Asian Journal of Information Technology 15 (8): 1307-1313, 2016
ISSN: 1682-3915

QoS Oriented Node Selection in High Performance Clusters

[1]R. Reshmi and [2]D. Shanthi
[1]Anna University, Chennai, India
[2]Department of Computer Science and Engineering,
PSNA College of Engineering and Technology, Dindigul, Tamil Nadu, India

Abstract: A scheduler has to assign the tasks to heterogeneously distributed computing sites to process with the objective to achieve the customers' Quality of Service (QoS) requirements as well as to optimize the performance. In this study, a QoS oriented mechanism, evaluates nodes in a high performance cluster based on various static and real-time parameters. It selects the most appropriate node to perform the task at hand with regard to their quality of service. The nodes are gauged on parameters, spanning a broad spectrum from power usage to fan speed. This system offers an inherent flexibility as it can be tailored to bias the selection towards a particular parameter or can be maintained on a balanced plane. In particular, the focus was on designing a power aware system, where power usage, CPU temperature etc. play a critical role in determining the QoS of a node. The result of our experiments that upholds the optimality of QoS based selection algorithm is also presented.

Key words: High-performance systems, quality-of-service, power aware systems, job scheduling, tailored

INTRODUCTION

The Quality of Service (QoS) provides an industry-wide set of standards and mechanisms for ensuring high-quality performance for critical applications. High performance clusters (Katz *et al*., 2002; Chare *et al*., 2001) were designed to meet the heavy requirements of such applications are called as massively parallel systems. Massively parallel algorithms that have huge resource requirements are needed to be run in quick succession for obtaining an optimal scheduling. The scheduling is an operation which select among own jobs stored in a buffer to be transmitted over a specific link. These clusters are constituted of many computing nodes comprised of multiple processors linked together as a single system. Performance is a critical aspect of these systems, as very often, the applications are simulations of high risk operations. The choice must be taken in a very small period of time and it is related to job turnaround time. Scheduling amounts to allocating the jobs and resources optimally to the distributed nodes with diverse Quality-of-Service (QoS) requirements. Scheduling oriented algorithms are used in many decision support systems which integrates human factors and agronomics for problem solving (Gacias *et al*., 2012). In large scale computing or in big Data context, efficient scheduling algorithms play an essential role. Sfrent and Pop (2015) deals with the problem of scheduling a set of jobs across a set of machines and specifically analyzes the behavior of the system at very high loads, which is specific to Big Data processing. Development of high-performance distributed applications are extremely challenging because of their complex runtime environment coupled with their requirements of high-performance and Quality of Service (QoS). Such applications typically run on a set of heterogeneous machines with dynamically varying loads, connected by heterogeneous networks possibly supporting a wide variety of communication protocols. In spite of the size and complexity of such applications, they must provide the high-performance and QoS, mandated by their users. In order to achieve the goal of high-performance, they need to adaptively utilize their computational and communication resources. Performance of a scheduler is based on how efficiently the resources of a cluster are utilized while maintaining the QoS provided to end-users within a pre-negotiated range. Selection of a node to perform a task at hand is crucial as this in turn affects the performance of the cluster. The nodes offer different quality of services (QoS) based on various static and dynamic parameters and it makes sense to choose the node with the best QoS to serve a requirement. QoS model aims to provide service stability and dependability on individual resources, leading to standard parts which enable predictable performance.

QoS of nodes in a high performance cluster has garnered a lot of attention, as it directly reflects performance and scalability of the system (Norris *et al*., 2004). Various parameters can be used in conjunction to determine the QoS of a node. Focusing on energy efficiency, we have devised

Corresponding Author: R. Reshmi, Anna University, Chennai, India

a power aware mechanism for QoS assessment that addresses the relevance of parameters that are indicative of the power consumption in a cluster. The power aware parameters considered were power usage of the nodes, the temperatures of the CPUs that constitute a node and the respective fan speeds. Apart from these, the other parameters like memory usage, network send/receive rate, performance, CPU load, CPU speed and CPU cache memory together determine the QoS of a node. In short, the QoS of a node is the weighted average based on the relative grade of these parameters.

