SEVENTH FRAMEWORK PROGRAMME

GreenDataNet

D2.4 – Racks Multi-Objective Energy Management

Final
EATON
EPFL
Rev 0.4

EATON: Maria Laura Corallini, Gerald Guillaume, Arnaud Quette
EPFL: Pablo Garcia, Ali Pahlevan, David Atienza

TABLE OF CONTENTS

REVISION SHEET

Revision Number	Date	Brief summary of changes
Rev 0.1	27/07/2015	Baseline document
Rev 0.2	23/12/2015	Added IT consumption
Rev 0.3	11/01/2016	Added Experiments
Rev 0.4	12/01/2016	Format check and added conclusions

KEY REFERENCES AND SUPPORTING DOCUMENTATIONS

[1] J. Kim and et al., “Correlation-aware virtual machine allocation for energy-efficient datacenters,” in Design, Automation & Test in Europe (DATE) Conference, 2013, pp. 1345–1350

[2] A. Verma, et al., “pMapper: power and migration cost aware application placement in virtualized systems,” in Proc. Middleware 2008.

[3] E. Pakbaznia, et al., “Minimizing data center cooling and server power costs,” in Proc. ISLPED, 2009.

[4] N. Bobroff, et al., “Dynamic placement of virtual machines for managing sla violations,” in Proc. IM 2007.

[5] D. Meisner, et al., “Power management of online data-intensive services,” in Proc. ISCA, 2011.

[6] A. Verma, et al., “Server workload analysis for powr minimization using consolidation,” in Proc. USENIX, 2009.

[7] X. Meng, et al., “Efficient resource provisioning in compute clouds via VM multiplexign,” in Proc. ICAC, 2010.

[8] M. Chen, et al., “Effective VM sizing in virtualized data centers,” in Proc. IM, 2011.

[9] K. Halder, et al., “Risk aware provisioning and resource aggregation based consolidation of virtual machines,” in Proc. Cloud, 2012.

[10] M. Ferdman, et al., “Clearing the clouds: a study of emerging scale-out workloads on modern hardware,” in Proc. ASPLOS, 2012.

[11] E. Schurman et al., “The user and business impact of server delays, additional bytes, and HTTP chunking in web search,” in Velocity, 2009.

[12] H. Goudarz, et al., “Energy-efficient virtual machine replication and placement in a cloud computing system,” in Proc. Cloud 2012.

[13] M. Pedram, et al., “Power and performance modeling in a virtualized server system,” in Proc. ICPPW, 2010.

[14] A. Menon, et al., “Diganosing performance overheads in the xen virtual machine environment,” in Proc. VEE, 2005.

[15] J. Kim, et al., “Free cooling-aware dynamic power management for green datacenters,” in Proc. HPCS, 2012.

[16] T. Benson, et al., “Understanding data center traffic characteristics,” in ACM SIGCOMM Computer Communication Review, 2010.

[17] Y. Guo, et al., “Reliability-aware power management for parallel realtime applications with precedence constraints,” in Proc. IGCC, 2011.

2. INTRODUCTION

2.1 DOCUMENT PURPOSE

In deliverable *D2.3 – Server Multi-level software management*, an IT power optimization SW based on real values coming from the Data Centre power chain has been deployed. The tool developed in D2.3 has been upgraded with a new module, called vCenter connector, which is able to communicate and control the virtual management tool, and it is described in *D3.9-Virtualisation of IT tasks*. This deliverable describes how the optimization algorithm has been modified in order to take into account the new data provided by the vCenter connector and how this algorithm has been improved so that it can optimize the IT power consumption at a rack level. Moreover, this document provides an explanation of how the GreenDataNet Rack controller is organized to integrate the new vCenter module.

2.2 DEFINITION, ACRONYMS AND ABBREVATIONS

2.2.1 KEY ACRONYMS AND ABBREVATIONS

BFD	Best-Fit-Decreasing
CPU	Central Processing Unit
DC	Data Centre
ESXI	VMware Hypervisor
IPC	Instruction Per clock Cycles
ISN	Index Searching Node
IT	Information Technology
LXC	Linux Container
MPKI	Miss-Per-Kilo-Instruction
PDU	Power Distribution Unit
ePDU	Rack Power Distribution Unit
PCP	Peak Clustering-based Placement
PSU	Power Supply Unit
RC	Rack Controller, synonym of GC – GreenDataNet Controller
SLA	Service Level Agreement
SW	Software
UPS	Uninterruptible Power Supply
VM	Virtual Machine
WP	Work Package

2.3 DOCUMENT OVERVIEW

In this deliverable we will first explain the SW architecture, detailing how the Rack Controller interacts with the Server Manager, including examples of the communication interface. Then, we will focus on the algorithm for rack energy management, describing the new opportunities for power optimization and how to effectively implement a correlation-aware algorithm. Finally, in the experiments part, we illustrate the applicability and advantages of the developed optimization framework through two real-life examples.

3. SOFTWARE ARCHITECTURE

3.1 COMMUNICATION MODEL OF THE GLOBAL SYSTEM

Figure 3.1 depicts an overview of how the optimization algorithm running in the Server Manager (with the help from the Green Energy Controller) interfaces with the Rack Controller in order to take into account the new data provided by the vCenter.

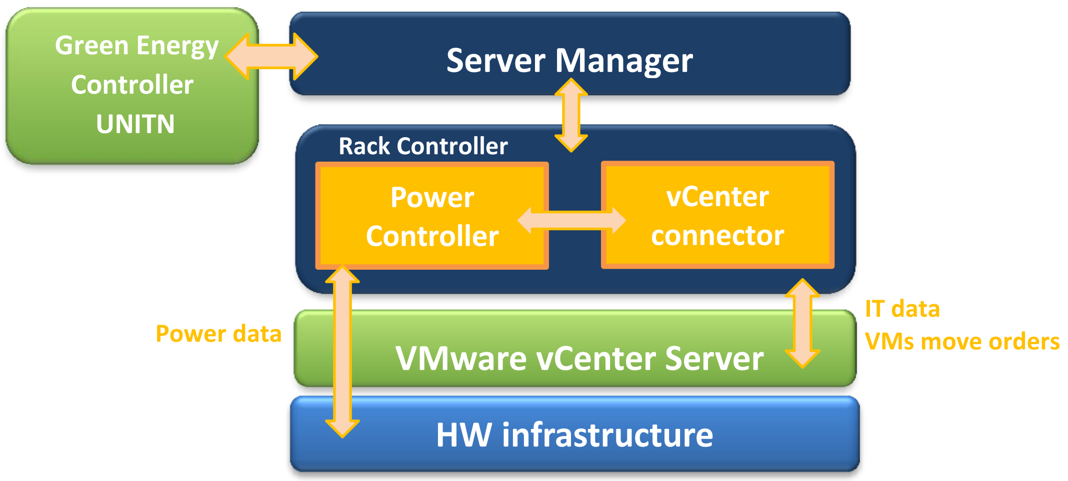


Figure 3.1 - Communication model of the global solution overview

3.2 INTERACTION BETWEEN RACK CONTROLLER AND SERVER MANAGER

3.2.1 JOINT SW STRUCTURE

As already explained in D2.3, the models developed by the different partners run as services.

