MŰEGYETEM 1782

Evaluation on Performance and Energy Efficiency of Distributed Computing Systems

Ph.D. Dissertation Summary

by

Tran Thi Xuan (MSc)

Supervised by

Prof. Do Van Tien (DSc)

Department of Networked Systems and Services

Budapest University of Technology and Economics

Hungary, 2020

1 Introduction

Distributed computing plays a key role in solving computational scientific problems as well as processing large volumes (terabytes) of data [1, 2, 3, 4, 5]. With the rapidly increasing scale of such systems, minimizing operation energy consumption has become more crucial than ever. Various approaches have been adopted to reduce energy consumption in IT industry.

Dynamic power management (DPM) practices are switching off technique and Dynamic Voltage and Frequency Scaling (DVFS) [6, 7, 8]. Energy optimization problem with the switching off idle servers has received considerable interest since the energy consumed in the idle state contributes a non-negligible fraction in the total energy cost [10, 11, 12, 13]. DVFS has been commonly used to reduce the power consumption of processors by scaling up/down the operating voltage and frequency of CPU. This technique, however, comes with degradation in system performance and requires a careful consideration before applied [15, 20].

Numerous researches focused on energy-efficient scheduling policies with the aim to gain the efficiency of DPM practices in computing systems [9, 13, 19, 20, 22]. Due to the continuous change of hardware design as well as the system infrastructure, the application of DPM remains an open research question. In addition, several studies pointed out that resource heterogeneity (i.e. computing processors have different power characteristics and performance capacities) can be taken into scheduling decisions to achieve energy conservation [14, 24, 25, 26, 28]. Those studies, however, lack of full consideration for the heterogeneity of resource characteristics and a solution to balance the system performance and energy efficiency. Furthermore, the resource heterogeneity and DPM application have not been investigated adequately in big data processing systems.

This study was motivated by the need to take into account different resource characteristics and DPM techniques. Both compute-intensive and big-data processing job schedulers have been investigated.

The rest of the booklet is organized as follows. The objectives with respect to specific problems are described in Section 2. The methodology used for performance evaluation is presented in Section 3. Section 4 briefly summarizes the achieved results. Finally, applications of results are given in Section 5.

2 Objectives

The thesis objectives were

1. creating new algorithms for scheduling jobs to computational clusters built from heterogeneous types of machines, taking aspects of performance, energy, and the varying arrival rate of customers;

2. proposing an energy-efficient resource allocation policy for parallel jobs processing in clusters of multicore processors; and

3. developing a new scheme for Hadoop Distributed FileSystem (HDFS) data layout to cope with the energy inefficiency when MapReduce workloads are executed by a Hadoop YARN cluster.

3 Methodology

The purpose of the thesis was to develop new scheduling algorithms for computational clusters and a new data layout scheme for HDFS. Therefore, a practical methodology was applied to reach the aims,

- different scenarios were established to compare existing algorithms with new ones,
- discrete-event simulation [31] models were created for the scenarios, that is, a simulation was used to evaluate and compare the developed algorithms. This is a common practice in the engineering world. Moreover, the batch means method [31, 30] was used to compute the confidence interval of the performance measures in scenarios,
- workloads that were captured in the real environment were used to drive the simulation experiments in some cases.

4 New results

4.1 Multi-objective scheduling in heterogeneous computational clusters [J1, J2]

Thesis Group I. [J1, J2] I proposed a set of scheduling algorithms for a computational cluster of heterogeneous servers, taking various aspects of energy, performance, and the varying arrival intensity into account. I provided a guideline on selecting an appropriate scheduling policy regarding the desired goal of the system operator.

This thesis addressed the scheduling issue in computational clusters of heterogeneous machines that serve jobs with varying arrival intensity. Aspects to be considered are the performance, energy consumption, and the varying arrival intensity (note that a lot of works only considered the fixed arrival rate). The problem is that there is no single solution that can be optimal for all aspects. This work was devoted to heuristics for multi-objective scheduling problem. Details of the thesis are presented in Chapters *2* and *3* of the dissertation.

Scheduling a job to a server can be done in various ways. A server can be selected randomly, or according to a specific characteristic (e.g. performance capacity, power consumption) , or based on the both server characteristics and customer-driven aspects (e.g. the arrival intensity at the current time).

Given a computing cluster of K different server types/classes; class i ($i = 1, \ldots, K$) contains $M(i)$ homogeneous machines. Let a server of class i be characterized by parameters[1] of the performance capacity[2] (C_i), the average active power ($P_{ac,i}$), the power consumed in the idle

[1]these parameters are measured according to *SPECpower_ssj2008* benchmark, website: http://spec.org/power_ssj2008

[2] the `ssj_ops` value defined by the number of operations finished during the measurement interval divided by the number of seconds defined for this interval

state ($P_{id,i}$), and the performance to power ratio ($C_i/P_{ac,i}$) that denotes the server's energy efficiency.

In a heterogeneous cluster, it is a requisite for scheduling solutions to take the resource characteristics into account. The scheduler may give priority to servers with (1) high performance capacity, (2) low performance capacity, (3) low power consumption, or (4) high energy efficiency. To achieve high performance, resource allocation according to the high performance capacity priority arises as an effective approach. To conserve energy, the high energy-efficiency priority based allocation appears a promising approach. I provided ranking methods to organize computing servers and applied them in two aforementioned scheduling policies (*Thesis 1.1*). The given approaches focus on only one aspect of either energy efficiency or performance. To consider the varying arrival intensity, I came up two new scheduling algorithms that use real-time statistics of workloads (*Thesis 1.2*). I offered a suggestion on which algorithm is proper for a particular goal of system operators (*Thesis 1.3*).

Thesis 1.1. *[J1] I provided two functions to rank servers according to their performance capacity and energy efficiency characteristic. I proposed High-Performance policy and Energy-Efficiency policy that take account of the performance rankings and energy efficiency rankings, respectively. I showed that the proposed policies outperform other solutions that use different server orderings or schedule jobs randomly. Mainly, High-Performance policy performs best and Energy-Efficiency policy consumes the least energy among alternatives.*

I defined two functions to calculate the High-Performance (HP) and Energy-Efficiency (EE) ranking of servers as:

$$r_p(s) = \frac{C_s}{\max\limits_{i \in S} C_i}, \quad s \in S, \tag{1}$$

and

$$r_e(s) = \frac{\frac{C_s}{P_{ac,s}}}{\max\limits_{i \in S} \frac{C_i}{P_{ac,i}}} \quad s \in S \tag{2}$$

, respectively.

Applying these ranking calculations, I organized physical servers in the HP priority ordered set S_p and the EE priority ordered set S_e. Let servers be indexed by (i,j) ($i = 1, \dots, K$, $j = 1, \dots, M(i)$).

- In S_e, server (1,j) (j = 1, ..., M(1)) has the highest EE ranking, server (K,j) (j = 1, ..., M(K)) has the lowest EE ranking.

- In S_p similarly, server (1,j) (j = 1, ..., M(1)) has the highest HP ranking, server (K,j) (j = 1, ..., M(K)) has the lowest HP ranking.