Literature review: Quality-of-service (QOS) provisioning becomes an issue of great importance, concerning communication networks and high performance distributed systems. QoS is specified through a set of measurable performance parameters that quantify and categorize the degree of service quality availed to end-users. Much attention has been focused on characterizing QoS requirements in high performance distributed clusters application level. Scheduling can be improved by focusing the Quality-of-Service which is derived from an objective metrics. Quality of Service plays a critical role in effective resource reservation.

Energy efficient computing clusters using improved hardware design techniques have been developed (Huang and Feng, 2009). Their highly modular approach regulates power consumption while not sacrificing the performance aspect of the cluster. Studies have been reported on managing energy and server resources (Jacobs and Bean, 1963) as well as energy proportional computing (Barroso and Holzle, 2007). Optimizing power consumption in data centers that handle huge quantum of data is presented by Moore *et al*. (2005). Xian-He and Ming proposes a prediction model in GHS system for long-term application-level performance prediction which addresses the challenge of non-dedicated clusters. (Shan *et al*., 2002) introduces mechanisms to correlate contents and priorities of incoming HTTP requests used for server process scheduling with the load balancing policies for Web-server clusters. This approach enables both load balancing and Web QoS.

Crovella propose a policy favoring short connections for static files. The shortest remaining processing time (SRPT) scheduling policy is analyzed by Bansal and Harchol (2001). Other mechanisms and policies for Web QoS, such as operating system control, server-side application-level-only mechanisms (Eggert and Heidemann, 1999), web content adaptation for server resource management, control-theoretical approach for performance guarantees (Abdelzaher and Bhatti, 1999) were discussed. The scientific community is acknowledging the need for efficiency even when high performance is the need of today.

In class of multi-criteria scheduling problems, vector processing can be considered to improve computational time, supported by fine-grained parallel computing Smutnicki *et al*., 2015). Distributed systems are expected to provide QoS that guarantees to users as obligatory by multimedia and other real-time applications.

In this study, we present a mechanism where the different nodes in a high performance cluster are assessed on a multitude of parameters to calculate their QoS which is used as selection criteria to choose the nodes to schedule a job. Our QoS oriented mechanism attempts to strike a tradeoff between efficiency and performance through careful monitoring of the parameters discussed.

MATERIALS AND METHODS

Cluster architecutre: Much of the prior work on performance monitoring for distributed systems had focused on high level architecture of clusters and its implementation issues. In this research, the cluster series promises higher efficiency and better computational facilities for large-scale applications.

Dhakshina-HPC cluster: Dhakshina Cluster Series II ($DKA_{clusterII}$) is a High Performance Computing System developed using the Beowulf Architecture, it has a peak speed of 180 Gflops. Inspiration and abundant factual assistance was drawn from the Linux cluster installations red books distributed by IBM.

Nodes in $DKA_{clusterII}$: The nodes in the cluster are divided based on their function. The functionalities of various nodes are explained below.

User: The user node is the gateway for the outside world to access the cluster. Users may login to the user node machine and compile or run their tasks. Hardware redundancy is recommended on this machine node as, if it fails, users may not access the cluster and use its powerful computational capacity. If the user node is not up and running, the cluster cannot function, because there is no opportunity to run a task. The user node can also be on the same server that is used for management or installation. RAID adapters are used to protect the data on the user node.

Control: The control node can be on the same machine that takes care of management and installation functions. It is

responsible for controlling the cue or batch jobs. The control node provides services such as Dynamic Host Configuration Protocol (DHCP), Domain Name System (DNS) and Network File System (NFS). Redundancy is required on hardware and data because, unlike in the case of compute nodes, the failure of the control node may affect the availability of the entire cluster.