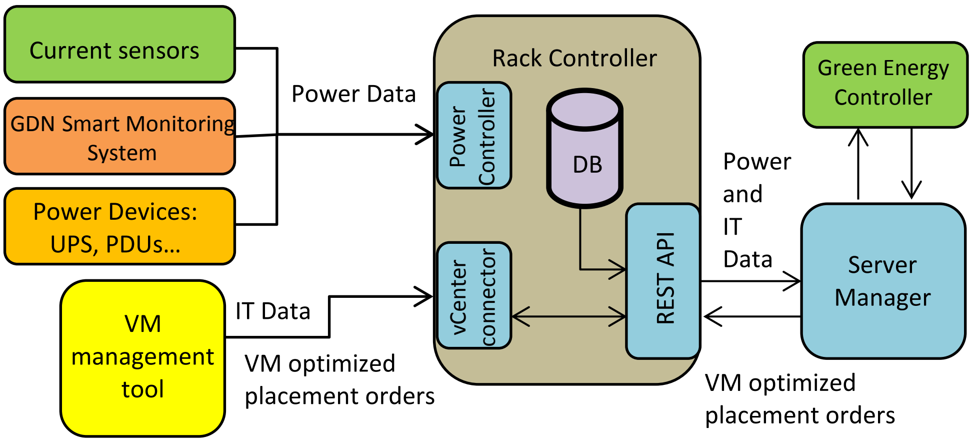


Figure 3.2 - Joint Software structure

The Rack Controller, from Eaton, is composed by 2 functional modules: the Power Controller and the vCenter connector that communicate via an internal REST API detailed in D3.9.

The power Controller provides control and monitoring features for the UPS, ePDU (Eaton´s PDU), and some environmental sensors. The vCenter connector communicates directly with the virtualisation management tool, collecting the IT data and sending the VM migration orders. All the information is stored in a database that can be accessed from Eaton´s API or directly through a local webserver.

3.2.1 PACKAGING OF THE RACK CONTROLLER AND SERVER MANAGER

In order to ease the installation of the GreenDataNet Software tools developed by the different partners, the Softwares developed by EPFL and by EATON have been packed in one single Linux container running under the Debian Jessie 64 bits OS.

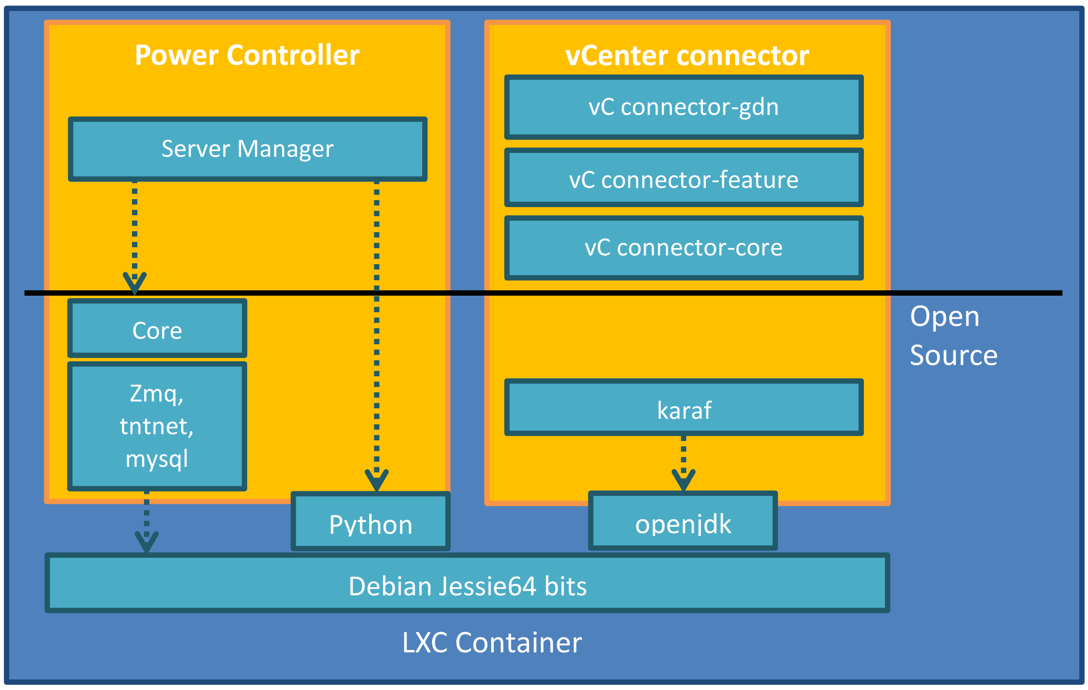


Figure 3-3 - Rack Controller packaging

3.2.2 COMMUNICATION INTERFACE

With the integration of the vCenter module, the JSON REST API of the Rack Controller has been updated with new requests for the Server Manager in order to be able to get IT information. Below it is detailed the list of all the new requests implemented, with an example of the format of the details returned based on the configuration of Figure 3-4:

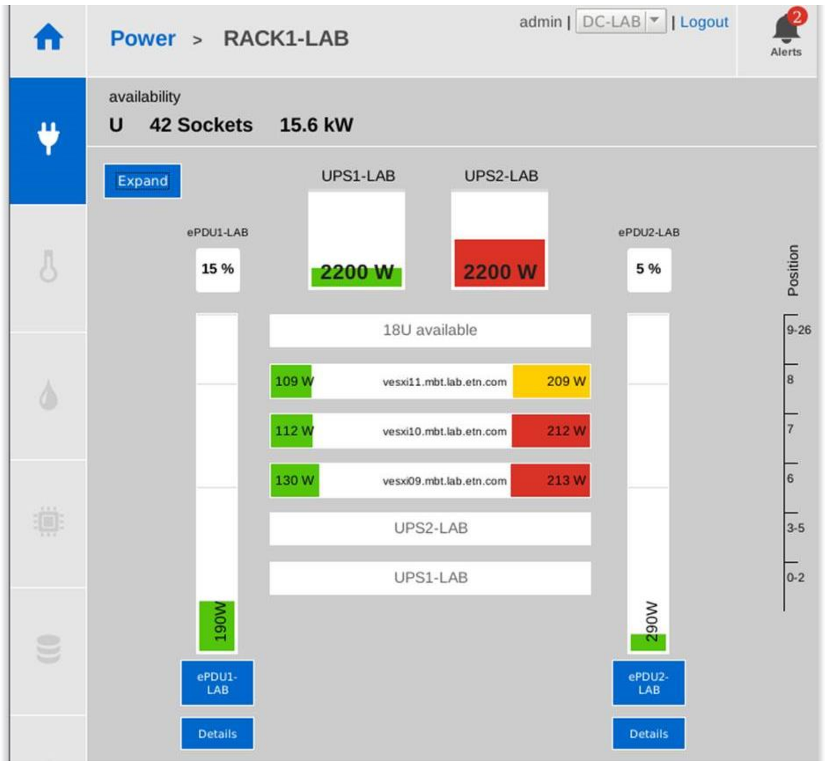


Figure 3-4 Test Rack configuration

- Get the list of the servers, the method returns the list of EXSI and VM

  GET https://<host>/api/v1/asset/devices?subtype=server

```
{"devices":[
      {"id" : "10","name" : "vesxi09.mbt.lab.etn.com"},
      {"id" : "11","name" : "vesxi10.mbt.lab.etn.com"},
      {"id" : "12","name" : "vesxi11.mbt.lab.etn.com"},
      {"id" : "13","name" : "vm-011"},
      {"id" : "14","name" : "vm-012"},
      {"id" : "15","name" : "vm-013"},
      {"id" : "16","name" : "vm-021"},
      {"id" : "17","name" : "vm-022"},
      {"id" : "18","name" : "vm-023"},
      {"id" : "19","name" : "vm-031"}]}
```

- Get the details of each server.

  In the example below, the request returns the details of an ESXI. It is important to notice that, in addition to the standard information such as hostname and location of the ESXI, the

request provides also the power chain of the server and, more specifically, in which socket of the power distribution unit the hosting server is plugged.
Moreover, the ext/v12n.type attribute has been implemented in order to distinguish an ESXI server (v12n.type = virtualization.host) from a VM (v12n.type =virtualization.machine).

GET https://<host>/api/v1/asset/device/12

```
{
	"id": "12",
	"name": "vesxi11.mbt.lab.etn.com",
	"location_uri":"/api/v1/asset/rack/5" <- the server is in Rack
	5,
	"type": "server",
	"hostname":"vesxi11",
	"fqdn":"vesxi11.mbt.lab.etn.com",
	"powers": [
		{"src_uri":"/api/v1/asset/device/8" <- the first Power
		Supply Unit (PSU) is connected to pdu id 8, on plug
		socket 9,
		"src_socket":"9",
		"dest_socket":"1"},
		{"src_uri":"/api/v1/asset/device/9" <- the second PSU of
		this server is connected to pdu id 9, socket 9,
		"src_socket":"9",
		"dest_socket":"2"}],
		"ext" : [
			{"description":"VMware ESXi 5.5.0 build-1331820
			hosted on vesxi11.mbt.lab.etn.com supervising by
			vcenter04.mbt.lab.etn.com",
			"read_only": false },
			{"v12n.type":"virtualization.host",
			"read_only": false } <- This is a ESXi]
}
```

This request returns the VM details
GET https://<host>/api/v1/asset/device/19

```
{
	"id": "19",
	"name": "vm-031",
	"status": "nonactive",
	"business_critical": "no",
	"priority": "P5",
	"location_uri":"/api/v1/asset/device/12" <- The VM runs on the
	server ESXi vesxi11 (device ID=12),
	"groups": [],
	"type": "server",
	"powers": [],
	"ext" : [{"asset_tag": "20855316-5F9A-4975-9C28-46F3C50188F1-
	vm-031", "read_only": true },
	{"v12n.type":"virtualization.machine","read_only": false }] <-
	This is a VM
..}
```

- Get the current metrics of a VM. The available data are described in Table 3-1
  GET https://<host>/api/v1/metric/current?dev=19

```
{
      "current":
      [
      {"id" : "19","name" : "vm-031",
      "cpu" : 31.000000, <- unit %
      "disk.nominal" : 307228632000000.000000,  <- unit: GB
      "operatingStatus" : "In_Service", <- VM is running
      "vm-011!comm" : 71.000000, <- data communication between vm-031
      and vm-011
      "vm-012!comm" : 59.000000,
      "vm-013!comm" : 67.000000,
      "vm-021!comm" : 79.000000,
      "vm-022!comm" : 76.000000,
      "vm-023!comm" : 22.000000}
      ]
}
```

Monitored IT data	Value	
CPU allocation per virtual machine per server	[Hz]	
Data communication between VMs	[MBs]	
Disk size of the VMs (Total provisioned one)	[GB]	
Time when the VM has been started	[Date]	
VM status	Name	Meaning
	In Service	Powered and working
	Stopped	Turned off
	Servicing	Turning on phase
	Stopping	Turning off phase
	Dormant	Stand-by mode

Table 3-1 List of monitored Data

- Get current metrics about the power consumption of the server (esxi11).
  To have this information, the Server Manager needs to request the power consumption per plug supplying the server, required previously with the request GET VM details.
  GET https://<host>/api/v1/metric/current?dev=8

```
{
   "current":[
    {"id" : "8","name" : "ePDU1-LAB",
,"realpower.outlet.9" : 109.000000, <- realpower of PSU1 of server
esxi09
"voltage.outlet.9" : 243.730000} , … ]}
{
   "current":[
    {"id" : "9","name" : "ePDU2-LAB,
"realpower.outlet.9" : 209.000000 , <- realpower of PSU2 of server
esxi09
}   ]
}
```

4. OPTIMIZING THE IT POWER CONSUMPTION AT THE RACK LEVEL

The previous section explained how the optimization algorithm obtains the data provided by the vCenter connector. Once the data is acquired, it will be used by the datacentre management algorithm so that it can optimize the IT power consumption at the rack level.

Server consolidation [3], minimizes the number of active servers by packing workloads, or virtual machines (VMs) in a virtualized environment, into the minimal number of active servers, is one of the widely used techniques to reduce the power consumption of datacentres. Instead of assuming the worst-case (or peak) utilization [2], recently, correlation of resources utilization patterns among VMs are also exploited, such that, un-correlated VMs are co-located into a server to enable overprovision of VMs under negligible QoS degradation [6]. Nonetheless, these existing solutions are mostly designed for high-performance computing (HPC) applications and do not work well for emerging cloud (or scale-out [10]) applications (e.g., web search, MapReduce, etc.) due to the lack of considerations of the characteristics of the scale-out applications:

- User-interactive; therefore, required computing capacity is highly variable and fast-changing.
- Latency is the first criteria to be satisfied.
- The amounts of required CPU and memory resources are usually far beyond the level that a single server can sustain.
- The memory footprint is far beyond the amount an on-chip cache can sustain; thereby, increasing the on-chip cache size only produces negligible performance improvement.