**** Energy-Energy ranking based scheduling (EE policy)***

The EE policy picks up a free server with the highest EE ranking by referring the EE ranking ordered set S_e. In case all servers are busy, the job must wait in a shared buffer for a free server.

**** High-Performance ranking based scheduling (HP policy)***

Similarly, the HP policy prioritizes the highest HP ranking servers by referring the S_p ordered set. The first free server found will be the one with highest HP priority and selected. If all servers are busy, the job waits in the buffer for a free server.

After choosing an ordering set (S_e or S_p), scheduling procedure (of EE policy or HP policy) is given in **Algorithm 1**. If it is not stated otherwise, the cluster provides computation service for compute-intensive jobs. A job is to be served by a physical machine after a scheduling decision. There is a common buffer to store waiting jobs.

Algorithm 1 SCHEDULE

```
for i = 1 → K do                                  ▷ Server classes are in a priority order
    for j = 1 → M(i) do
        if $(i, j)$ is FREE then                    ▷ Find a free server in class $i$
            $free\_server \leftarrow (i, j)$
            GOTO SCHEDULE
        end if
    end for
end for
SCHEDULE:
if found $free\_server$ then
    ASSIGN job to $free\_server$
else
    RETAIN job in Common Queue
end if
```

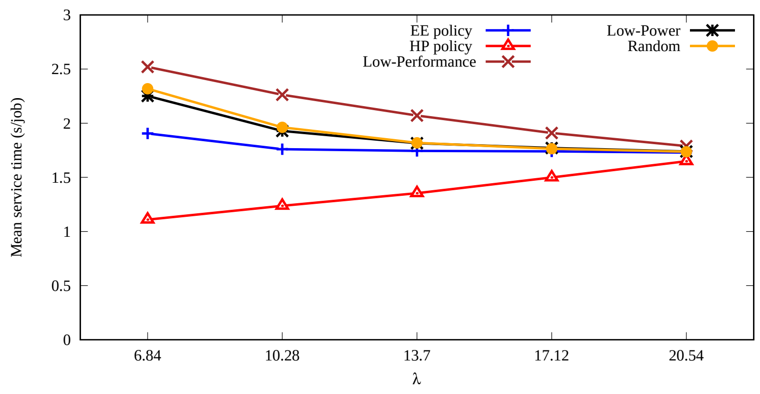
Figure 1 presents the average execution time and the average energy consumption with two proposed policies, HP and EE, and other scheduling options of random allocation, low-power, and low-performance based allocation.

It is shown that the HP and EE policies outperform others, wherein HP policy yields the best performance. EE policy consumes the lowest energy. Low-power based scheduling yields the second lowest energy consumption and performs worse than EE policy. Random and low-performance allocation methods also consume more energy than the two proposals EE and HP policy.

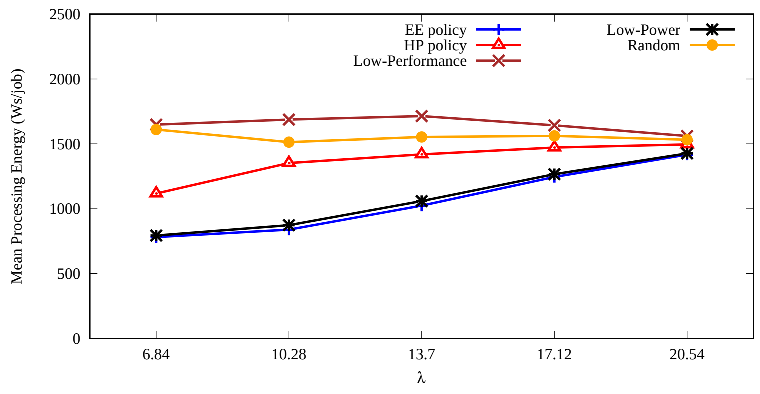
Thesis 1.2. *[J2] I proposed two algorithms (named the Real-Load 2 and the AVR_ST procedure 3), which make scheduling decisions based on the real-time statistics of workloads and servers' characteristics. I showed that the Real-Load (RL) algorithm achieves a tradeoff between energy consumption and service time, while AVR_ST algorithm results in fairly stable service time and energy consumption regardless of the varying arrival intensity.*

**** Real-Load algorithm***

The Real-Load algorithm attempts to make a scheduling decision adapting to the variation of the intensity in real-time. If the "instantaneous" load is low, EE priority servers are preferred to save energy; when the "instantaneous" load becomes high, HP priority servers should be selected to speed up the service.



(a) Average execution time



(b) Average energy consumption (with switching-off)

Figure 1: *System metrics*

To realize the load variation, I introduced real-time statistics in terms of Quasi-instantaneous load and Average Load as follows.

Definition 1: Quasi-instantaneous load

Let TW be the length of time windows in the operation duration and w denote the window at the current instant. Let $a(w)$ be the number of arrivals and $\lambda(w) = \frac{a(w)}{TW}$ be the quasi-instantaneous arrival rate of the time window w. The quasi-instantaneous load $u(w)$ is defined as the average load of current time window w and calculated as

$$u(w) = \frac{\lambda(w)}{\mu} \tag{3}$$

, where μ is the service rate of the system.

Definition 2: Average load

Let $A(t)$ be the accumulative number of jobs arriving system until the current instant t. The arrival rate at time t is $\lambda = A(t)/t$. The average load of system at time t is defined as the ratio of the arrival rate at t over the system service rate.

$$\rho = \frac{\lambda}{\mu} \tag{4}$$

Real-Load (RL) algorithm (**Algorithm 2**) compares the real-time quasi-instantaneous load ($u(w)$) to the average load (ρ) at the arrival time of *job*. If $u(w) \leq \rho$, the scheduler refers to the EE ranking ordered set S_e. Otherwise, HP ranking order S_p is selected.

Algorithm 2 Real-Load Procedure

```
CALCULATE real-time Average load $\rho$ and Quasi-instantaneous load $u(w)$
if $u(w) \leq \rho$ then
    CHOOSE EE ranking order
else
    CHOOSE HP ranking order
end if
GOTO Algorithm 1
```

**** Average service time (AVR_ST) based algorithm***

The AVR_ST algorithm approaches the problem by attempting a fair job-server assignment such that jobs with short service time demand should go to EE machines, while ones with long service time demand should get powerful HP servers. In case the job service demand is unknown, the scheduler uses statistical measures for its estimation. I defined two statistics of Average service time and Weighted mean service time used for the estimation in what follows.

