest to reduce the energy consumption of these specialized data centers.

Energy conservation by improvements in job scheduling techniques has been one such area of active research. Job scheduling is an important aspect of any HPC data center. The function of the job scheduler is to allocate the data center resources such as CPU's, storage and network, to the incoming jobs and increase the overall cluster utilization. From a energy conservation point of view it has to do much more than that. It needs to allocate the resources that will reduce the overall energy consumed by the cluster without deviating much from the job turnaround times. A few current day job schedulers take this issue of power conservation into consideration and are being widely used across academic and commercial HPC installations [3]. These HPC installations use this feature of job schedulers to turn on/off the idle nodes or even perform dynamic scaling of power supply to the cluster components, thereby conserving considerable energy. Most of these job schedulers are based at node level granularity, which means the scheduler views the data center as a set of nodes (and sometimes also as the node’s sub-components). Though scheduling at the node level is consistent with the resource demands for jobs, we consider it unsatisfactory from an energy conservation perspective.

In this paper we show that we can achieve additional savings in power by considering the job scheduling at the rack level granularity. A rack is an enclosure of nodes and often has certain additional components associated with it.

These components are Interconnect bays, Fans, Power distribution unit (PDU), Blowers, etc. We propose a set of enhancements to the scheduling algorithm for performing resource allocations at the rack level granularity. We also implement it as an enhancement to the existing Maui scheduler and test our proposal on real datacenter workloads. The Maui scheduler is one of the widely used schedulers in HPC data centers [4].

This paper also highlights certain statistics of the energy costs associated at the rack level. This data is gathered from an operational commercial HPC data center. The data is utilized to perform simulations using the simulation mode of the Maui scheduler. The statistics gathered from these simulations show considerable energy conservations.

The remainder of this paper is organized as follows: Section II briefly describes the related work with regard to enhancements in job scheduling from energy conservation point of view. In Section III we provide the current rack level power statistics. In Section IV we introduce the Maui scheduler and its high level algorithm. In Section V we briefly explain the benefit of node allocation by considering the rack level granularity. In Section VI we propose our algorithm and give details of its implementation with reference to the Maui Scheduler. In Section VII we describe the experimental setup and Section VIII describes the results obtained. We then conclude our work in Section IX and also discuss the future enhancements.

II. Related Work

Various techniques have been proposed for improved power management in HPC clusters. SLURM [5] is a widely used resource management and scheduling system for supercomputers. It has a power management facility to put the idle nodes in a lower power state. It has facilities to contain the surge in the workload and alters the node states gradually. However, not much research has been done to provide a better power management policy [2].

In [6], Pinheiro et al. proposed a load concentration policy. This turns on or off cluster nodes dynamically according to the workload imposed on the system. Chase et al. [7] take this further and propose Muse which uses Service Level Agreement (i.e., SLA) to adjust active resources by making a trade-off between service quality and power consumption. These works can be classified as dynamic cluster reconfigurations. Chen et al. [8] applied dynamic voltage frequency scaling techniques along with dynamic cluster reconfiguration to achieve much more improvements.

Scheduling solutions based on the use of virtualization have also been proposed. The general idea is to utilize VM consolidation or intelligent node allocation and achieve overall energy conservation [2]. Verma et al. [9] investigated aspects of power management in HPC applications by determining the VM placements based on CPU, cache and Memory footprints. Dhiman et al. [10] proposed a multi-tiered software system called as vGreen, a solution which takes migration overhead into consideration. Nathuji et al. [11] suggested Virtual Power, which defines virtual power states for the servers based on their scheduling policies as well as the CPU frequency. They used power management hints provided by the guest OS to implement global or local power policies across the physical machines. In [24] Hermenier et al propose a consolidation manager, *Entropy*, which uses constraint programming (CP) to solve the VM placement problem. The idea is to define the consolidation problem as a set of constraints and use a standard library such as Choco to solve the problem. Berral et al [25] use a machine learning based approach for power modeling where they predict the power requirements for a VM and make VM placement and migration decisions based on this prediction. Most of these server consolidation techniques focus on utilizing as few nodes as possible, but they do not concentrate on the packing of nodes based on actual node locations.