Because of these aforementioned discrepancies with HPC workloads, existing datacentre power management solutions, which neglect or only partially consider the characteristics of scale-out applications, do not exploit all the opportunities to achieve global power savings. By analysing the workload characteristics of scale-out applications we can uncover new opportunities for power management in virtualized server environments.

Our multi-objective optimization algorithm is a dynamic power management solution for servers hosting these new scale-out applications, especially accounting for the correlation information among VMs, while satisfying peak resource requirements.

4.1 NEW OPPORTUNITIES FOR POWER MANAGEMENT

Power management solutions for datacenters hosting scale-out applications should be different from the case of hosting HPC applications due to the distinctive characteristics of these type of applications. In this section, we present three principles of dynamic power management solutions that jointly utilize server consolidation and voltage and frequency (hereafter, v/f) scaling.

4.1.1 CONSERVATIVE RESOURCE PROVISION WITH V/F SCALING

Scale-out applications are user-interactive. Therefore, responsiveness, in terms of latency, is the first priority to be met [11]. Moreover, every application (or VM) is assumed to be equally important in clouds. Thus, we should conservatively provision VMs based on the peak (or Nth percentile according to QoS requirement) resource demand of each VM. The required QoS level can be achieved by assigning the right number of cores because the performance is highly scalable to the number of allocated cores due to the high parallelism of such applications. Moreover, the resource demand is time-varying and is mostly lower than the value used for the core allocation. However, as described in [5], dynamic power gating (turning on/off cores) cannot be applicable to such applications due to the significant performance degradation caused by the long transition latency between power modes and fast changes of resource demands. Thus, dynamic v/f scaling is the only solution to achieve power savings while satisfying the performance requirement. Motivated by this observation, the proposed solution allocates the number of cores for each VM according to its peak (or off-peak depending on QoS level) resource demand to guarantee equal QoS levels to all VMs while scaling v/f level to achieve power savings.

4.1.2 SHARING CORES AMONG CO-LOCATED VMS

In scale-out applications, massively parallel nodes are cooperatively working by forming a cluster architecture [12]. For instance, in a web search application, a big set of search indexes is divided into multiple smaller datasets, and then allocated into multiple VMs (or servers), each of which is called an index searching node (ISN). Once a query arrives, each ISN independently searches matched data with the allocated dataset and a master (i.e., front-end) node gathers the search results from the multiple ISNs, then sends the results to clients.

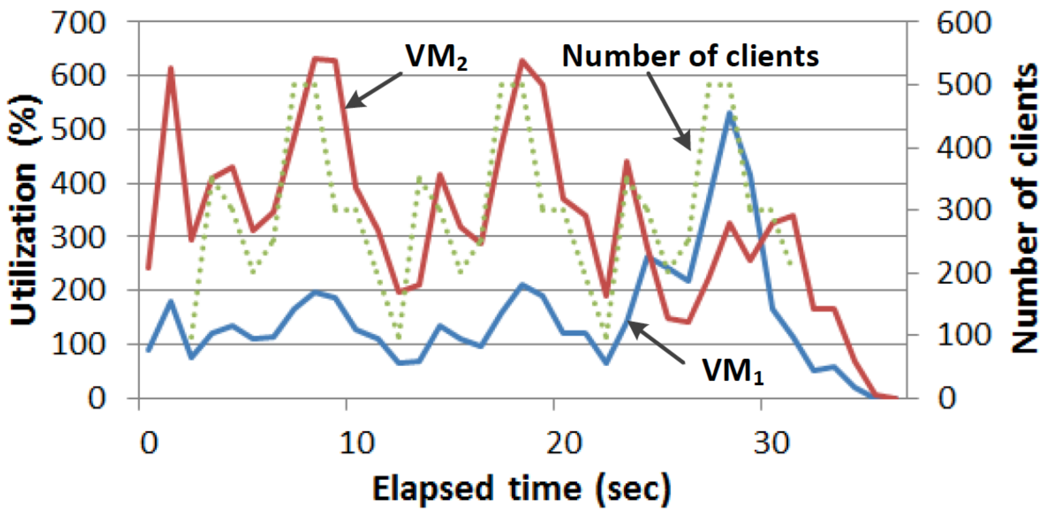


Figure 4.1 - Variations of CPU utilization of two index searching nodes (ISNs) with respect the number of clients

The amount of required CPU utilization varies as the amount of user requests to servers changes over time. Figure 4.1 shows the CPU utilization traces for two VMs (all data presented in this section is measured using an AMD Opteron 6174 architecture within a DELL PowerEdge R815 server), each of which is an index serving node (ISN), in a single web search cluster to process queries requested from the varying number of clients. As shown in the figure, the CPU utilizations of both VMs are highly synchronized with the variation of the number of clients. Furthermore, loads between VMs in a cluster are not perfectly balanced because the CPU utilization depends on the amount of matched results corresponding to a user request. Thus, we can improve the resource utilization by sharing cores among multiple VMs, such that they can flexibly use cores depending on their time-varying resource demands. Furthermore, as analyzed in [10], the overhead of sharing cores is negligible due to the large memory footprint of scale-out applications, i.e., far beyond the capacity of on-chip caches.

	IPC	L2 MPKI	L2 miss rate (%)
w/ Backshcoles	0.76 (0.75)	2.38 (2.40)	11.28 (11.57)
w/ Swaptions	0.75 (0.77)	2.32 (2.43)	11.02 (9.63)
w/ Facesim	0.70 (0.70)	2.41 (2.36)	11.41 (11.31)
w/ Canneal	0.76 (0.78)	2.46 (2.43)	11.76 (11.67)

Table 4.1 - Performance metrics of a web search application co-located with a VM running parsec benchmark: numbers in parenthesis show the case when a web search application is running alone

Table 4.1 shows the measured performance metrics used of a web search application when it is co-located with various applications (from PARSEC benchmark suite). We compared instruction per clock cycles (IPC), L2 miss-per-kilo-instruction (MPKI), and L2 miss ratio (%). The values are obtained using Xenoprof patched for the AMD15h Bulldozer architecture [14]. The numbers in parenthesis show the case before colocation. As can be seen, there are only negligible variations over all the metrics, which correspond to a negligible performance degradation due to cores sharing. Motivated by these observations, the proposed solution allocates VMs to servers such that all co-located VMs share cores assuming that the performance degradation is negligible.