Definition 3: Average service time

Let $n(t)$ be the number of completed jobs up to current time instant t. Let s_l be the service time of job l, $1 \leq l \leq n(t)$. Let $\theta(t)$ be the sum of service times of $n(t)$ jobs up to time t, then $\theta(t) = \sum_{l=1}^{n(t)} s_l$. The overall average service time $ST(t)$ per job up to time t is

$$ST(t) = \frac{\theta(t)}{n(t)} \tag{5}$$

Definition 4: Weighted mean service time

Let $n_i(t)$ and $\theta_i(t)$ be the number of completed jobs and the total processing time of servers of server class i ($i = 1, \ldots, K$) until time t. The average service time of jobs served in class i up to time t is $st_i(t) = \theta_i(t)/n_i(t)$. The weighted mean service time at the instant t ($mean\_ST(t)$) is defined as

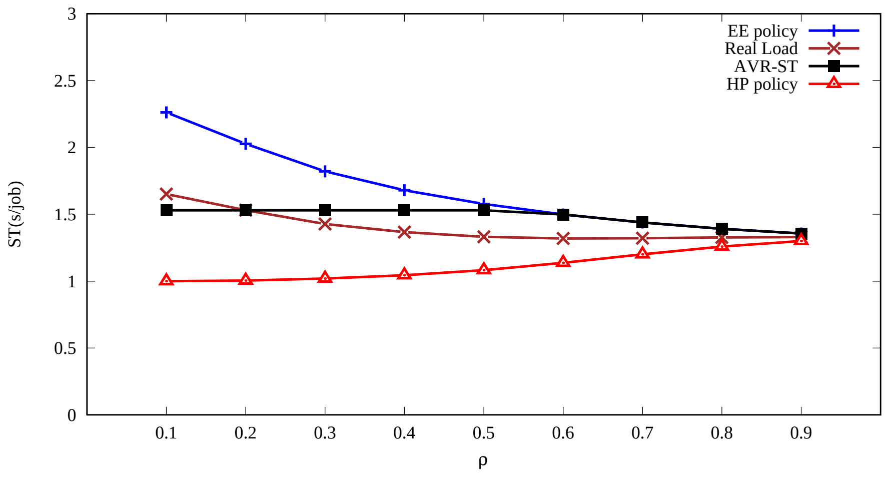
$$mean\_ST(t) = \frac{\sum_{i=1}^{K}(M(i) \times C_i \times st_i(t))}{\sum_{i=1}^{K} M(i) \times C_i} \tag{6}$$

, where the ratio of the service capacity of class i over the total system service capacity, $\frac{M(i) \times C_i}{\sum_{i=1}^{K} M(i) \times C_i}$ ($i = 1, ..., K$), is the weight associated to the class.

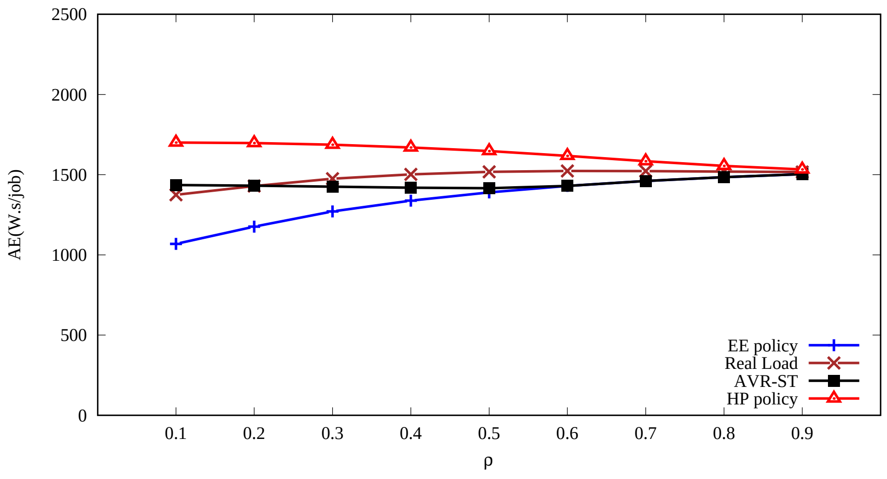
Average service time (AVR_ST) algorithm (**Algorithm 3**) refers to the EE ranking set if the average service time $ST(t)$ is shorter than the weighted mean service time $mean\_ST(t)$. Otherwise, HP priority servers are preferred. .

Algorithm 3 AVR_ST Algorithm

```
CALCULATE the Average service time $ST(t)$ and Weighted mean service time $mean\_ST(t)$
if $ST(t) \leq mean\_ST(t)$ then
    CHOOSE EE ranking order
else
    CHOOSE HP ranking order
end if
GOTO Algorithm 1
```



(a) Average execution time



(b) Average energy consumption (with switching off)

Figure 2: *System metrics*

The numerical results obtained by EE policy, HP policy, Real-Load procedure, and AVR_ST algorithm are shown in Figure 2.
With Real-Load policy, the average energy consumption and the average service time are average compared to those achieved by EE and HP policy in the full range of the offered loads. Especially, the average energy consumption and service time do not have noticeable change regarding the load variation and remains almost constant (≈1516 Ws and 1.33 s, respectively) in a large range of loads (0.4-0.9).
The AVR_ST solution also achieves the middle values of service time and energy consumption at low to medium load (0.1 to 0.5), compared to those obtained by EE and HP policy. At higher load, it performs as good as EE policy. With AVR_ST, system performance and energy trivially change when the load varies.

Thesis 1.3. *[J1, J2] I suggested that*

- *EE policy is the best solution if energy conservation is the superior aspect;*
- *HP policy is the choice when the customer's service requirement comes first;*
- *When operators want to balance between energy consumption and service time in the full range of the offered loads, the Real-Load algorithm is the winner;*
- *When operators expect system performance to be independent of the varying intensity, AVR_ST can be selected.*

Given the numerical results achieved by four proposed algorithms (Figure 2), we can see that each solution has an advantage and a drawback.
EE policy provides the lowest, but increasing average energy consumption (ranging from 1068 to 1503 Ws) with the expense of long execution time (decreasing from 2.26 to 1.38 s) in the full range of loads. Clearly, EE policy is an appropriate solution when energy savings is the most crucial aspect.
HP policy attains minimum processing time (1.0 - 1.30 s) but causes high energy (decreasing from 1700-1532 Ws/job). Only high performance aspect is held by this policy.
EE policy and HP policy have varying degrees of success with the change in job arrival rate. Contrastingly, Real-Load and the AVR_ST solutions do not depend on the load variation during operation and provide medium performance and energy consumption.
Therefore, this thesis claim is a guideline on algorithm selection that can be useful for a system operator.

4.2 Energy-efficient scheduling in clusters of multicore processors [J3],[C2],[C4]

Thesis Group II. [J3],[C2],[C4] I proposed an energy-efficient scheduling solution for computational clusters of multicore processors. I showed that the proposed policy consumes the least energy among alternatives that are driven by dynamic power management, while it reduces energy by over 50% and response time by 18% in comparison to a random allocation.

The evolution of multicore technology and its multiprocessing capability makes the application of DPM practices questionable. This work was motivated by a need to investigate the impact of DPM and resource heterogeneity in a cluster of multicore machines and then propose an energy-efficient scheduling solution for serial and parallel jobs processing. The details of this thesis are given in Chapter *4* of the dissertation.

4.2.1 ***The impact of DPM and resource heterogeneity in multicore processors***

In the previous thesis, I addressed that server performance capacity and energy efficiency characteristics can be taken in scheduling jobs to a resource-heterogeneous cluster. However, the investigation did not consider the multiprocessing capacity of computing machines. In this thesis, the effectiveness of the approach was examined in multiprocessing scenarios, wherein the two ranking ordered sets S_e and S_p given in *Thesis Group I* were used.

