Noname manuscript No.
(will be inserted by the editor)

Server Consolidation Problems

Lionel Eyraud-Dubois · Adrien Lébre · Patrick Martineau · Ameur Soukhal · Vincent T'kindt · Denis Trystram

the date of receipt and acceptance should be inserted later

Abstract Keywords Scheduling · Virtual Machines · Multicriteria

1 Definition and Model

Consider an off-line scheduling problem, called a *Consolidation Server Problem*, occuring in a computers network where a set of N Virtual Machines (VM), required by *service providers*, have to be scheduled, by *an infrastructure provider*, on M Physical Machines (PM) along a time horizon T. Each VM belongs to a class which defines its requirements in terms of resources: for instance, we can find the class of *small-size database VMs*, the class of *medium-size web server VMs*,... Besides, the number of VMs needed by a service provider may vary over time. Then, the aim is to guarantee to the service providers that their VMs will be available as requested along the time horizon. In the remainder, the PM will be referred to as the *machines*. Each machine j is defined by, at any time t, a maximum acceptable load m_j^c of CPU, a maximum acceptable load m_j^g of GPU, a quantity m_j^r (Mb) of RAM and a quantity m_j^h (Mb) of memory space on harddrives. Besides, for any machine j, let us define by α_j^c the cost for using CPUs, α_j^g the cost for using GPUs, α_j^r the cost for using one Mb of RAM, α_j^h the cost for using one Mb of harddrive. At last, we refer to β_t as the cost of using a machine at time t: it is assumed that if a machine does not process any VM at time t, then it is turned off and, so, does not cost β_t. **[Remarque de Patrick : Il faudra qu'on prenne le temps de détailler l'architecture physique d'un cluster de calcul type "Blade" de Bull. Sans remettre en cause le modèle, cela nous permettrait de mieux connaitre les approximations qui sont faites.]** All machines are connected by means of a network which enables data flows to be exchanged between VMs assigned to machines and we assume that this

Address(es) of author(s) should be given

network is located inside the infrastructure provider. Formally, this network can be modelled by an outtree $G = (V, E)$ where each vertex $v_\ell \in V$ is either a machine or a switch. There is an edge $e_{\ell,\ell'} \in E$ (a link in the network) connecting two vertices v_ℓ and $v_{\ell'}$ if a data flow can be exchanged between them. As the network is located inside the infrastructure provider, it is sufficiently small to assume that transfer times on the network are negligible and the links are all equivalent. Let us defined by b the maximum bandwidth of any link in the network. Notice that, unlike for the CPUs, GPUs, RAM and harddrive resources, the maximum bandwidth capacities of both the VMs and PMs is not directly modeled. They are taken into account when defining the value of b which is, then, defined as the minimum value of the real value of the maximum bandwidth of a link and the value of the maximum bandwidths of PMs and VMs. This way to proceed enables us to simply model various configurations of the network infrastructure as, for instance, when the PMs have a maximum bandwidth lower than the one of the network. Additionaly, we refer to $\mathcal{P}$ as the set of paths $P_{j,j'}$ between two machines j and j', each path being defined by a set of edges. Notice that an edge can belong to several paths and that b is also the maximum bandwidth which can be booked along any path $P_{j,j'}$.

Now let us turn to the Virtual Machines. Each VM is considered as a *task*, and all VMs of the same class are all equivalent in terms of resource requirements provided by the machines: the notion of class is taken into account when establishing the requirements of the tasks. Accordingly, each task i to be scheduled is defined by its requirements in terms of resources: a required load n_i^c of CPUs, a required load n_i^g of GPUs, a quantity n_i^r of RAM and a quantity n_i^h of harddrive. A task need to be scheduled only at certain times since the needs of the users may vary over time: therefore we define by $u_{i,t} = 1$ if task i is likely to be processed at time t and 0 otherwise.
Noteworthy, the processing of a task i may be strongly related to the processing of another task k in the sense that non negligibe flows of data can be exchanged between them. In that case, we are talking about the *affinity* between two tasks and it is aimed to assigning them to machines between which the cost of data flows must be minimized. Let be $a_{i,k} = 1$ if there is such an affinity between tasks i and k and let us denoted by $A = (a_{i,k})$ the *matrix of affinities*. If two tasks i and k have an affinity then they require a bandwidth $b_{i,k}$ on the network. Notice that all $a_{i,i}$'s are equal to 1 to enable the migration of tasks through the network. At last, some tasks may be pre-assigned to a specific subset of machines and, thus, only processed by one of that machines: let us denote by $Q = (q_{i,j})$ the *matrix of pre-assignments* with $q_{i,j} = 1$ if task i is pre-assigned on machine j; 0 otherwise. Then, a task i which can be processed by any machine has $q_{i,j} = 1$, $\forall j = 1, ..., M$.
A particularity of the problem is that *reconfiguration* operations may occur: the system allows VMs (tasks) to be preempted, *i.e. suspended* and *resumed* later on the same machine, or even *migrated* from one machine to another but using by the way bandwidth on the network. For each task i we define by $R_i = 1$ if the associated VM is preemptable, and 0 otherwise. Besides, at each