Management: Management is the function that manages and controls the cluster and its components, for example, switches. It is either on dedicated machines called management nodes, or, for budget reasons, shares the machine with other functions, such as the master, installation and compute nodes. It may also be found on a dedicated virtual local area network (VLAN), defined as management and accessible only by the administrator (for security reasons). As the cluster manager, it takes control of the nodes and collects Simple Network Management Protocol (SNMP) alarms.

Storage: The storage function is performed by dedicated machines and is responsible for storing large amounts of data needed by the applications running on compute nodes. The storage node provides a solution to the problem of feeding large amounts of data to the cluster. Since, it is difficult to store large amounts of data on a single server or on one of the cluster nodes, there is a need for a SAN (Storage Area Network) or dedicated servers; the network may be a bottleneck at this point. Storage nodes are equipped with ServeRAID controllers (that give data protection using one of the different RAID levels available), Gigabit Ethernet for higher bandwidth performance.

Installation: The installation node is responsible for the installation of the compute nodes and is sometimes referred to as a staging server or node. It is a server that exports a file system containing the operating system, libraries and all the necessary software for the compute nodes. It has a Gigabit Ethernet PCI adapter to allow more bandwidth and to speed up the cluster installation. An installation server shares a file system that can be accessed via NFS, Web or File Transfer protocol (FTP). It is the only node on the public network, since users need to access it to be able to run their jobs.

Computation: The compute function or node is the computational heart of the cluster. Its main activity is just to perform calculations. The choice of hardware and configuration parameters of the compute node is based on the applications that will be run on the system. The following characteristics are normally considered:

- Processor Type
- Size and speed of L2 cache
- Number of processors per node
- Speed of front-side bus
- Memory subsystem scalability
- PCI bus speed

The application drives the decision on how to build the compute nodes. For example, if the compute node is required to access the cache frequently, the size of the L2 cache and memory subsystem should be considered; a large cache may enhance the performance. On the other hand, applications that use intensive amounts of memory will benefit from a faster CPU, system bus and memory subsystem. From processor performance point of view, one processor per compute node. Budgetary restrictions make an SMP machine a viable option. It is possible that a compute node can take control of other functions too such as management, installation and control.

Configuration: Here we discuss the characteristics of nodes that constitute $DKA_{clusterII}$ .We also describe how the nodes are interconnected in order to behave as a single functional unit:

Hardware Configuration: The hardware configuration of the components of Dakshina cluster are as follows:

- Nodes: 40 execution nodes and all are Intel Pentium 4 (3.6 Ghz), 945 G Chipset, 1.23 GB RAM, 80 GB SATA HDD, 1 G Ethernet
- Monitor Node: exactly 1 monitor node which is Intel Pentium 4 (2.6 GHz), 865 G Chipset, 1.23 GB RAM, 80 GB PATA HDD, 100 M Ethernet
- User Login Machines: 5 different user login machines, all Intel core 2 Duo 2.6 Ghz, 1 GB RAM, 80 GB SATA HDD, 1 G ethernet, 15" TFT monitor, 52x CD ROM, 104 Std keyboard, Optical Mouse
- SGE Server: The cluster has one Sun Grid Engine server which is Intel Xeon 2.2 Ghz Quad Core x 2, 2 GB RAM, 146 GB SAS HDD x2, 1 G ethernet x2
- NFS + LDAP Server: A single such server that is Intel Xeon 2.2 Ghz Quad Core x 2, 2 GB RAM, 146 GB SAS HDD x2, 1 G ethernet x2 is an integral part of Dhakshina
- Project Server: The cluster has a project server which is Intel Pentium 4 2.6 Ghzx2, 512 MB RAM, 120 GB HDD, 1 GB and 100M ethernet
- Switches: Dhakshina uses two 24 Port Gigabits/sec switches

Connectivity: The cluster nodes in Dhakshina are interconnected using the Gigabit Ethernet LAN. The data transfer takes place at the speed of around 990 Mbps among the servers and around 665 Mbps among the client nodes. The Gigabit Ethernet uses copper cables. As we require effective connection to the already existing copper cable network of the campus, the cluster also uses UTP copper cables in the LAN. The cabling uses a category 6 Unshielded Twisted Pair (UTP) patch chord IEC 332-1 four pair, ETL verified to TIA/EIA568-B; cables from Molex.