In addition, a question about the DPM in cores of processors was investigated. I should note that switching off individual cores is not possible in general-purpose processors used in contemporary data centers. This thesis was to find out whether the DPM in cores could be a candidate solution for energy conservation. Moreover, switching off idle servers has been used widely in data centers. The technique has been applied in all investigations throughout my research works. In fact, there are costs of latency and power consumption to bring a sleeping server back the active state; if a server stays Off in a period shorter than the exit latency, energy can not be saved. Therefore, *I carried out a switching policy to avoid short sleep durations of servers.*

The following job assignment scenarios are to cover possible ways of employing DPM in servers and cores of processors.

- *Single-job* scenario assumes a job is executed by a physical server at the full load capacity. Switching off technique is applied for idle servers.
- *Full-active* follows assumptions that a job is processed by a physical core (hence multiple jobs can run concurrently on a machine) and unused cores in the machine stay idle. Switching off technique is applied for idle servers.
- *Dynamic-active* assumes that a job is processed by a physical core. Yet idle servers and unused cores can be switched off dynamically.

Note that to gain the effectiveness of switching off technique applied to idle physical servers, the number of active servers should be as small as possible. Therefore, in two multiprocessing scenarios (Full-active and Dynamic-active), a free server with the least unused cores is preferred if there are many free servers in the same class.

Given two ways of server orderings and three scenarios, six policies of Single-job EE, Single-job HP, Full-active EE, Full-active HP, Dynamic-active EE, and Dynamic-active HP were evaluated. Algorithm 4 presents the scheduling procedure for multiprocessing scenarios.

Algorithm 4 Scheduling algorithm for multiprocessing

```
for $i = 1 \to K$ do                                ▷ Server classes are in the ordering $S_e$ or $S_p$
    for $j = 1 \to M(i)$ do
        if server $(i, j)$ has LEAST free_cores and is FREE then
            $free\_server \leftarrow (i, j)$
            GOTO ALLOCATE
        end if
    end for
end for
ALLOCATE:
if found $free\_server$ then
    ROUTE job to $free\_server$
else
    RETAIN job to Common Queue
end if
```

Thesis 2.1. *[J3], [C4] I suggested dynamic power management (DPM) in cores of processors can be a candidate for energy conservation in multicore clusters. I demonstrated that applying DPM in cores (Dynamic-active scenario) can reduce ≈30% energy in comparison to Full-active, regardless of server orderings.*

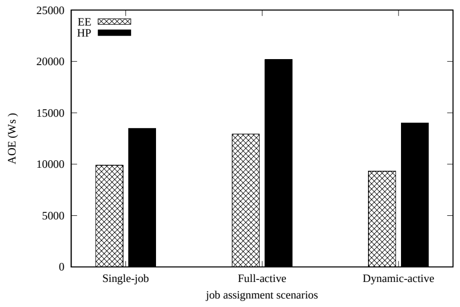
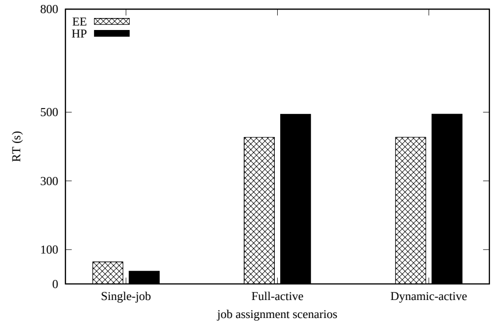


Figure 3: *Average Energy consumption*

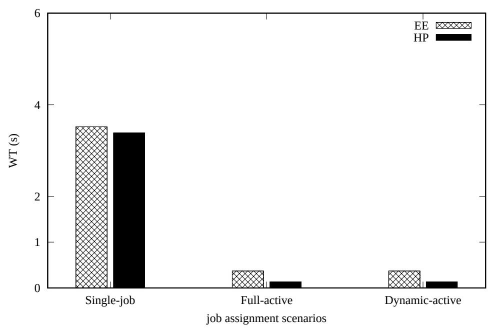
Figure 3 plots the average energy consumption regarding different scenarios. It shows that the energy can be saved if a server processes a job with the full performance capacity (Single-job). In case of multiprocessing, energy consumption can be reduced if the DPM is applied to cores of processors (in Dynamic-active). Dynamic-active EE consumes the average energy of ≈ 9.28 KWs for a job's execution, while it is 13.5 KWs with Full-active EE. Dynamic-active HP results in the average energy consumption of 14.4 KWs, reducing $\approx 30\%$ compared to Full-active HP (20.3 KWs).

Thesis 2.2. *[J3], [C4] I showed that*

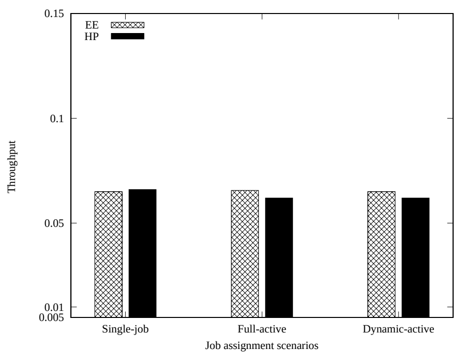
- *the server energy-efficiency characteristic is an important aspect to reduce energy consumption in multiprocessing machines, and*
- *the server performance capacity characteristic is inconsiderate when jobs are processed by cores of processors. Therefore, I suggested the scheduler should take account of the processor features to improve the performance.*



(a) Average respone time



(b) Average waiting time



(c) Throughput

Figure 4: *System performance measures vs. operation scenarios*

As shown in Figure 3, all EE based scenarios reduce energy consumption compared to corresponding HP based scenarios. Specifically, Dynamic-active EE reduces energy consumption by 35% compared to Dynamic-active HP, and Full-active EE saves $\approx 35\%$ energy compared to Full-active HP. Moreover, Dynamic-active EE consumes the least energy among alternatives. Figure 4 plots the performance measures in terms of average response time, average waiting time, and system throughput. Figure 4a shows that the Full-active HP and Dynamic-active HP perform worse than Full-active EE and Dynamic-active EE. In other words, ordering servers based on their performance capacity characteristic is not considerate to achieve high performance in multiprocessing scenarios. In case the system performance is the desired goal, the scheduler must consider the processor features and find a new factor defining the performance priority of servers. Though the average response time is higher in the cases of Dynamic-active and Full-active in comparison to Single-job, we observed an equivalent throughput at ≈ 0.06 job/s (see Fig. 4c). Moreover, jobs have to wait much longer in Single-job scenario (as showed in Figure 4b).