Topology aware scheduling considers the scheduling of jobs, based not only on the properties of the requested machines but also on the data center properties. However, largely this has been to do with machine interconnections. Gabrielyan et al. [12] discuss one such strategy, but concentrate on inter-node collective communication aspects. There has also been significant work in thermal management such as thermal management system by Heath et al. [13] and a study of temperature aware workload placement by Moore et.al [14].

In [23], Ranganathan et al. propose an ensemble (enclosure) level power management scheme. However they mainly focus on power capping at the enclosure level. A controller exists at the enclosure level which is responsible to ensure that the total power consumption of the enclosure does not cross a predefined limit. This scheme does not consider a cluster wide view and is in-effective from a parallel job scheduling perspective.

Most of these cluster wide techniques consider the physical node level granularity. None of them view the data center from rack perspective. We show that we can consider the rack level granularity and achieve additional energy savings. Our technique can be applied in tandem with most of these other techniques. Thereby we can incorporate our technique into existing data center installations and realize the additional power conservation benefits.

III. Rack Level power statistics

We gathered statistics from a commercially running HP Bladesystem p-class enclosure [15]. A typical rack consists of 2 to 3 enclosures. We use the term rack invariably to mean enclosure in this paper as it does not affect our discussion and our technique can be evenly applied with very few changes. We gather the runtime power statistics of such racks using the proprietary monitoring tools provided by HP [16]. This information is typical of any rack installation and scales evenly across different racks. We utilize this statistics to estimate the energy savings achieved by the implementation of our technique.

The HP rack that we considered can support up to 16 blades and has 10 enclosure fans. The fans consume about 500 W of power on an average. The interconnect bays connects the nodes within a rack. They consume about 133 W of power. These power consumptions are immaterial of the node states. The same amount of power is consumed even when the nodes are turned off completely.

In addition typically there are blowers associated with each rack which help to bring down the rack temperature. We can turn off these blowers by the use of our technique and conserve much more energy. For simplicity, we do not include the savings from the blowers in our results, which we consider to be also significant. We also exclude the energy consumed/savings associated with nodes for our discussion.

We argue that turning off the racks through the use of remotely controlled power distribution units (PDU's such as APC's Switched Rack PDU) [17] can significantly enhance the power conservation.

IV. Maui Scheduler

The Maui scheduler is one of the widely used schedulers in HPC clusters [4, 18]. Like many other batch schedulers, Maui determines which job needs to be run when and where and informs the resource manager. Torque is one of the most commonly used resource managers [19]. Its function is to issue commands in interest of Maui's scheduling decisions and also provide up to date information about the cluster. Torque also acts as an interface for the users to interact with their jobs. Thus Maui learns the job and user information from Torque.

Maui is well known for its highly configurable components. Changing job priorities based on adjustable parameter weights and node allocation policies are some of the things among many that are relevant for our discussion. Apart from being run along with a resource manager such as torque, Maui can also be run in simulation mode, where it can simulate years of workload in just a few hours. Thereby, Maui is a very powerful tool to study scheduling of different workloads in HPC clusters.

At every periodic time interval Maui performs one cycle of the following steps, which is called as a scheduling iteration [4]. (Only the steps relevant to our discussion are described here)

a) Change the priorities of the jobs in the job input queue. (Job Priority polices like weights to queue time, quality of service, usage, etc. are applied here)
b) Choose a batch of jobs from the input job queue based on the new priorities.
c) Allocate nodes for this batch of jobs one at a time based on the node allocation policy and inform the resource manager.
d) Backfill – a scheduling optimization.

We are interested in step (c) of the scheduling iteration described above. We implement our algorithm as a node allocation policy plug-in. We also discuss a possible enhancement to step (a) which affects the job priorities.

V. Concept

Our goal is to bring rack awareness into the scheduler. Mainly we would want to allocate nodes more intelligently to a particular job which will reduce the total number of racks being utilized during the job's execution.