4.1.3 CORRELATION-AWARE VM PLACEMENT

Due to the distributed operations of multiple VMs in a cluster, we can observe a high correlation within a cluster of scale-out applications, called intra-cluster correlation, rather than the correlation among different clusters (or services) targeted in other correlation-aware scheme [6]–[9]. In Figure 4.1, we can observe the intra-cluster correlation between two VMs in a cluster, both of which are strongly synchronized with the variation of the number of clients. Thus, the proposed solution takes into account the pervasive correlation in scale-out applications, i.e., within a cluster as well as among clusters, such that correlated VMs are not co-located.

4.2 CORRELATION-AWARE POWER MANAGEMENT

In this section, we present the proposed datacentre power management solution. First, we define a cost function to efficiently quantify the level of correlation used in the proposed VM placement. Second, we propose the correlation-aware VM allocation scheme while sharing cores among co-located VMs. Finally, we provide a way to scale the v/f level to achieve power savings without any QoS degradation.

We assume homogeneous servers; and each of them consists of Ncore cores with multiple frequency levels.

4.2.1 EFFICIENT CORRELATION MEASURE FOR VM ALLOCATION

The correlation of used CPU utilization between two VMs is mostly quantified with Pearson product-moment correlation coefficient, or Pearson’s correlation [8], which is calculated as the ratio of covariance of the two random variables to the product of their standard deviations. However, the overhead to calculate the metric for a certain time interval is high for a short time period because the computation is concentrated at the end of the time period, as it utilizes the average values of CPU utilization samples, which are collected during each time period. In addition, Pearson's correlation is also partly inefficient because the value reflects correlation throughout the corresponding time interval, even though we only require the correlation at (off-)peak utilizations in VM placement. To overcome the drawback and inefficiency in this metric, we propose a new cost function to quantify correlation between two VMs (in terms of CPU utilization), as follows:

(1) $$Cost_{i,j}^{vm} = \frac{\hat{u}_{cpu}(VM_i) + \hat{u}_{cpu}(VM_j)}{\hat{u}_{cpu}(VM_i + VM_j)}$$

where $Cost^{vm}_{i;j}$ represents the newly defined correlation measure between VM_i and VM_j. $\hat{u}_{cpu}(VM_i)$ is a reference utilization of VM_i, which is either the peak or the Nth percentile value depending on QoS requirement. The numerator represents the worst-case peak CPU utilization when the peaks of two VMs coincide, while the denominator is an aggregated actual peak utilization when VM_i and VM_j are collocated into a same server. Thus, the higher $Cost^{vm}_{i;j}$, the lower correlation between VM_i and VM_j . Moreover, we can update the values at each sampling period of utilization. Thus, we can save memory space to store all samples as well as evenly distributing computational effort to measure the correlation across a certain time horizon.

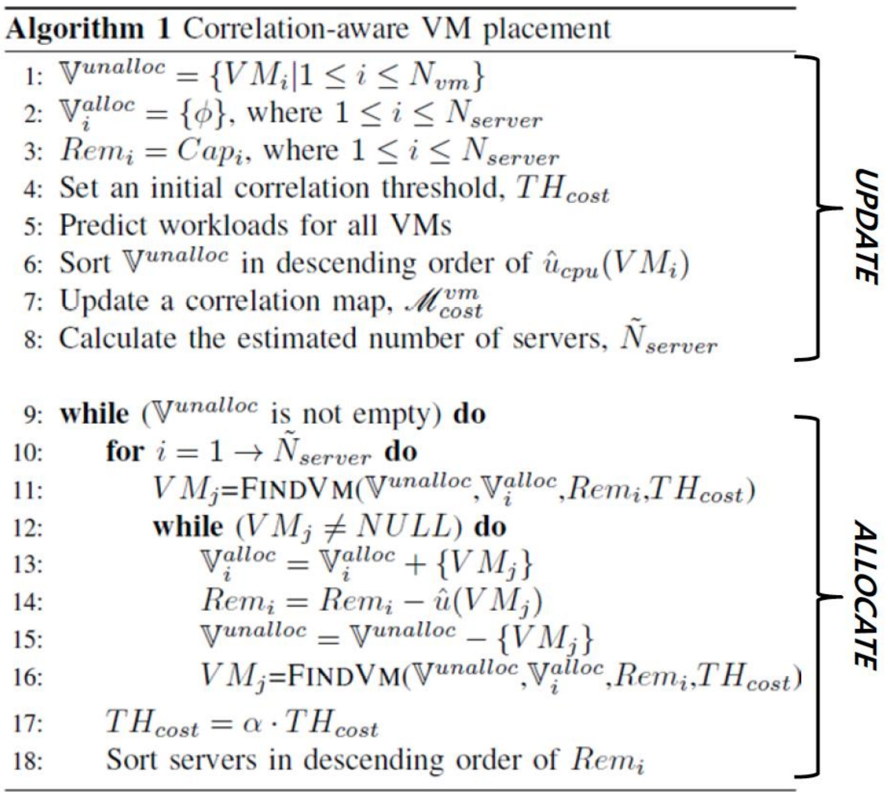
Using our new $Cost^{vm}_{i;j}$ function, we can model correlations among all VMs by constructing a 2-D matrix, namely, M^{vm}_{cost} where the (i,j)-th element corresponds to $Cost^{vm}_{i;j}$.

4.2.2 CORRELATION-AWARE VM ALLOCATION

We allocate VMs such that the correlation among the allocated VMs in Serveri, i.e., $\mathbb{V}_i^{alloc} = \{VM_{i,1}, \cdots, VM_{i,n_i^{vm}}\}$
where n^{vm}_i is the number of VMs allocated to Serveri, is minimized, while the sum of ûcpu(VMi;j) in the server does not exceed the total CPU capability of the server, i.e., Capi, as well as the number of the active servers is minimized. The correlation of Serveri is defined as shown in

(2) $$\overline{Cost_i}^{server} = \sum_{j=1}^{n_i^{vm}} w_{i,j}^{vm} \cdot \left(\sum_{k=1\&k\neq j}^{N_{vm}} \frac{Cost_{j,k}^{vm}}{n_i^{vm} - 1} \right)$$

where $w^{vm}_{i;j}$ represents a weight of VMi;j , defined as the ratio of û(VMi;j) to the sum of û(VMi;j)'s of all co-located VMs in Serveri. The problem of finding optimal sets of VMs is a well-known bin-packing problem [15]. To reduce the solution complexity, we propose a solution based on a First-Fit-Decreasing heuristic as shown in Figure 4.2. Our proposed algorithm is periodically invoked at every tperiod. The algorithm is largely divided into two phases: 1) UPDATE (lines 1-8) and 2) ALLOCATE (lines 9-18). In the UPDATE phase, we initialize parameters and update CPU utilization statistics. Then, we allocate VMs to servers in the ALLOCATE phase.