4.2.2 ***The proposed energy-efficient scheduling algorithm***

Thesis 2.3. *[C2] I developed an energy-efficient scheduling algorithm for parallel jobs processing in multicore computing clusters. I indicated that the proposed solution obtains energy reduction by over 50% and performance improvement by 18% compared with a random allocation procedure.*

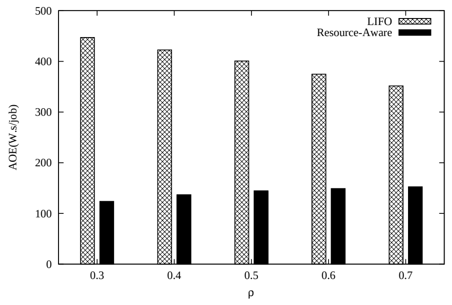
Theses 2.1 and 2.2 exposed that the aspects of the server energy-efficiency characteristic and DPM in cores of a processor play important roles for energy savings in resource-heterogeneous, multicore computing clusters. This thesis carried out an energy-efficient scheduling solution for parallel job processing, taking the revealed aspects into account.

Assuming that a job is characterized by (j, TS_j), where j is job identification and TS_j is the set of tasks, of which each requires a computing unit of one CPU core. Tasks of a job are assumed to arrive and depart together. Hence job j needs $|TS_j|$ cores simultaneously for its execution. The Resource-aware (RA) scheduling policy (Algorithm 5) works as follows. When a job arrives, the scheduler selects free servers according to their EE priority and the number of available cores. Server classes are ordered with EE priority. In the same class, a free server with the least free cores is preferred. The allocation procedure is fulfilled if there are sufficient resources to run all tasks of the job. Otherwise, job must wait in a buffer and be allocated later. An idle server as well as an unused core are to be switched off dynamically.

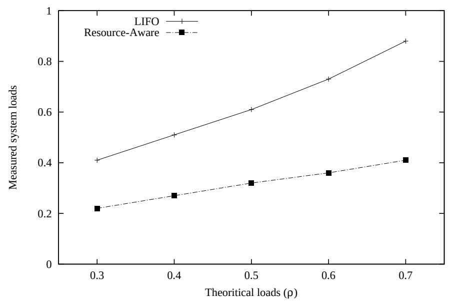
Figure 5 plots measured system metrics (averages of consumed energy, load, waiting time, and response time) regarding the RA algorithm and LIFO allocation policy (a best-effort random allocation in which free servers are listed in a Last-In First-Out stack so that the most recently added server is picked first).
The average measured load (calculated by the ratio of average busy time of servers over the operating time) falls in the range from 0.2 to 0.4 with RA algorith. In contrast, it is ranged from 0.4 to 0.8 with LIFO solution. Consequently, the DPM technique can be applied more efficiently when RA algorithm is used. Numerical results show the proposed algorithm conserves energy consumption of $\approx$ 55% - 68%, compared with LIFO policy. Interestingly, the proposal also improves the system performance with 18% reduction in the average response time.

Algorithm 5 Resource-aware (RA) SCHEDULE((j, TS_j))

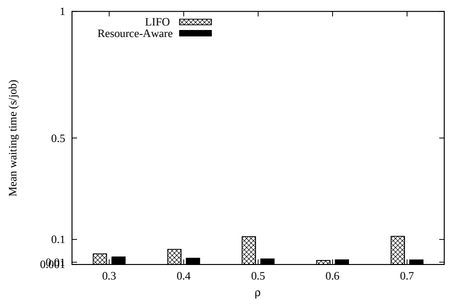
```
$R \leftarrow \emptyset$                                        ▷ Initiate an empty list of allocated servers
$cores\_sum \leftarrow 0$                                ▷ Initiate the sum of free cores of servers in $R$
$task\_number \leftarrow size(TS_j)$
for $i = 1 \rightarrow K$ do                                 ▷ Classes are in the EE priority order
    for $m = 1 \rightarrow M(i)$ do                        ▷ Servers are in the order of the least free cores
        if server $(i, m)$ is FREE then
            INSERT server $(i, m)$ into $R$
            ADD $free\_cores$ of server $(i, m)$ to $cores\_sum$
        end if
        if $cores\_sum \geq task\_number$ then                   ▷ Found sufficient resources
            GOTO ALLOCATE
        end if
    end for
end for
ALLOCATE:
if $cores\_sum < task\_number$ then
    RETAIN job $(j, TS_j)$ in the queue
else
    for each server $s$ in $R$ do
        do
            ASSIGN the first task $t$ in $TS_j$ to server $s$
            DECREMENT $free\_cores$ of $s$
            DELETE the first task $t$ from $TS_j$
        while server $s$ is FREE and task set $TS_j$ is NOT EMPTY
    end for
end if
```



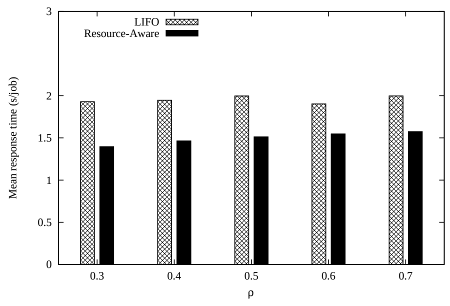
(a) Average energy consumption



(b) Measured load



(c) Average waiting time



(d) Average respone time

Figure 5: *System metrics vs. load*

4.3 A novel scheme of data layout for HDFS [J4]

Thesis Group III. [J4] I devised a new data layout scheme for Hadoop Distributed File System to gain the effectiveness of dynamic power management applied in Hadoop based clusters. Compared with the existing layout, I showed that the proposed scheme leads to a reduction of over 50% in the average energy consumption and an increase of 6% in the average computation time of MapReduce jobs.

In the two previous theses, I showed that taking account of resource heterogeneity and DPM technique can be beneficial to energy efficiency and performance of computational clusters. The approach, however, can not be directly applied to a big data processing system. In big data processing, system performance and energy are strongly dependent on both computing and storage facilities. This thesis made an effort to coordinate resource characteristics and DPM aspects in a typical big data processing framework - Hadoop [38]. The detailed description of the approach is presented in Chapter *5* of the dissertation.

In an ordinary Hadoop system (see Figure 6), Hadoop YARN (Yet Another Resource Negotiator) [35] manages and allocates resources for big data applications (e.g. Mapreduce) that access and compute big data stored in the HDFS (Hadoop Distributed File System). The HDFS components (NameNode and DataNodes) and YARN elements (ResourceManager and NodeManagers) are collocated in a cluster of servers. We supposed the NameNode (HDFS master) and ResourceManager (YARN master) are hosted in a dedicated node, while every worker node hosts DataNode and NodeManager. DataNodes hold datasets in the form of data blocks, while NodeManager spawns computing containers. A YARN container runs either an ApplicationMaster (AM)-an application-specific process that makes resource requests and manages all aspects of the job) or a job task.