TABLE I. THE STATE OF THE CLUSTER WHEN A NEW JOB J_T ARRIVES

RACK ID	NUMBER OF BUSY NODES	NUMBER OF FREE NODES
1	0	3
2	2	1
3	1	2

Consider for example a cluster depicted by Table (1). The cluster has 3 racks with 3 nodes each. Each rack has a set of free nodes and occupied nodes. The occupied nodes are being utilized by currently running jobs.

Suppose at this time a new job J_t has to be allocated nodes on the cluster. Assume J_t requires 3 nodes to start running. If we allocate one node from each rack, we will be using all the three racks which will increase the power consumption by an additional amount due to the rack components that we described in Section III. Had we allocated all the 3 nodes from the rack with Rack Id 1, we would still end up utilizing all the three racks. This would be how a possible rack unaware job scheduler would perform the node allocation for the job J_T.

However, a rack aware scheduler like ours would choose one node from the rack with Rack Id 2 and 2 nodes from the rack with Rack Id 3, resulting in keeping the rack with Rack Id 1 shut off. Thereby, we can have more energy savings. Our algorithm takes advantage of this form of node allocation as shown in the following section.

VI. ALGORITHM AND IMPLEMENTATION DETAILS

We formulate and implement the following algorithm as a node allocation policy in the Maui scheduler. The reason we choose the rack with the maximum remaining time is that, we expect that particular rack to remain occupied for longer period of time when compared to any of the other

ALGORITHM: Allocate Nodes to a job J_t.

INPUT: Node Requirements for job J_t, Rack occupancy.
OUTPUT: Node Allocation for job J_t.

1. *Categorize the racks as utilized and not-utilized.*
2. *Sort the racks in the utilized category based on the maximum remaining time of the jobs on the nodes in that particular rack.*
3. *Allocate nodes from the rack with the maximum remaining time first and then with lesser maximum remaining time.*
4. *If the request is still not satisfied we allocate from the not-utilized racks.*

racks. Thus by choosing the rack with the job that has longer maximum remaining time, we will keep certain number of racks always utilized. To avoid this situation we make certain additions to the proposed algorithm.

The additions to the algorithm keep track of the rack utilization over time and categorize the racks as utilized, over-utilized and relaxed. These additions are incorporated to reduce the repeated usage of the same hardware. These additions do not cause note worthy deviation from our results. Therefore for simplicity of the discussion we exclude this aspect.

This algorithm affects the STEP (c) of the Maui scheduling iteration. Maui provides a “LOCAL” Node Allocation Policy, which calls a well defined function MlocalJobAllocateResources() with input parameters being a list of eligible nodes and job details (including the job’s node requirements). This function is called when we need to allocate nodes for a job that is selected to run based on its priority. We extend this particular function to implement the above described algorithm.

We also model the racks into the frame data structure of Maui. The frame data structure till date mainly served the purpose of node organization, for displaying the node information properly. By using this data structure of Maui most of the scheduler code remains untouched. We have also added code to gather and generate runtime statistics related to the energy consumption of the cluster. We take care that the statistics generation code is not included in the scheduling time calculation, which is discussed in detail below.

VII. EXPERIMENTAL SETUP

The simulation mode of Maui is a useful feature which helps to simulate different workloads for various cluster and scheduler configurations. It generates lot of useful statistics which can be used to further tune the scheduler to suit your cluster environment. We use the Maui scheduler in this mode to test our implementation.

In the simulation mode Maui accepts a resource trace file and workload trace file, which depict cluster configuration and workload logs respectively. We utilize a reduced version of the workload log of HPC2N [20], which is about 1.5 years of cluster log. The HPC2N is a 120 node cluster with 2 processors per node. We generate the resource trace file for different cluster utilizations for our experiment.

VIII. RESULTS

Figure (1) shows the utilizations of different real world clusters. The log information is sourced from various academic and commercial HPC installations [21, 22]. We see that most of these clusters are around 40% to 50% utilization and rarely do we find high utilization clusters. Hence we can safely assume that there are many clusters in the real world which have 40% to 50% utilization. On an average they have about 45% utilization. We shall see that our algorithm is very well suited for clusters which fall in this range of utilization. We have also used this assumption for few of our experiments. To mention we compare our algorithm with existing node allocation policies at 45% cluster utilization.