QoS-parameters and measurement: Quality of Service of a node in a High Performance Cluster is a dynamic estimate of its performance. Various measurable parameters that are descriptive of a node's current service quality are closely monitored to work out this estimate. When a process request is handed over to the HPC, the QoS of the participant node serves as an indicator of the quality of service that can be expected out of it. As a proactive measure, this QoS could be used to choose the node for executing the process request. The various parameters used to measure node's QoS are elaborated in this section. The measurement process for each parameter is also briefly outlined. Every node in an HPC might have one or more constituent processors. The parameters monitored essentially belong to the categories, node parameter and CPU parameter.

Node parameters: These parameters are measured directly from a node. Power Usage: This is the power consumed by a node at a given point of time. It is expressed in MW and is a dynamic parameter. By dynamic parameter, we mean this parameter has to be measured every time a decision has to be made regarding the selection of next node. The power usage is measured by means of the Baseboard Management Controller (BMC) which is a specialized microcontroller embedded on the mother- board of a computer and is the intelligence component in Intelligent Platform Management Interface (IPMI), a standard Intel specification.

Memory Usage: This value is the memory of the node that is being currently used. It is also a dynamic parameter and is measured in MB. Depending upon the OS in the node, this measure can be read off an appropriate file.

Send/Receive Usage: This is yet another dynamic parameter that speaks of the current network interface rate of the node. If the node has multiple NICs, averages of the respective send/receive rates is a good enough indicator. Measurement is taken from the standard TCP/IP parameter dump.

Status: The guard measure of the node, it has to be 'ON' for the node to be considered available. Performance: This is measured in FLOPS, number of floating point operations per second. It is a static parameter and is collected using benchmarking tools (nBench for Linux).

CPU parameters: Very often it is the case that a node is functionally comprised of multiple processors. The CPU parameters are those that are measured off every individual CPU and are subsequently aggregated sum or average depending on the nature of the parameter. These parameters are:

CPU Name: The CPU name brings along the attached brand value. This qualitative measure can also be a contributor to the QoS of the node, if one is presented with relative processor ratings. CPU Temp: This measure plays a critical role in designing power aware systems. It is measured by

means of the BMC sensors. It is a dynamic measure expressed in degree centigrade. If a node has multiple CPUs, the temperature value attributed to the node is an average of the constituent CPU temperatures.

CPU Load: a dynamic parameter, that is a percentage indicator of the load of a CPU. The value is refreshed every 2 seconds to ensure that the QoS calculated is as current as possible. It is read off OS specific files. Averaging aggregates multiple CPU loads appropriately for our purpose.

CPU Speed: this is a static parameter that clearly projects the performance of a CPU. The greater its value offers better QoS of the node. An average of the speeds in case of multiple processors proves to be an unbiased measure.

CPU Type: like the CPU name, this qualitative measure can also be utilized if relative grades of the different CPU types are known beforehand. It is a static parameter.

CPU Cache: This is a static parameter for the cache memory of a CPU. In a multi-processor case, sum of the different cache values evenly carries off the reading to the QoS calculation.

CPU State: this is the CPU power state according to the Advanced Configuration and Power Interface (ACPI) specification. A state 'C_0' is an equivalent of an operating state and is mandatory for the CPU to be considered an effective part of a particular node.

In addition to the node and CPU parameters, we also keep note of the number of fans and

their respective speeds for a given node. The average fan speed can be an index of a node' s power awareness.

RESULTS AND DISCUSSION

The QoS of the nodes in a cluster can help the scheduler to take an informed decision while selecting the node to process a given request. When the QoS value is determined by taking into consideration many current and performance indicative quantities, it serves a fair ranking for each node in a cluster. In this section, our approach to QoS calculation of the nodes in $DKA_{clusterII}$ is explained in detail.