time a preemptable task i is suspended a unitary penalty ρ_i is induced. The migration of a VM can be applied whatever it is preemptable or not since this implies no service interruption. When a task is suspended at a time t then it cannot be resumed before a minimum given time which is given by the time necessary to load in memory its processing context. Let us assigned to each machine j a speed v_j in loading this context, then the total preempted time is at least equal to the resuming time $rt_{i,j} = \frac{n_i^r}{v_j}$ (Figure 1). [**Remarque de Patrick : Si les machines sont homogènes en génération, alors cette vitesse est uniforme. (pas de réseau ? accès disque ?). Que fait-on?**]

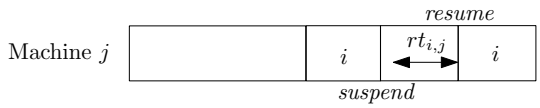

Fig. 1 The operations *suspend* and *resume* of a task i

When a task is migrated from a machine j towards a machine j' then the resuming time on machine j' is assumed to be negligible with respect to the migration time over the network. It is also assumed that the migration operation of any task i requires a fixed amout of bandwidth, *i.e.* $b_{i,i} = b_{i',i'}$, $\forall i, i'$. We assume that, when a task is migrated, it is resumed immediatly after on the target machine. Notice that the data in the harddrive and memory of task i located on the source machine are migrated towards the target machine: thus, enough free space must be available on the target machine to perform the migration. By the technique called *live migration*, the migrated task continues to run on the source machine during the data transfer, after which the it starts running on the target machine (Figure 2). Besides, we assume that on the target machine the quantity of free space must be equal, all along the migration process, to the requirements of task i.

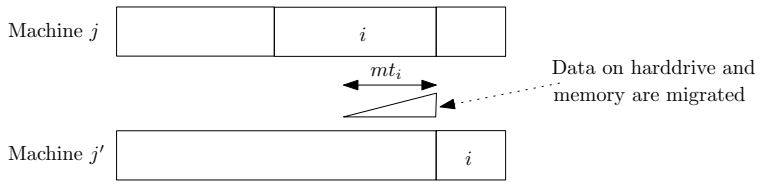

Fig. 2 The *migrate* operation of a task i

The reconfiguration operations have a strong influence on the VM processing as, for instance, a set of preemptable VM having affinities may be suspended on some machines to free bandwidth on the network and make possible migrations. Then, not only the resource constraints may influence the

assignment of VM on machines.

The aim of the *Server Consolidation Problem*, as introduced, is to minimize the total cost for the service provider as well as the time spent in reconfiguration operations. Let us denote by TC the total cost of using the machines and processing VM and by RE the total time spent in reconfiguration operations. The problem to solve is then a bicriteria scheduling problem for which we are looking to Pareto optima formally defined as follows: a schedule S is a Pareto optimum iff there does not exist another schedule S' such that $TC(S') \leq TC(S)$ and $RE(S') \leq RE(S)$ with at least one strict inequality ([4]).

General data:	
T,	the scheduling horizon,
N,	the number of tasks,
M,	the number of machines,
For tasks:	
n_i^c	the required load by task i in terms of CPU,
n_i^g	the required load by task $i$in terms of GPU,
n_i^r,	the quantity of RAM required by task i,
n_i^h,	the harddrive capacity required by task i,
$u_{i,t}$,	a boolean indicating if task i is likely to be processed at time t,
$A = (a_{i,k})_{i,k}$,	the matrix of affinities,
$Q = (q_{i,j})_{i,j}$,	the matrix of pre-assignements,
$b_{i,j}$,	the bandwidth required by two tasks i and j to communicate,
R_i,	a boolean indicating if task i is preemptable or not,
ρ_i,	a unitary penalty induced by suspending task i, if preemptable ($R_i = 1$), 0 otherwise,
mt_i	the migration time of task i, with $mt_i = \lceil \frac{n_i^h + n_i^r}{b_{i,i}} \rceil$,
$rt_{i,j}$	the resume time of task i on machine j, with $rt_{i,j} = \lceil \frac{n_i^r}{v_j} \rceil$,
For machines:	
m_j^c	the maximum accepted load of CPUs of machine j,
m_j^g	the maximum accepted load of GPUs of machine j,
m_j^r,	the quantity of RAM of machine j,
m_j^h,	the harddrive capacity of machine j,
α_j^c,	the cost for using a CPU of machine j,
α_j^g,	the cost for using a GPU of machine j,
α_j^r,	the cost for using one Mb of RAM of machine j,
α_j^h,	the cost for using one Mb of harddrive capacity of machine j,
β_t,	the cost of turning on a machine at time t,
v_j,	the speed of machine j in loading a processing context,
For the network:	
$G = (V, E)$,	the graph modelling the network,
b,	the maximum bandwidth associated to any edge $e_{\ell,\ell'} \in E$,
$\mathcal{P}$,	a set of paths between machines, a path being a set of edges,
$P_{\ell,\ell'}$,	the set of couples of machines (j, j') which use the edge $e_{\ell,\ell'}$,