Figure (2) shows how much energy is saved for different percentages of cluster utilizations by the use of our technique. We use the same HPC2N workload trace file, but change the cluster configuration (number of nodes) via the resource trace file to obtain the savings for different utilizations. We see that our algorithm gives considerable savings at lower and middle levels of cluster utilization and the savings decay as the utilization rises. Most of the real world clusters fall in the lower and middle levels of cluster utilization as discussed below.

We define rack utilization as a metric to measure the number of racks utilized over the entire period of the workload. Higher the rack utilization, more energy will be consumed. Figure (3) shows the rack utilization comparison of our algorithm with other node allocation policies currently existing in Maui. The First Available node allocation policy simply selects the nodes it finds first and matches the requirements. The Max Balance allocates the nodes with a balanced set of node speeds. In a homogeneous cluster it simply falls through First Available. The fastest selects the fastest node first. The CPU load selects nodes based on the current CPU node. In min resource those nodes with the fewest configured resources which still meet the job's resource constraints are selected. We see that our algorithm consistently performs better than any other node allocation policy.

Figure (4) shows the energy savings in terms of Megawatt Hours (MWh) in comparison with other node allocation policies described above. We see that our algorithm gives about 7 % more energy savings than the existing best node allocation policy (in terms of energy savings). It gives about 14% more savings when compared to Min Resource which is the default node allocation policy for Maui.

Figure (5) shows the comparison of the average scheduling time for the different node allocation policies. Our algorithm does not lead to any significant increase in the scheduling time. It remains on par with other policies.

Figure (6) compares the change in number of racks for the different node allocation policies. We see that our technique does not lead to frequent changes in the number of racks utilized in consecutive scheduling iterations as compared to other policies. This means that we will have sufficient time to perform the rack power on/off and would keep the cluster utilization stable enough as compared to the other policies. This does not mean that we have to take the rack power on/off decisions at every scheduling iteration. We suggest doing it at some small multiple of scheduling iterations.

Figure (7) shows the savings for different number of nodes per rack, cluster configuration. We see that the savings due to interconnect bays remain more or less consistent, where as the savings due to the fans increases as the number of nodes per rack decrease. This is expected as the number of fans is in direct proportion to the number of nodes in a rack.

IX. CONCLUSION

This paper demonstrates that at rack level granularity we can further enhance existing energy aware scheduling techniques and achieve significant energy conservation. We have consistently demonstrated savings of 7% to 14% over existing node allocation schemes. We have also seen that our technique can be applied in tandem with many other existing energy conservation schemes. As further improvements we can modify the job priorities based on the existing node allocation patterns in the racks. We also need to provide a quantification of the energy savings due to the blowers associated with the racks. We believe these savings would also be significant. Another aspect that we need to consider is the energy loss due to possible increased traffic across the network switches that connect the racks. However, these losses are for communication intensive parallel jobs.

REFERENCES

[1] Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431. *U.S. Environmental Protection Agency ENERGY STAR Program,* August, 2007.

[2] Johnathan Komey, Christian Belady, Michael Patterson, Anthony Santos, Klaus-Dieter Lange, "Assessing trends over time in performance, costs and energy use for servers", *LLNL, Intel Corporation ,Microsft Corporation and Hewlett-Packard Corporation* released on the web on August 17, 2009.

[3] Yongpeng Liu and Hong Zhu, "A survey of the research on power management techniques for high-performance systems", *Software Practive and Expierence Journal*, John Wiley and Sons, Inc, vol. 40, issue 11, October 2010.

[4] David Jackson, Quinn Snell and Mark Clement, "Core Algorithms of the Maui Scheduler", *SprinkerLink*, January 01, 2001.

[5] LLNL, HP and Bull, "The Simple Linux Utility for resource Management (SLURM)". Available at http://www.llnl.gov/linux/slurm/. Revision 2.0.3, June 30, 2009.

[6] Pinheiro E, Bianchini R, Carrera E, Health R, "Load balancing and unbalancing for power and performance in cluster-based systems", Technical Report DCS-TR-440, Department of Computer Science, Rutgers University, May 2001.

[7] Chase J, Aderson D, Thakar P, Vahdat A, Doyle R, "Managing energy and server resources in hosting centers", *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP'01)*, Canada, October 2001.