The node parameters like Power Usage, Memory Usage, Network Send/Receive Ratio, Performance and CPU parameters like Load, Temperature, Speed, Cache, CPU Type and the fan parameters. Fan Speed are measured periodically and stored into a relational data store. The node parameter' s and the CPU parameters like Cache, CPU Type (are static quantities) requires only one time measurement. All other parameters are dynamic variables and are periodically refreshed in the database. Every time the scheduler is faced with a new process request, the algorithm calculates the QoS of every available node using the most current values of these parameters to let the scheduler take an informed decision regarding the node to be chosen to process the request. To keep the approach open, flexible and extensible, the QoS is currently calculated as a weighted average. It is often the case that a single node in an HPC has multiple CPUs and multiple fans. Hence these parameters are aggregated as average or sum, as given in the proposed algorithm.

The parameters such as node performance, CPU speed and CPU cache bear a direct proportionality to their grades, that is the higher the absolute value of these parameters, the better its grade. The other parameters like power usage, memory usage, network send/receive usage, CPU load, temperature and fan speed have an inverse relationship with their grades. Subsequently, every parameter value is graded into one of the five grades: ('A','B','C','D','E') with 'A' having the highest weight and 'E' having the least weight in a range from 10-2 respectively. The grade values corresponding to these grades are shown in Table 1.

The parameters also carry a weight that had been assigned after careful analysis. Parameter weights can be modified by reallocating current job scheduling which help one to bias the algorithm towards these parameters. As the focus is on devising a power optimal mechanism, the weightages are given to power aware parameters such as CPU temperature, power usage and fan speed which are > the rest. These

Table 1: Grades and values

Grades	values
A	10
B	8
C	6
D	4
E	2

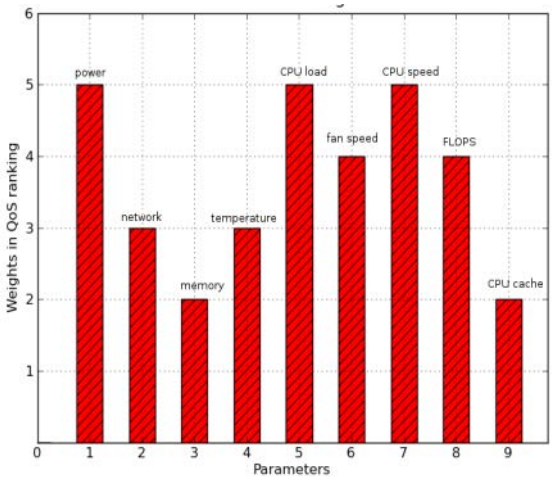

Fig. 1: Parameter weights

weights are on a scale from 1-5 and shown in the Fig. 1. Once the parameters are fetched, aggregated and graded the QoS is calculated as a weighted average. The HPC scheduler, now takes the QoS as a cue to choose the possibly best node to process the current request.

Algorithm for QoS determination:

Input : Node parameters, CPU parameters, Fan parameters of all nodes in HPC
Output: QoS of nodes in the HPC
Read designated weights for relevant parameters
Read designated grade values for the relative grades
for every node with status 'ON'
do
Read node parameters: Power Usage, Memory Usage, Network Send/Receive Usage, Performance
for every CPU of this node with ACPI status 'C_0'
do
Read CPU parameters: Load (l_{ij}), Temperature (t_{ij}), Speed(s_{ij}), Cache(c_{ij})
Calculate net node load(L_i), net node temp(T_i), net node speed (S_i), net node cache(C_i)

$$L_i = 1/n \times \sum_{j=1}^{n} j * l_{ij}$$
$$T_i = 1/n \times \sum_{j=1}^{n} j * t_{ij}$$
$$S_i = 1/n \times \sum_{j=1}^{n} j * S_{ij}$$
$$C_i = \sum_{j=1}^{n} j \times c_{ij}$$

for every fan of this node with status 'OK'
do
Read fan parameter: fan speed (f_{ij})
Calculate net node fan speed (F_i)

$$F_i = 1/n \times \sum_{j=1}^{n} j \times f_{ij}$$

Assign relative grade for each of these parameters as ['A','B','C','D','E'] for each node.
Calculate QoS of each node,
QoS_i _ (1/ ∑Parameter Weight)×(∑Grade Value× Parameter Weight)