Table 1 Notations used to define the SCP

2 State of the Art

This section needs to be developped: put references around the Consolidation Server Problem. The references already identified are: [1], [2], [3].

3 Mathematical programming formulation

We provide hereafter a mixed integer programming formulation for the *Consolidation Server Problem* where we assume that all data introduced in Table 1 are given. We focus on a time indexed formulation defined by means of the following boolean decision variables $x_{i,t}^j$:

$$x_{i,t}^j = \begin{cases} 1 & \text{if task } i \text{ is processed on machine } j \text{ in } [t; t+1[, \\ 0 & \text{otherwise} \end{cases}$$

We also make use of a set of instrumental variables. Let be $y_{i,i',t}^{j,j'} = 1$ if tasks i and i' are processed on machines j and j', respectively, at time t and needs to communicate over the network; 0 otherwise. To limit redundancy, we only create variables $y_{i,i',t}^{j,j'}$, $\forall t = 1, ..., T$, for: (i) $\forall i < i'$ and $\forall j < j'$; (ii) $\forall i = i'$, $\forall j \neq j'$. Notice that in case where a task i migrates at time t from machine j towards machine j' then variable $y_{i,i,t}^{j,j'}$ is equal to 1 (but $y_{i,i,t}^{j',j}$ is equal to 0). Notice that we always have $y_{i,i',t}^{j,j} = 0$.
Let be $z_{t,j} = 1$ if machine j processes at least one task at time t; 0 otherwise. Similarly, z_t is the number of machines which are turned on at time t, *i.e.* which process at least one task.
At last, we denote by $d_{i,t}$ the duration of reconfiguration operation of task i at time t: it is either the duration of a resume operation starting at time t or the duration of a migration operation ending at time t.

The *Server Consolidation Problem* can be modeled as follows.

$$\text{Minimize } TC = \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M}\Big(x_{i,t}^j(\alpha_j^c n_i^c + \alpha_j^g n_i^g + \alpha_j^h n_i^h + \alpha_j^r n_i^r)\Big) +$$
$$\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j,k=1,k\neq j}^{M}\Big(y_{i,i,t}^{j,k}(\alpha_k^h n_i^h + \alpha_k^r n_i^r)\Big) + \sum_{t=1}^{T}\sum_{i=1}^{N}(1 - \sum_{j=1}^{M} x_{i,t}^j)\rho_i u_{i,t} + \sum_{t=1}^{T}\beta_t z_t$$
and
$$\text{Minimize } RE = \sum_{i=1}^{N}\sum_{t=1}^{T} d_{i,t}$$
subject to

$$\sum_{i=1}^{N} n_i^c x_{i,t}^j \leq m_j^c \qquad \forall t = 1, ..., T, \forall j = 1, ..., M \qquad \text{(A)}$$

$$\sum_{i=1}^{N} n_i^g x_{i,t}^j \leq m_j^g \qquad \forall t = 1, ..., T, \forall j = 1, ..., M \qquad \text{(B)}$$

$$\sum_{i=1}^{N} n_i^h x_{i,t}^j + \sum_{k=1, k\neq j}^{M} n_i^h y_{i,i,t}^{k,j} \leq m_j^h \qquad \forall t = 1, ..., T, \forall j = 1, ..., M \qquad \text{(C)}$$

$$\sum_{i=1}^{N} n_i^r x_{i,t}^j + \sum_{k=1,k\neq j}^{M} n_i^r y_{i,i,t}^{k,j} \leq m_j^r \qquad \forall t = 1,...,T, \forall j = 1,...,M \qquad \text{(D)}$$

$$y_{i,i',t}^{j,j'} \geq x_{i,t}^j + x_{i',t}^{j'} - 1 \qquad \forall t = 1,...,T,\ \forall i = 1,...,N, \quad \forall i' = i+1,...,N \text{ such that } a_{i,i'} = 1,\ \forall j = 1,...,M, \forall j' = j+1,...,M \qquad \text{(E)}$$