[8] Chen Y, Das, Qin W, Sivasubramaniam A, Wang Q, Gautam N, "Managing server energy and operational costs in hosting centers", *Proceedings of the 2005 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'05)*, Canada, June 2005.

[9] Verma A, Ahuja P, Neogi A, "Power-aware dynamic placement of HPC applications", *Proceedings of the 22nd International Conference on Supercomputing (ICS'08),* Greece, June 2008.

[10] Dhiman G, Marchetti G and Rosing T, "vGreen : A System for Energy Efficient Computing in Virtualized Environemnts", *ISLPED, California, USA* , August 2009.

[11] R. Nathuji and K. Schwan. "VPM Tokens: Virtual Machine-Aware Power Budgeting in Datacenters". In *High Performance Distributed Computing*, June 2008.

[12] Gabrielyan E, Hersch R D, "Network Topology aware scheduling of collective communications", *10th International Conference of Telecommunications*, March 2003.

[13] Heath T, Centeno A, george P, Ramos L, Jaluria Y, Bianchini R, "Mercury and Freon temperature emulation and management for server systems", *ASPLOS*, October 2006.

[14] Moore J, Chase J, Ranganathan P, Sharma R,"Temperature-aware workload placement in data centers",*USENIX*,April 2005.

[15] HP BladeSystem p-Class Infrastructure Specification - http://h18004.www1.hp.com/products/quickspecs/12330_div/12330_div.html.

[16] HP Systems Insight Manager, version 6.2.

[17] Product description of APC Switched Rack Power Distribution Unit , http://www.apc.com/products/family/ind-ex.cfm?id=70.

[18] Maui Scheduler Administrative Guide, Version 3.2, http://www.clusterresources.com/products/maui/docs/mauiadmin.shtml.

[19] Torque Admin Manual, Version 3.0, http://www-.clusterresources.com/products/torque/docs/

[20] HPC2N Log from Parallel Workloads Archive. HPC2N is a linux cluster located in Sweden. http://www.cs.huji.ac.il/-labs/parallel/workload/l_hpc2n/index.html.

[21] Parallel Workload Archive: http://www.cs.huji.ac.il/-labs/parallel/workload/logs.html.

[22] SCD FY 2003, ASR :http://www.cisl.ucar.edu/docs/asr2003/-mss.html.

[23] Hermenier,Lorca, Menaud,Muller,Lawall, "Entropy: a Consolidation Manager for Clusters", *VEE*, Washington, 2009.

[24] Beral, Goiri, Nou, Julia, Guitart, Gavalda, Torres, "Towards energy-aware scheduling in data centers using machine learning", *e-Energy*, Germany, 2010.