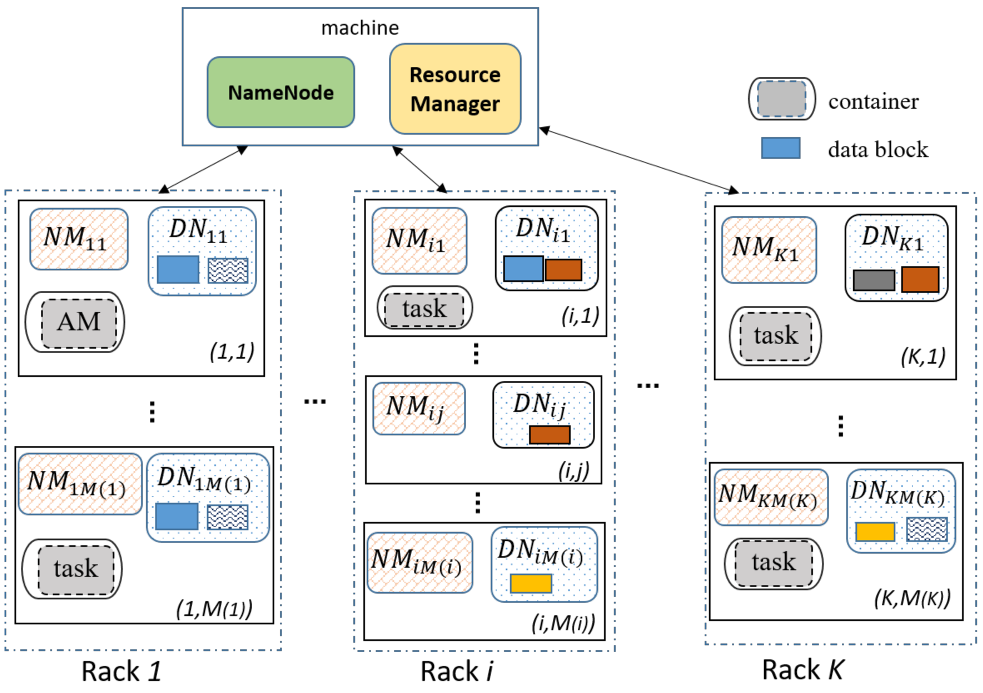


Figure 6: *Collocation of HDFS and YARN components in a computational cluster*

In a nutshell, the resource allocation procedure implemented by YARN leans towards the data locality optimization (i.e., it keeps the computation as close to the data as possible). Furthermore, neither resource heterogeneity nor DPM technique has been taken into account by the framework. Since the root cause of the energy inefficiency lies in data layout scheme [36], this work was devoted to a new data layout scheme that supports efficient resource utilization and effective application of DPM technique. Aspects to be considered are the resource heterogeneity, data heterogeneity, and DPM practice.

Regarding to different types of physical machines, I calculated the computing rankings and efficiency rankings of server racks (See details in Section 5.4, Chapter 5 of the dissertation). Using those rankings, I divided the cluster into three subsets of racks, including

- High-Compute subset SS_c containing $\lfloor \frac{K}{3} \rfloor$ racks that have the highest computing rankings;
- Energy-Efficient subset SS_e having $\lfloor \frac{K-|SS_c|}{2} \rfloor$ racks that do not belong to SS_c and are with the highest efficiency rankings;
- the saving subset SS_s formed by the remainder racks.

The proposed data layout algorithm (with the replication factor $RF = 3$ by default) works as follows. When a write request for data d is submitted, the data placement algorithm (Algorithm 6) calculates block number B_d based on given information (e.g. type, size, block size, etc...), compares the block number of input data B_d with the average number of data blocks B_{avg} and selects

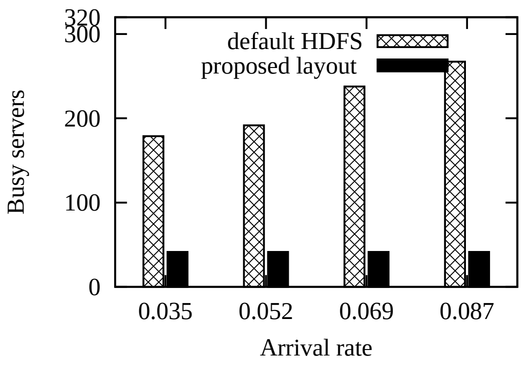
- a DataNode in *High-Compute* subset SS_c for storing all the first replicas of input file if $B_d \geq B_{avg}$; or a DataNode *Energy-Efficient* subset SS_e otherwise, and
- two random DataNodes on the same remote rack for the two other replicas of a block.

Nodes in the saving subset SS_s are never selected for storing the first replicas of data. However, they still participate in the data placement if the NameNode picks them randomly for a replication.

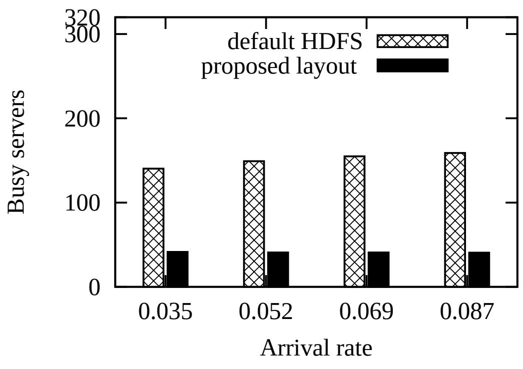
Algorithm 6 A new HDFS layout procedure

```
Output: List of DataNodes $\{L_{b,r}; r \in [1, RF]\ \}$ for block $b$
if $B_d \geq B_{avg}$ then
    $i \leftarrow$ a random number in $[1, \ldots, |SS_c|]$
else
    $i \leftarrow$ a random number in $[|SS_c| + 1, \ldots, |SS_c| + |SS_e|]$
end if
$j \leftarrow$ random number in $[1, ..., M(i)]$
for $b \in B_d$ do
    $L_{b,1} \leftarrow DN_{ij}$                          ▷ the first replicas on chosen $DN_{ij}$
    $k \leftarrow$ a random number in $\{1,...,K\} \backslash i$
    $L_{b,2} \leftarrow DN_{kl}$ ; $l$ is a random number in $\{1, ..., M(k)\}$
    $L_{b,3} \leftarrow DN_{km}$ ; $m$ is a random number in $\{1, ..., M(k)\} \backslash l$
    ESTABLISH data pipe and WRITE block $b$ to selected DataNodes
end for
```

Achieved results regarding the average number of busy servers, average energy consumption per job, and mean response time per job are plotted in Figures 7-9. With the existing layout, the number of busy servers is high and varying with a small change of the workload intensity. On the other hand, this metric remains unchanged with the proposed scheme. In other words, my solution improves resource efficiency and gains the use of switching off technique. As a result, my proposal achieves a dramatic reduction in energy consumption (by over 50%) with a small increase in response time (by $\approx$ 6%) in comparison with the existing layout.

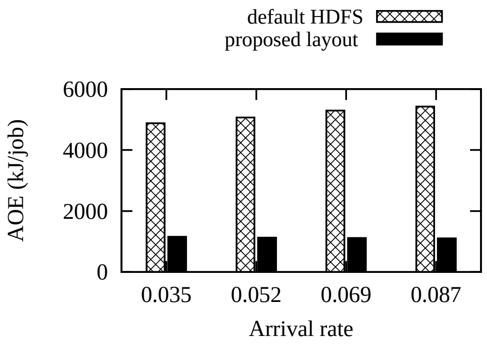


(a) Uniform Dist.