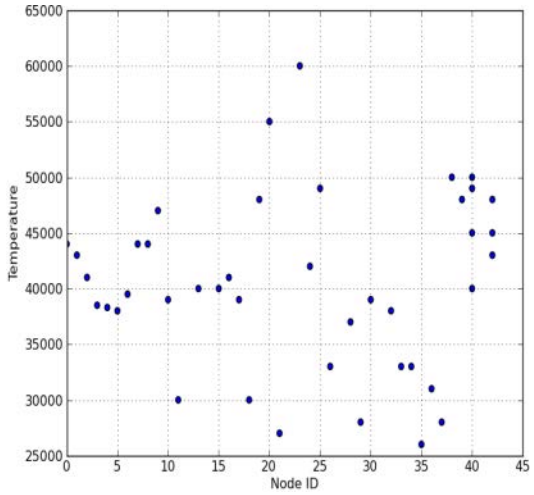

Fig. 2: Node ID temperature

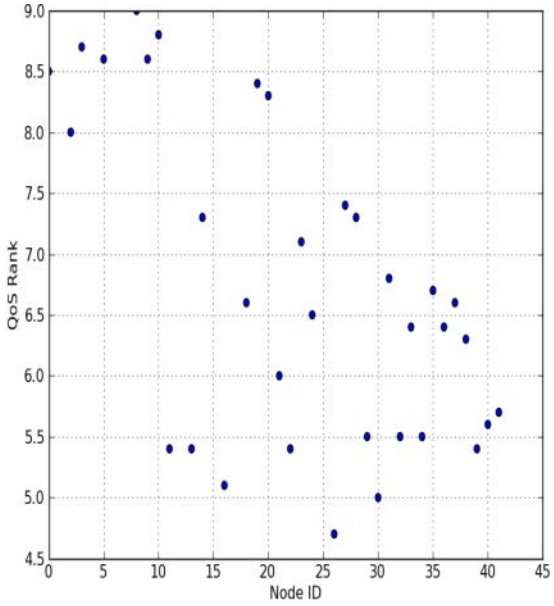

Fig. 3: Node ID vs QoS Rankp

Experimental results: The result of our experiments for QoS determination of the nodes in $DKA_{clusterII}$ is presented. The HPC consisted of 40 different nodes, all at different states and differently loaded. The QoS of the active nodes were determined at different instances of time and fed to the cluster's scheduler. The node vs net CPU temperature scatter diagram and the node vs QoS scatter diagram is shown in Fig. 2 and 3 respectively, were chosen to show the inverse relationship between CPU temperature and QoS.

For node 23, the net CPU temperature is 60 degree centigrade, the QoS is 7.18 on a scale of 10. Some nodes with relatively high temperature (Node id 20) have a high QoS (8.3) as well.

This is because, in spite of the high temperature, many other parameters are on a favorable zone for this node. The weightages for the different parameters as discussed can be revisited and improved after careful analyses and constant monitoring. Currently, the system has been biased towards power aware parameters as efficiency and performance need to be given due importance.

CONCLUSION

High Performance clusters address the growing need of processing mammoth data for extremely intricate calculation. When the scheduler in HPC is faced with the task of allotting a node(s) for processing a certain request, it greatly helps to gauge nodes on their current quality of service. Our approach thus uses QoS as a critical input for node selection. The QoS is based on various measurable parameters of the nodes in a cluster. The parameters considered span a relatively broad spectrum thus ensuring that the QoS is all inclusive. In order to help the scheduler to behave in a power aware manner, the parameters like power consumption and CPU temperature were given higher weightage. The QoS determination mechanism however is inherently flexible and can be fine-tuned to suit a different requirement for tomorrow. The algorithm can be extended by introducing more parameters to make it more efficient. Also, by keeping track of the node selection over a period of time, machine learning techniques can be employed for auto scheduling in turn eliminating the need to calculate the QoS, every time a new request is made.