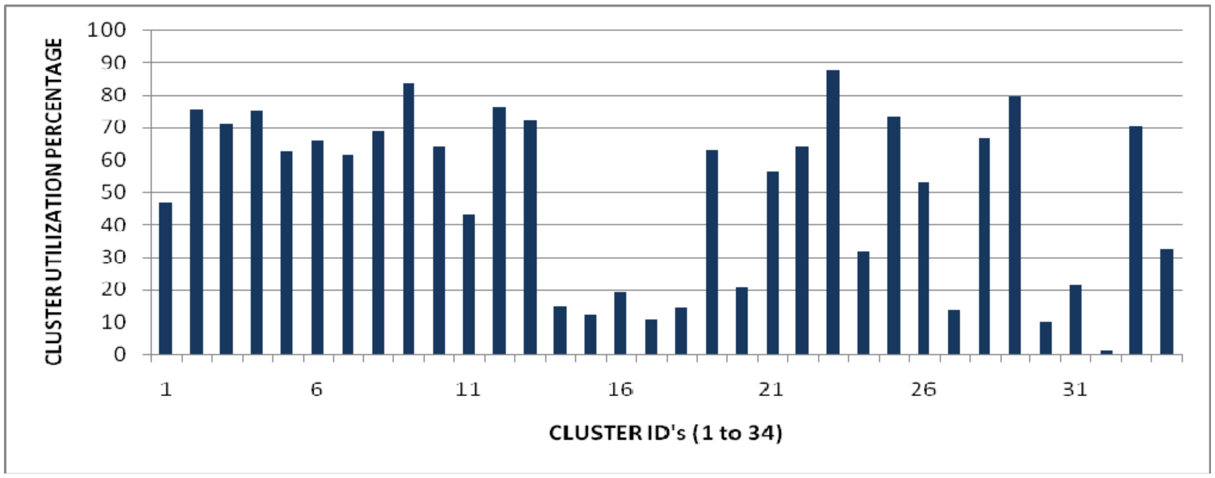

Figure 1: Cluster Utilizations data of academic and commercial clusters

1	NASA iPSC	8	KTH SP2	15	DAS2 fs1	22	SHARCNET	29	NCAR-BlueskyB8
2	LANL CM5	9	SDSC SP2	16	DAS2 fs2	23	LLNL Atlas	30	NCAR-Dave
3	SDSC Par95	10	LANL O2K	17	DAS2 fs3	24	NCAR-Babyblue	31	NCAR-Dataproc
4	SDSC Par96	11	OSC Cluster	18	DAS2 fs4	25	NCAR-BlackForest	32	NCAR-Mouache
5	Early CTC SP2	12	SDSC Blue	19	SDSC DataStar	26	NCAR-BlackForst(2)	33	NCAR-Chinook
6	CTC SP2	13	HPC2N	20	LPC EGEE	27	NCAR-Bluedawn	34	NCAR-Chnookfe
7	LLNL T3D	14	DAS2 fs0	21	LCG	28	NCAR-BlueskyB32		