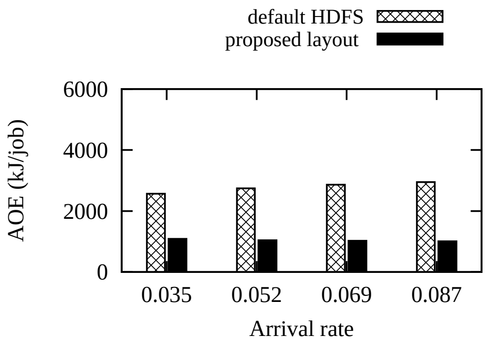


(b) Normal Dist.

Figure 7: *Average number of busy servers*

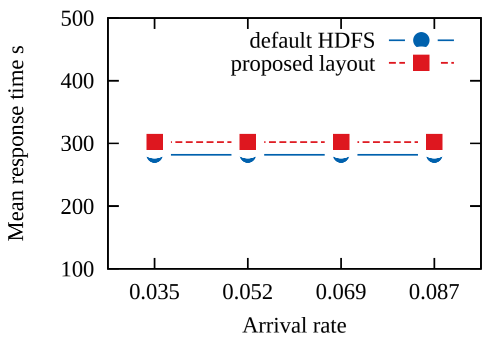


(a) Uniform Dist.

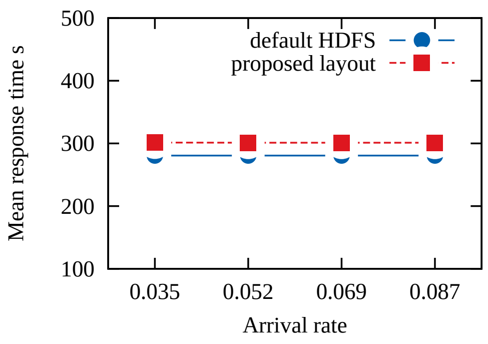


(b) Normal Dist.

Figure 8: *Average energy consumption per job*



(a) Uniform Dist.



(b) Normal Dist.

Figure 9: *Mean response time per job*

5 Applications of results

The first achieved result provides a set of scheduling heuristics applicable for computational clusters of heterogeneous machines and a guideline that helps a system operator to select an appropriate scheduling algorithm from a specific perspective.

The second result group points out the possibility of energy savings in computing systems if dynamic power management (DPM) could be applied in cores of processors. It suggests a prospective trend of future processor design and manufacture to enable switching on/off cores dynamically and individually. Resource-aware algorithm in this group can be used straightforwardly in the field of efficient resource management and allocation for multicore systems.

The third group provides the insight of how a big data job scheduler-Hadoop YARN allocates resources among tasks and how a data layout scheme affects on system performance in terms of energy consumption and service time. The proposed scheme is applicable in any context of processing datasets that are stored in Hadoop Distributed File System (HDFS). Moreover, the proposal of taking account of compute resources and data layout scheme should be considered in operating any big data processing system to attain energy efficiency.

Own publications

Journal Papers

[J1] Tien V. Do, Binh T. Vu, **Xuan T. Tran**, and Anh P. Nguyen. A generalized model for investigating scheduling schemes in computational clusters. *Simulation Modelling Practice and Theory*, 37(0):30–42, 2013. (*Impact Factor = 2.426*)*.

[J2] **Xuan, T. T.** and Tien, V. D. and Binh, T. V. New algorithms for balancing an energy consumption and performance in computational clusters. *Journal of Computing and Informatics*, 36(2):307–330, 2017. (*Impact Factor = 0.524)*.

[J3] **Xuan, T. T.** and Tien, V. D. and Chakka, R. The Impact of Dynamic Power Management in Computational Clusters with Multi-Core Processors. *Journal of Scientific and Industrial Research (JSIR)* , 75:339–343, June 2016. (*Impact Factor = 0.735*)*.

[J4] **Xuan, T. T.** and Tien, V. D. and Csaba, R. and Dosam, H. . A New Data Layout Scheme for Energy-Efficient MapReduce Processing Tasks. *Journal of Grid Computing*, Feb 2018. (*Impact Factor = 3.288*)*.

Conference Papers

[C1] N. H. Do, T. Van Do,**X. Thi Tran**, L. Farkas, and C. Rotter. A scalable routing mechanism for stateful microservices, *20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*,pages 72–78. March 2017.

[C2] **Xuan T. Tran.** Resource-Aware Scheduling in Heterogeneous, Multi-core Clusters for Energy Efficiency, *Advances in Information and Communication Technology: Proceedings of the International Conference, ICTA 2016*, pages 520–529. Springer International Publishing, Cham, 2017.

[C3] N. H. Do, T. V. Do, **X. T. Tran**, L. Farkas, and C. Rotter. Data I/O provision for Spark applications in a Mesos cluster, *19th IEEE International Conference on Innovation in Cloud Internet and Networking: ICIN 2016*, pages 45–52. March 2016.

[C4] **Tran Thi Xuan** and Tien Van Do. Job Scheduling in a Computational Cluster with Multicore Processors, *Advanced Computational Methods for Knowledge Engineering: Proceedings of the 4th International Conference on Computer Science, Applied Mathematics and Applications, ICCSAMA 2016*, pages 75–84. Springer International Publishing, Cham, 2016.

* Impact Factor of 2018

[C5] **X. T. Tran**, T. V. Do, N. H. Do, L. Farkas, and C. Rotter. Provision of Disk I/O Guarantee for MapReduce Applications, *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 2, pages 161–166. Aug 2015.

[C6] **Xuan T. Tran** and Binh T. Vu. A New Approach for Buffering Space in Scheduling Unknown Service Time Jobs in a Computational Cluster with Awareness of Performance and Energy Consumption, *Advanced Computational Methods for Knowledge Engineering: Proceedings of the 2nd International Conference on Computer Science, Applied Mathematics and Applications (ICCSAMA 2014)*,pages 129–139. Springer International Publishing, Cham, 2014.

Bibliography

[1] Yeo, Chee Shin and Buyya, Rajkumar and Pourreza, Hossein and Eskicioglu, Rasit and Graham, Peter and Sommers, Frank. Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers. *Handbook of Nature-Inspired and Innovative Computing: Integrating Classical Models with Emerging Technologies*, pages 521–551, 2006.

[2] Ejaz Ahmed, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Imran Khan, Abdelmuttlib Ibrahim Abdalla Ahmed, Muhammad Imran, and Athanasios V. Vasilakos. The role of big data analytics in internet of things. *Computer Networks*, 129:459 – 471, 2017. Special Issue on 5G Wireless Networks for IoT and Body Sensors.

[3] Dutta K. Distributed Computing Technologies in Big Data Analytics. *Distributed Computing in Big Data Analytics. Scalable Computing and Communications. Springer, Cham*,pages 57–82, 2017.

[4] Albert Reuther, Chansup Byun, William Arcand, David Bestor, Bill Bergeron, Matthew Hubbell, Michael Jones, Peter Michaleas, Andrew Prout, Antonio Rosa, Jeremy Kepner. Scalable System Scheduling for HPC and Big Data. *Journal of Parallel and Distributed Computing*, Vol. 111, January 2018, Pages 76–92, 2018.