Separate graphs are plotted for every parameter for its relative grading. This study supports an overall effort to achieve end-to-end QoS assurance for individual high-priority jobs in a high performance distributed cluster. This study focuses on specification and parameterization of QoS requirements for performance monitors in high performance monitors systems. The problem of scheduling workflows in terms of certain quality of service (QoS) requirements is challenging and it significantly influences performance of clusters.

REFERENCES

Abdelzaher, T.F. and N. Bhatti, 1999. Web server QoS management by adaptive content delivery. Proceedings of the 1999 Seventh International Workshop on Quality of Service, 1999, 31 May-4 Jun, 1999, IEEE, London, UK., ISBN: 0-7803-5671-3, pp: 216-225.

Bansal, N. and M. Harchol-Balter, 2001. Analysis of SRPT scheduling: Investigating unfairness. ACM. Sigmetrics Perform. Eval. Rev., 29: 279-290.

Barroso, L.A. and U. Holzle, 2007. The case for energy-proportional computing. Comput. 12: 33-37.

Chase, J.S., D.C. Anderson, P.N. Thakar, A.M. Vahdat and R.P. Doyle, 2001. Managing energy and server resources in hosting centers. ACM. SIGOPS. Operating Syst. Rev., 35: 103-116.

Eggert, L. and J. Heidemann, 1999. Application level differentiated services for web servers. World Wide Web, 2: 133-142.

Gacias, B., J. Cegarra and P. Lopez, 2012. Scheduler-oriented algorithms to improve human-machine cooperation in transportation scheduling support systems. Eng. Appl. Artif. Intell., 25: 801-813.

Huang, S. and W. Feng, 2009. Energy-efficient cluster computing via accurate workload characterization. Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, May 18-21, 2009, IEEE Computer Society, Washington, USA., ISBN: 978-0-7695-3622-4, pp: 68-75.

Jacobs, I.S. and C.P. Bean, 1963. Fine Particles, Thin Films and Exchange Anisotropy: Effects of Finite Dimensions and Interfaces on the Basic Properties of Ferromagnets. Research Information Section, Knolls, USA., Pages: 73.

Katz, M.J., P.M. Papadopoulos and G. Bruno, 2002. Leveraging standard core technologies to programmatically build linux cluster appliances. Proceedings of the 2002 IEEE International Conference on Cluster Computing, September 23-26, 2002, IEEE, Chicago, Illinois, ISBN: 9780769517452, pp: 47-53.

Moore, J.D., J.S. Chase, P. Ranganathan and R.K. Sharma, 2005. Making scheduling cool: Temperature-aware workload placement in data centers. Proceedings of the Annual Conference on USENIX Annual Technical Conference General Track, April 13, 2005, USENIX Association, Berkeley, USA., pp: 61-75.

Norris, B., J. Ray, R. Armstrong, L.C. McInnes and D.E. Bernholdt *et al*., 2004. Computational quality of service for scientific components. In: Component-Based Software Engineering. Crnkovic, I., J.A. Stafford, H.W. Schmidt and K. Wallnau (Eds.). Springer Berlin Heidelberg, Berlin Heidelberg, Germany, ISBN: 978-3-540-21998-9, pp: 264-271.

Sfrent, A. and F. Pop, 2015. Asymptotic scheduling for many task computing in big data platforms. Inf. Sci., 319: 71-91.

Shan, Z., C. Lin, D.C. Marinescu and Y. Yang, 2002. Modeling and performance analysis of QoS-aware load balancing of web-server clusters. Comput. Netw., 40: 235-256.

Smutnicki, C., J. Pempera, J. Rudy and D. Zelazny, 2015. A new approach for multi-criteria scheduling. Comput. Ind. Eng., 90: 212-220.