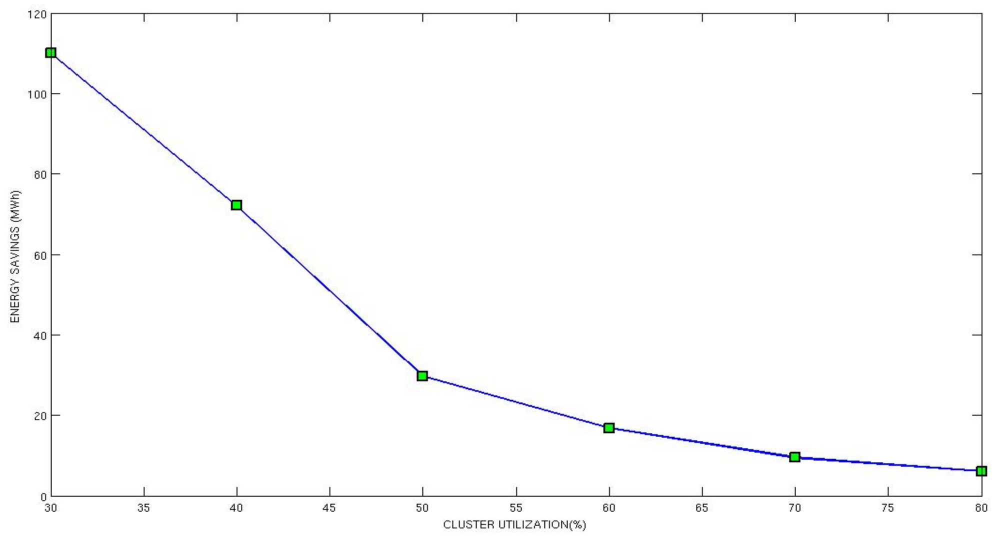

Figure 2: Power Savings for different percentages of cluster utilizations

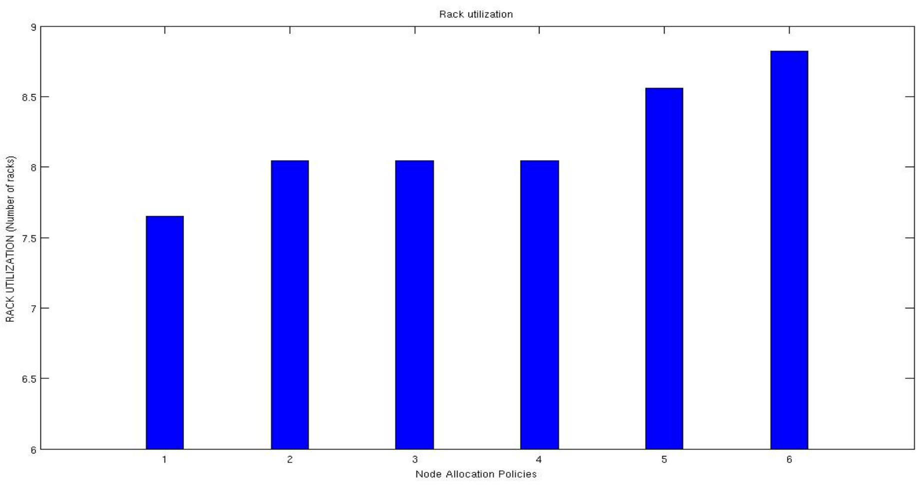

Figure 3: Rack Utilizations for different Node Allocation Policies
Rack Aware (1), First Available (2), Max Balance (3), Fastest (4), CPU Load (5), Min resource (6)

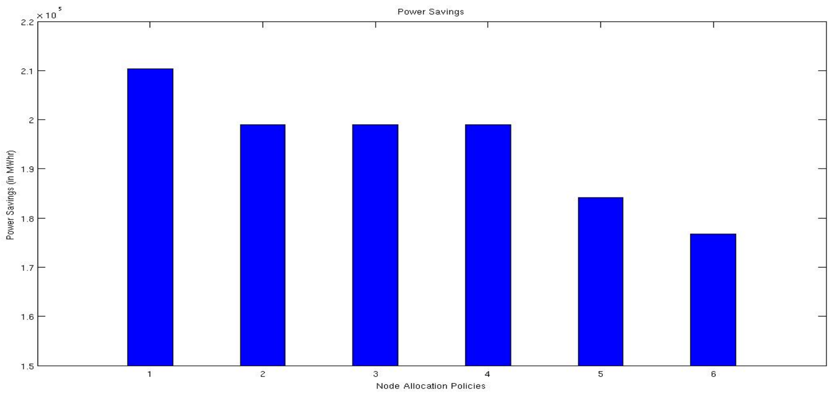

Figure 4: Power Savings for different Node Allocation Policies
Rack Aware (1), First Available (2), Max Balance (3), Fastest (4), CPU Load (5), Min resource (6)

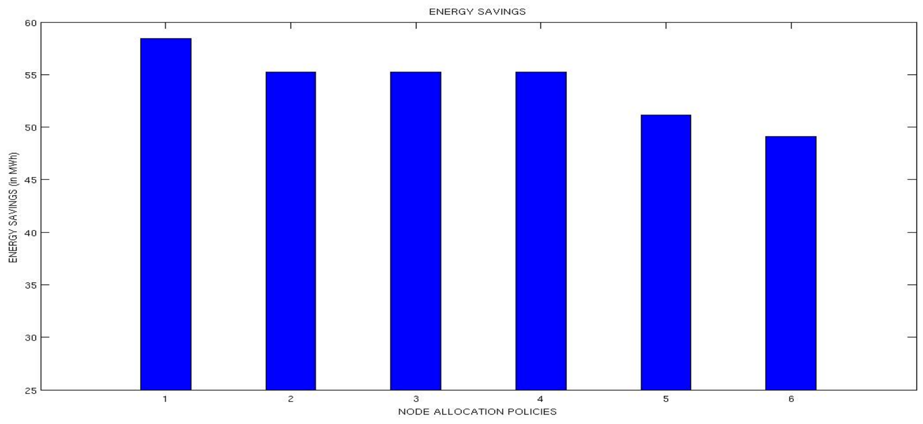

Figure 5: Average Scheduling Times for different node allocation policies
Rack Aware (1), First Available (2), Max Balance (3), Fastest (4), CPU Load (5), Min resource (6)