[5] Rajkumar Buyya. High Performance Cluster Computing: Architectures and Systems, Vol. 1. *Prentice Hall*, 1999.

[6] Charles Lefurgy, Karthick Rajamani, Freeman Rawson, Wes Felter, Michael Kistler, and Tom W. Keller. Energy management for commercial servers. *Computer*, 36(12):39–48, December 2003.

[7] Brad Ellision and Lauri Minas. *Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers.* Intel press, 2009.

[8] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 8(3):299 –316, june 2000.

[9] Rini T. Kaushik and Milind Bhandarkar. GreenHDFS: Towards an Energy-conserving, Storage-efficient, Hybrid Hadoop Compute Cluster. In *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, HotPower'10, pages 1–9, Berkeley, CA, USA, 2010. USENIX Association.

[10] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy aware consolidation for cloud computing. *Cluster Computing*, 12:1–15, 2009.

[11] Jiankang, D., Hongbo, W., Yangyang, L., Shiduan, C.. Virtual machine scheduling for improving energy efficiency in IaaS cloud. *Communications*, 11:1–12, 2014.

[12] Beloglazov, A., Abawajy, J., Buyya, R.. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Futur. Gener. Comput. Syst.*, 28:755–768, 2012.

[13] Do, Tien Van and Rotter, Csaba. Comparison of Scheduling Schemes for On-demand IaaS Requests. *J. Syst. Softw.*, 85(6):1400–1408, June 2012.

[14] Stylianos Zikos and Helen D. Karatza. Performance and energy aware cluster-level scheduling of compute-intensive jobs with unknown service times. *Simulation Modelling Practice and Theory*, 19(1):239–250, 2011.

[15] Tien V. Do, Binh T. Vu, Xuan T. Tran, and Anh P. Nguyen. A generalized model for investigating scheduling schemes in computational clusters. *Simulation Modelling Practice and Theory*, 37(0):30–42, 2013.

[16] Anshul Gandhi, Mor Harchol-Balter, and Michael A. Kozuch. Are Sleep States Effective in Data Centers? In *Proceedings of the 2012 International Green Computing Conference (IGCC)*, IGCC '12, pages 1–10, Washington, DC, USA, 2012. IEEE Computer Society.

[17] Kyriaki Gkoutioudi and Helen D. Karatza. Multi-criteria job scheduling in grid using an accelerated genetic algorithm. *J. Grid Comput.*, 10(2):311–323, 2012.

[18] Goiri, Íñigo and Le, Kien and Nguyen, Thu D. and Guitart, Jordi and Torres, Jordi and Bianchini, Ricardo. GreenHadoop: Leveraging Green Energy in Data-processing Frameworks. In *Proceedings of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pages 57–70, New York, NY, USA, 2012. ACM.

[19] Yan Ma, Bin Gong, and Zou Lida. Energy-efficient scheduling algorithm of task dependent graph on DVS-Unable cluster system. *Proceedings of the 2009 10th IEEE/ACM International Conference on Grid Computing*, October 13-15, 2009, Banff, Alberta, Canada.

[20] Chia-Ming Wu, Ruay-Shiung Chang, and Hsin-Yu Chan. A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. *Future Generation Computer Systems,* Volume 37, Pages 141–147, July 2014.

[21] Brian Guenter, Navendu Jain, and Charles Williams. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. *In Proceedings of IEEE INFOCOM* , pages 1332–1340, IEEE, 2011.

[22] Bin Liao, Jiong Yu, Tao Zhang, Guo Binglei, Sun Hua, and Changtian Ying. Energy-efficient algorithms for distributed storage system based on block storage structure reconfiguration . *Journal of Network and Computer Applications* , 48(0):71 – 86, 2015.

[23] Sergio Nesmachnow, BernabĂŠ Dorronsoro, JohnatanE. Pecero, and Pascal Bouvry. Energy-aware scheduling on multicore heterogeneous grid computing systems. *Journal of Grid Computing*, pages 1–28, 2013.

[24] Yi Yao, Jiayin Wang, Bo Sheng, Jason Lin, and Ningfang Mi. HaSTE: Hadoop YARN Scheduling Based on Task-Dependency and Resource-Demand. In *Proceedings of the 2014 IEEE International Conference on Cloud Computing*, CLOUD '14, pages 184–191, Washington, DC, USA, 2014. IEEE Computer Society.

[25] Nezih Yigitbasi, Kushal Datta, Nilesh Jain, and Theodore Willke. Energy Efficient Scheduling of MapReduce Workloads on Heterogeneous Clusters. In *Green Computing Middleware on Proceedings of the 2Nd International Workshop*, GCM '11, pages 1:1–1:6, New York, NY, USA, 2011. ACM.

[26] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. Improving MapReduce Performance in Heterogeneous Environments. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 29–42, Berkeley, CA, USA, 2008. USENIX Association.

[27] Nitesh Maheshwari, Radheshyam Nanduri, and Vasudeva Varma. Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework . *"Future Generation Computer Systems "*, 28(1):119 – 127, 2012.

[28] L. Mashayekhy, M.M. Nejad, D. Grosu, Dajun Lu, and Weisong Shi. Energy-Aware Scheduling of MapReduce Jobs. In *Big Data (BigData Congress), 2014 IEEE International Congress on*, pages 32–39, June 2014.

[29] Arun C. Murthy, Vinod Kumar Vavilapalli, Doug Eadline, Joseph Niemiec, and Jeff Markham. *Apache Hadoop YARN: Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2*. Addison-Wesley Professional, 1st edition, 2014.

[30] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley & Sons, INC., 1991.

[31] Averill M. Law and W. David Kelton. Simulation Modeling and Analysis. *McGraw-Hill Education - Europe*, 2000.

[32] Paul A. Fishwick. Simpack: Getting started with simulation programming in c and c++. In *Proceedings of the 24th Conference on Winter Simulation*, WSC '92, pages 154–162, New York, NY, USA, 1992. ACM.

[33] V. Hlupic. Simulation software: an Operational Research Society survey of academic and industrial users. In *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, volume 2, pages 1676–1683 vol.2, 2000.

[34] L. Kleinrock. On the modeling and analysis of computer networks. *Proceedings of the IEEE*, 81(8):1179–1191, Aug 1993.

[35] Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. Apache Hadoop YARN: Yet Another Resource Negotiator. In *Proceedings of*

the 4th Annual Symposium on Cloud Computing, SOCC '13, pages 5:1–5:16, New York, NY, USA, 2013. ACM.

[36] Jacob Leverich and Christos Kozyrakis. On the Energy (in)Efficiency of Hadoop Clusters. *SIGOPS Oper. Syst. Rev.*, 44(1):61–65, March 2010.

[37] Willis Lang and Jignesh M. Patel. Energy Management for MapReduce Clusters. *Proc. VLDB Endow.*, 3(1-2):129–139, September 2010.

[38] Tom White. *Hadoop: The Definitive Guide.* O'Reilly Media, Inc., 4th edition, 2015.