UPDATE
ALLOCATE

```
Algorithm 1 Correlation-aware VM placement
1: $\mathbb{V}^{unalloc} = \{VM_i | 1 \le i \le N_{vm}\}$
2: $\mathbb{V}_i^{alloc} = \{\phi\}$, where $1 \le i \le N_{server}$
3: $Rem_i = Cap_i$, where $1 \le i \le N_{server}$
4: Set an initial correlation threshold, $TH_{cost}$
5: Predict workloads for all VMs
6: Sort $\mathbb{V}^{unalloc}$ in descending order of $\hat{u}_{cpu}(VM_i)$
7: Update a correlation map, $\mathscr{M}_{cost}^{vm}$
8: Calculate the estimated number of servers, $\tilde{N}_{server}$

9: while ($\mathbb{V}^{unalloc}$ is not empty) do
10:     for $i = 1 \to \tilde{N}_{server}$ do
11:         $VM_j$=FINDVM($\mathbb{V}^{unalloc}$,$\mathbb{V}_i^{alloc}$,$Rem_i$,$TH_{cost}$)
12:         while ($VM_j \ne NULL$) do
13:             $\mathbb{V}_i^{alloc} = \mathbb{V}_i^{alloc} + \{VM_j\}$
14:             $Rem_i = Rem_i - \hat{u}(VM_j)$
15:             $\mathbb{V}^{unalloc} = \mathbb{V}^{unalloc} - \{VM_j\}$
16:             $VM_j$=FINDVM($\mathbb{V}^{unalloc}$,$\mathbb{V}_i^{alloc}$,$Rem_i$,$TH_{cost}$)
17:     $TH_{cost} = \alpha \cdot TH_{cost}$
18:     Sort servers in descending order of $Rem_i$
```

Figure 4.2 - The proposed correlation-aware VM placement consisting of UPDATE and ALLOCATE phases

In the UPDATE phase, we first initialize a set of unallocated VMs (Vunalloc), sets of allocated VMs (Valloc i), remaining capacity (Remi) for all servers, and a correlation threshold (THcost) in lines 1-4. Second, we predict the workload based on history, as we previously prepared in [15] (line 5). Third, we sort VMs in Vunalloc in descending order of predicted ûcpu(VMi) to reduce the fragmentation of the bin-packing problem (line 6). Fourth, we update M^{vm} corr by updating the $Cost^{vm}$ i;j for all VM pairs (line 7). Finally, we determine the number of estimated active servers, i.e., Ñserver , as presented in Eqn. (3) (in line 8):

$$\tilde{N}_{server} = \frac{\sum_{i=1}^{N_{vm}} \tilde{\hat{u}}_{cpu}(VM_i)}{N_{core}} \quad (3)$$

where ~ûcpu represents an estimate of ûcpu . Then, Ñserver is equal to the minimum number of servers to accommodate all VMs in Vunalloc. We provision VMs to reduce the number of active servers while satisfying performance requirements. The ALLOCATE phase is iterated until all VMs are allocated to Ñserver servers (line 9). First, we select a server having the largest remaining CPU capability, i.e., Remi (line 10). Second, we find a VM to be allocated into Serveri (line 11), which has the highest Costserver i with VMs in Valloc i , while satisfying two conditions: 1) Cost server i should be larger than THcost ; and 2) ûcpu(VMi) should be less than or equal to Remi . In case we find a VM, we update Valloc i , Remi , and Vunalloc accordingly (lines 12-15). The procedure to find VMs to be allocated in Serveri is iterated until there is VM left (lines 12-16). If we have unallocated VMs at the end of the iteration, we repeat the procedure (from lines 10-16) with a degenerated THcost by a factor of α (line 17) along with a list of servers sorted in descending order of Remi (line 18).

4.2.3 DECISION OF V/F LEVEL

Once all VMs are allocated into servers, we determine an optimal v/f level for each server. However, we cannot exactly estimate how much we can lower v/f level when multiple VMs are allocated in a server because Cost^{vm} i;j only captures the correlation between two VMs.

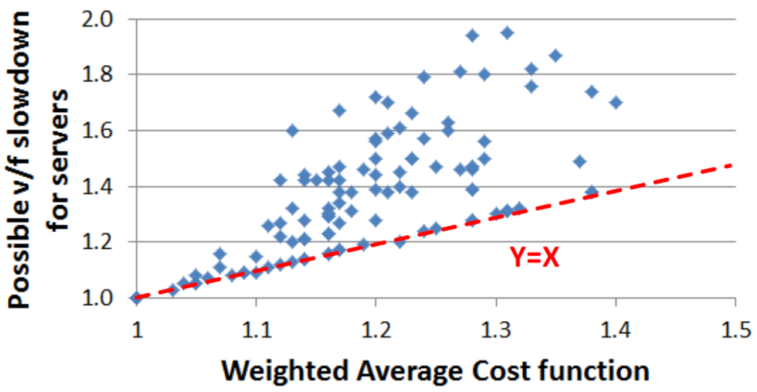


Figure 4.3 - Relationship between weighted average correlation in Eqn. (2) and possible v/f scaling factor: the lower bound of the possible v/f scaling factor has linear relationship with Cost server i

Therefore, we empirically calculate the lower bound of v/f slowdown through Cost server i in Eqn. (2), as shown in Figure 4.3. X- and Y-axes, respectively, represent a weighted average cost function calculated with Eqn. (2) and the ratio of the sum of ûcpu(VMi)'s of collocated VMs to the aggregated peak value of the server, which represents possible v/f slowdown. Based on the relationship, we can determine the frequency level of Serveri , i.e., fi , as presented in Eqn. (4):

(4) $$f_i = \left(\frac{1}{\overline{Cost_i^{server}}}\right) \cdot \left(\frac{\sum_{j=1}^{n_i^{vm}} \hat{u}_{cpu}(VM_{i,j})}{N_{core}^{server}}\right) \cdot f^{max}$$

where f^{max} is the maximum frequency level. fi is set by lowering the worst-case peak required frequency level (i.e., the second parenthesis assuming the situation when peaks of VMs coincide) with a factor of 1=Cost Server i .

5. RESULTS

We validated the proposed datacentre power management approach in two setups. First, we applied the proposed solution to two web search clusters running on DELL PowerEdge R815 servers to validate the applicability of the proposed correlation-aware scheme for scale-out applications. Second, we further investigated the effectiveness to larger scale problems with the utilization traces obtained from a real datacenter setup.