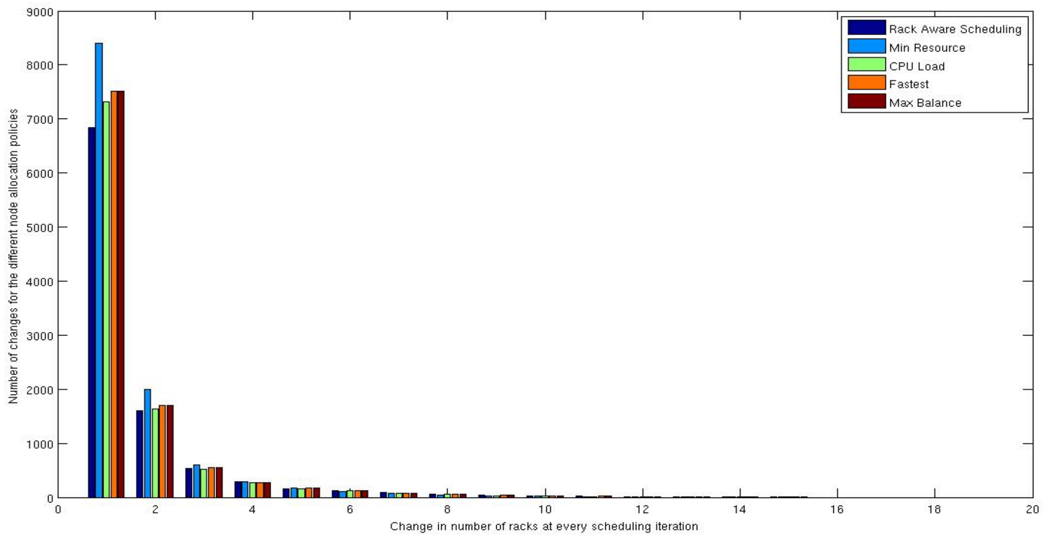

Figure 6: Number of changes in rack count effected by the different node allocation policies at every iteration

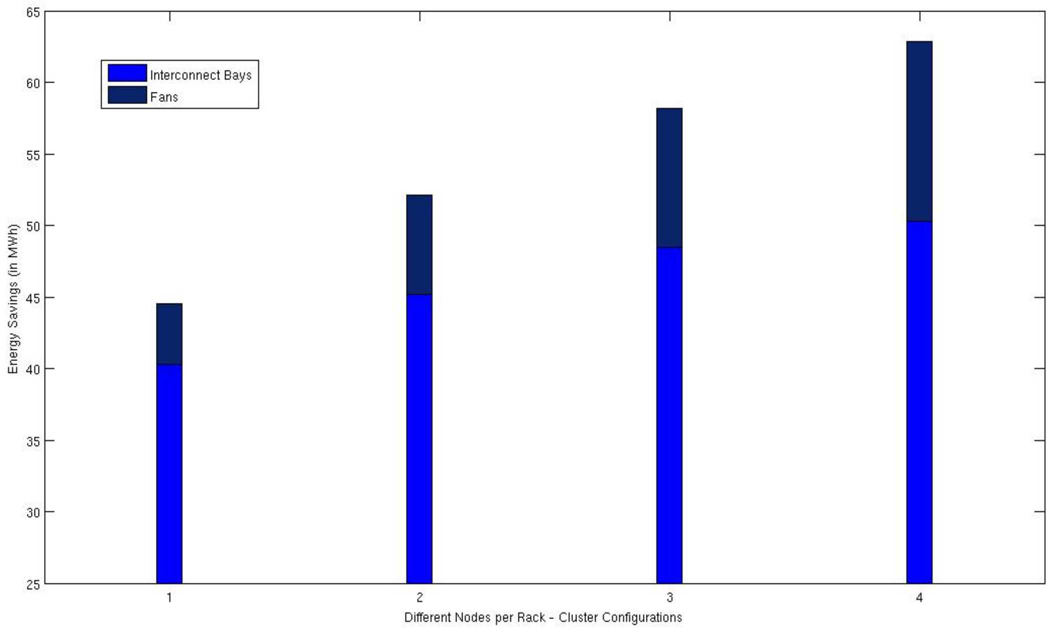

Figure 7: Savings for different number of nodes per rack cluster configuration
8 racks-30 nodes per rack (1), 12 racks-20 nodes per rack (2),
16 racks–15 nodes per rack (3), 20 racks-12 nodes per rack (4)