5.1.1 SETUP-1: DISTRIBUTED WEB SEARCH APPLICATIONS

We built two web search clusters, i.e., Cluster1, and Cluster2, using the CloudSuite benchmarks [10]. Each cluster consists of three VMs: one is front-end (Tomcat-7.0.23) and two are ISNs (Nutch-1.2). Note that the CPU utilization of the front-end is quite low compared to ISNs. Thus, we simply varied the allocation of VMs hosting ISNs. We annotate four ISNs as VM1;1, VM1;2, VM2;1, and

VM2;2 where {VM1;1; VM1;2} and {VM2;1; VM2;2} are included in Cluster1 and Cluster2, respectively. We used Xen-4.1 hypervisor for server virtualization and each VM has Ubuntu11.10 as its operating system (OS). We emulated clients' behavior using Faban-0.7 and varied the number of clients from 0 to 300 with the form of sine and cosine waves for Cluster1 and Cluster2, respectively. We used two servers each of which consists of 8 cores having two frequency levels, i.e., 1.9GHz and 2.1GHz.

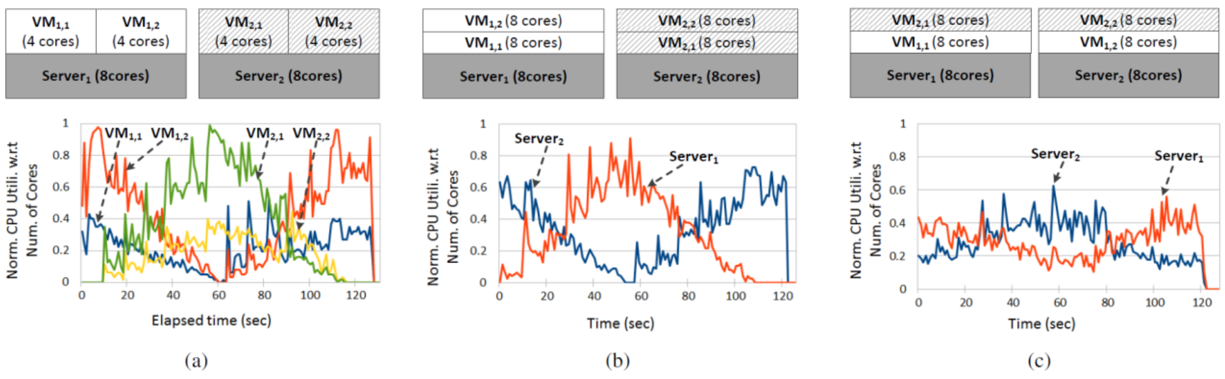


Figure 5.1 - VM placements and CPU utilization traces of (a) Isolated, (b) Shared-UnCorr, and (c) Shared-Corr

We compared three different VM allocations, as illustrated in the upper part of Figure 5.1. 1) Segregated where each VM is independently running on 4 cores each, 2) Shared-UnCorr where 8 cores are shared with two VMs in a same cluster (i.e., correlation unawareness), and 3) Shared-Corr where 8 cores are shared with two VMs in different clusters (i.e., including correlation awareness). Then, Figure 5.2 shows comparisons in terms of the 90th percentile response time.

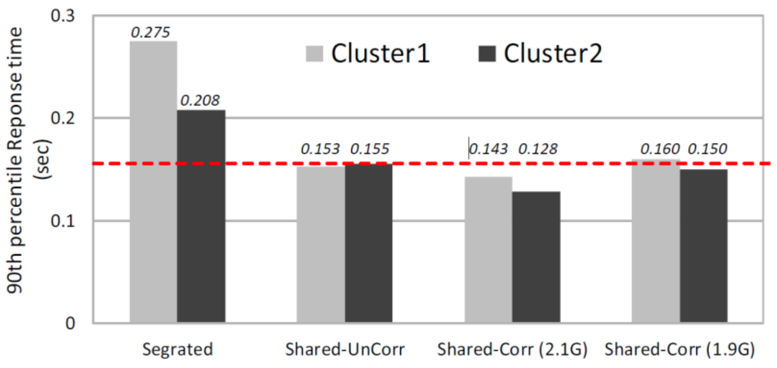


Figure 5.2 - 90th percentile response time of Cluster1 and Cluster2 for three different VM allocations

As this figure indicates, the 90th percentile response time in Shared-UnCorr is lower than Segregated by 43.6% (from 0.275 to 0.155 sec) while Shared- Corr provides another 7.7% lower response time (from 0.155 to 0.143 sec) than Shared-UnCorr under 2.1GHz. The results can be explained by observing the CPU utilization traces in Figure 5.1. The X- and Y-axes represent the elapsed time (in sec) and the normalized CPU utilization with respect to the number of servers, respectively. The samples are collected at every 1 sec using a Perl script monitoring tool Xenstat.pl. The reason of the high response time in Segregated case is the inefficient utilization of the allocated cores. As shown in Figure 5.1(a), VM1;1 and VM2;2 are under-utilized while VM1;2 and VM2;1 are over-utilized, i.e., approaching their maximum CPU utilization levels, and needs more than 4 cores.

Note that the response time of the distributed web search cluster is constrained by the latest VM because a front-end sends results to clients only after collecting the search results from all ISNs. Thus, due to the deficiency of the CPU capability of the over-utilized VMs, queries must wait in a queue for a longer time before being processed. Thus, the response time of Segregated case becomes longer. On the contrary, Shared-UnCorr enables to efficiently use all the 8 cores in each server by flexibly scheduling VMs to the cores according to their time-varying demands. This result supports our claim where we anticipated that the gain attaining from sharing cores among VMs is much higher than the performance degradation caused by the interference among co-located VMs. However, the maximum CPU utilization reaches up to 0.88 because two VMs within the same cluster are highly correlated. Hence, the peaks of the CPU utilizations coincide. Such high CPU utilization can result in longer response times [13]. We can reduce the peak utilization by allocating VM considering correlations among VMs in Shared-Corr (Figure 5.1 (c)). In Shared-Corre, the maximum CPU utilization becomes even and lowered down to 0.6. The improved response time in Shared-Corr can be used to save power consumption by lowering the frequency level. As shown in Figure 5.2, Shared-Corr running with 1.9GHz provides almost similar response time (0.155 vs. 0.160 sec) to Shared running with 2.1GHz, which results in approximately 12% power savings.

5.1.2 SETUP-2: UTILIZATION TRACES OBTAINED FROM DATACENTER SETUPS

To further investigate the effectiveness of the proposed solution, we performed another set of simulations using utilization traces obtained from an actual datacenter. As most of VMs are severely under-utilized, we selected the top 40 VMs in terms of CPU utilization. We sampled the CPU utilization every 5 min. for a day while synthesizing fine-grained samples per 5 sec. with a lognormal random number generator [16], whose mean is the same as the collected value for the corresponding 5-minute sample rate. Using these utilization traces, we evaluated the effectiveness of the proposed solution with a virtual testbed consisting of 20 servers. We targeted an Intel Xeon E5410 server configuration which consists of 8 cores and two frequency levels (2.0GHz and 2.3GHz), and used the power model proposed in [13]. We performed VM placement every 1 hour, i.e., t_{period}=1 hour, with predictions of upcoming workloads using a last-value predictor. Then, we compared the following three approaches of power management for datacenters:

- Best-Fit-Decreasing (BFD): a conventional best-fit-decreasing heuristic approach.
- Peak Clustering-based Placement (PCP) [6]: a correlation-aware VM allocation which clusters VMs using its Envelope-based correlation classification.
- Proposed: the proposed correlation-aware VM allocation.

(a)

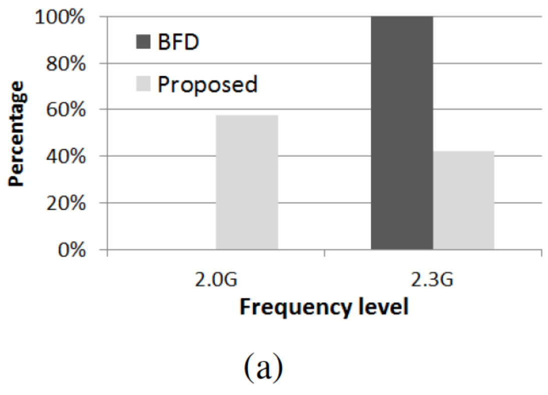
	Normalized power	Maximum violations (%)
BFD	1	18.2
PCP [6]	0.999	18.2
Proposed	**0.863**	**2.6**

(b)

	Normalized power	Maximum violations (%)
BFD	1	20.3
PCP	0.997	20.3
Proposed	**0.958**	**3.1**

Table 5.1 - Comparisons for (a) static and (b) dynamic v/f scaling

Table 5.1(a) compares the power consumption and performance violations of the three approaches when we statically set the v/f level at the time of VM placement, i.e., t_{period}. The power consumption results are normalized with respect to the power consumed by BFD, and the maximum violation shows the maximum per-period ratio of the number of over-utilized time instances (i.e., when the aggregated utilization among collocated VMs is beyond the CPU capacity of a corresponding server) to t_{period}, during the entire periods, i.e., 24 hours. The proposed solution provides up to 13.7% power savings compared to BFD and PCP, while drastically reducing the number of the violations. It is noteworthy that PCP provides almost similar results with BFD because, due to high and fast-changing correlations among VMs in our utilization traces, PCP classifies VMs into only '1' cluster during the most of the time periods (22 out of 24 time periods). When the number of clusters is '1', PCP behaves exactly same with BFD.



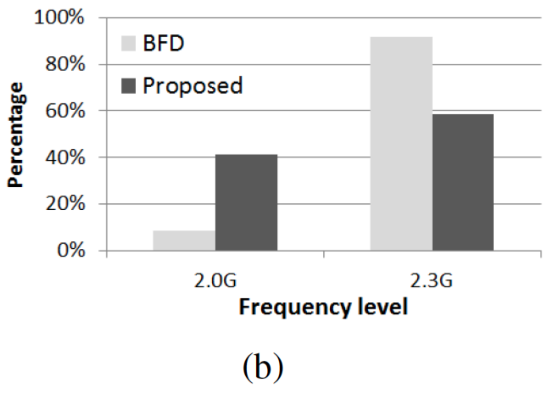


Figure 5.3 - Comparison of frequency distributions in (a) Server1 and (b) Server3

The power savings obtained by our proposed solution are due to the aggressive-yet-safe v/f settings utilizing the lowered actual peak resource demand, i.e., Eqn. (4). Figure 5.3 compares the distributions of used frequency levels of BFD and the proposed solution in two servers (we omit the distribution of PCP, as it is similar to BFD). As shown in the histograms, the proposed solution uses the lower frequency levels more frequently. Moreover, the proposed solution provides a drastic reduction of the violations (i.e., 15.6%) compared to the other approaches. Note that we allocated VMs based on their peak utilizations, which were predicted from their history. Despite the provision based on the peak utilization, we observed quality degradation over the three approaches due to the mispredictions of the peak utilization, especially during abrupt workload changes. However, the proposed solution can statistically reduce the probability of the violation by co-locating uncorrelated

VMs. Thus, the probability of joint under-predictions among the co-located VMs is drastically decreased. To further investigate the effectiveness of the proposed solution, we also simulated the case of servers using dynamic v/f scaling. To prevent frequent oscillations of v/f level (which affects server reliability [17]), we performed the v/f scaling at every 12 samples (i.e., 1 min). As shown in Table 5.1(b), the power savings become smaller compared to the static v/f scaling because the other approaches also adaptively scale v/f level according to the time-varying utilization demand. However, the amount of the violations is unacceptably high in the other approaches. Thus, more servers need to be activated to achieve the same QoS level obtained by the proposed solution, which leads to higher power consumption.

6. CONCLUSIONS

With the changes introduced in D2.3 – Server Multi-Level SW Management, the optimization algorithm for VM allocation was able to get information about the complete power chain, by communicating with the power devices installed in the DC (such as intelligent Power Distribution Units (PDUs), Uninterruptible Power supplies (UPS) or the smart current sensors developed in D2.1), through a dedicated module that provides a clean interface: the Rack Controller. In this deliverable, the functionality of the Rack controller has been enhanced with the vCenter connector (described in *D3.9-Virtualisation of IT tasks)*, which is able to communicate and control the virtual management tool. This deliverable also describes how the optimization algorithm has been improved so that it can optimize the IT power consumption at a rack level, using the information coming from the forecasting algorithms explained in D2.5 – Forecasting Algorithms for Energy Consumption.

Regarding the optimization algorithm, we have presented a novel dynamic power management solution for datacenters targeting the execution of scale-out applications by jointly harnessing server consolidation and v/f scaling, in order to reduce the global power consumption while satisfying QoS requirements. We have first analyzed the characteristics of scale-out applications and evaluated three fundamental approaches for datacenter dynamic power management solutions: 1) conservative resource provision based on (off-)peak utilization, 2) sharing cores among co-located VMs, and 3) correlation-aware VM placement. As an example, we have then proposed a novel VM placement solution utilizing the new definition of correlation and aggressive-yet-safe v/f scaling. Finally, we have validated the applicability of our proposed correlation-aware scheme with the application of multiple web search clusters in [10] and the utilization traces obtained from real datacenter setups. Our experimental results show that the proposed solution provides up to 13.7% power savings and up to 15.6% improvement of QoS level compared to conventional VM placement solutions.