$$y_{i,i,t}^{j,j'} \geq (x_{i,t_1}^j - \sum_{k=1}^{M} \sum_{t'=t_1+1}^{t_2-1} x_{i,t'}^k + x_{i,t_2}^{j'} - 1) \qquad \forall i = 1,...,N, \quad \forall j, j' = 1,...,M, j \neq j', \quad \forall t = 1,...,T,\ \forall t_1 = t,...,\min(t + mt_i, T), \quad \forall t_2 = t_1+1,...,T \qquad \text{(F)}$$

$$mt_i(x_{i,t}^j - \sum_{k=1}^{M} \sum_{t'=t+1}^{t_1-1} x_{i,t'}^k + x_{i,t_1}^{j'} - 1) \leq \sum_{t'=\min(t+1-mt_i,0)}^{t} x_{i,t}^j \qquad \forall i = 1,...,N,\ \forall j, j' = 1,...,M, j \neq j',\ \forall t = 1,...,T,\ \forall t_1 = t+1,...,T \qquad \text{(F')}$$

$$\sum_{j=1}^{M} x_{i,t}^j = u_{i,t} \qquad \forall i = 1,...,N, \text{ such that } R_i = 0, \forall t = 1,...,T, \qquad \text{(G)}$$

$$\sum_{j=1}^{M} x_{i,t}^j \leq u_{i,t} \qquad \forall i = 1,...,N, \text{ such that } R_i = 1, \forall t = 1,...,T, \qquad \text{(H)}$$

$$x_{i,t}^j \leq u_{i,t} q_{i,j} \qquad \forall t = 1,...,T,\ \forall j = 1,...,M,\ \forall i = 1,...,N, \qquad \text{(I)}$$

$$\sum_{(j,j')\in P_{\ell,\ell'}, j<j'} \Big(\sum_{i,i'=1, i<i', a_{i,i'}=1}^{N} b_{i,i'} y_{i,i',t}^{j,j'} + \sum_{i=1}^{N} b_{i,i}(y_{i,i,t}^{j,j'} + y_{i,i,t}^{j',j}) \Big) \leq b \qquad \forall t = 1,...,T,\ \forall e_{\ell,\ell'} \in E \qquad \text{(J)}$$

$$z_{t,j} \geq x_{i,t}^j \qquad \forall t = 1,...,T, \forall i = 1,...,N, \forall j = 1,...,M \qquad \text{(K)}$$

$$z_t = \sum_{j=1}^{M} z_{t,j} \qquad \forall t = 1,...,T \qquad \text{(L)}$$

$$x_{i,t_2}^j \leq 1 - x_{i,t_1}^j + x_{i,t_1+1}^j \qquad \forall t_1 = 1,...,(T - rt_{i,j}), \quad \forall t_2 = t_1+1,...,t_1 + rt_{i,j},\ \forall j = 1,...,M,\ \forall i = 1,...,N \text{ such that } R_i = 1 \qquad \text{(M)}$$

$$d_{i,t_1} \geq (t_2 - t_1)(x^j_{i,t_1-1} - \sum_{k=1}^{M} \sum_{t=t_1}^{t_2-1} x^k_{i,t} - 1 + x^j_{i,t_2}) \quad \forall t_1 = 1, ..., (T - rt_{i,j} - 1), \ \forall t_2 = t_1 + rt_{i,j} + 1, ..., T, \ \forall j = 1, ..., M, \ \forall i = 1, ..., N \text{ such that } R_i = 1 \quad \text{(N)}$$

$$d_{i,t_1} \geq mt_i(x^j_{i,t_1} + x^{j'}_{i,t_1+1} - 1) \quad \forall t_1 = 1, ..., (T - mt_i) - 1), \ t_2 = t_1 + mt_i + 1, \ \forall j, k = 1, ..., M, \ j \neq k, \ \forall i = 1, ..., N \quad \text{(O)}$$

$x^j_{i,t} \in \{0; 1\}$, $y^{j,j'}_{i,i',t} \in \{0; 1\}$, $z_{t,j} \in \{0; 1\}$, $z_t \geq 0$, $d_{i,t} \geq 0$

Constraints (A) guarantee that for each machine no more than the total capacity of the CPUs is used whilst constraints (B) define similarly the constraints on the use of the GPUs. Contraints (C) and (D) completes the bunch of constraints on the resources requirements: contraints (C) (resp. constraints (D)) imply that on each machine no more harddrive (resp. RAM) than available is used either by tasks being process on the machine, or by the tasks being migrated towards the machine. Constraints (E) and (F) define the network use (communications and migrations) at any time point. From constraints (F) we can derive that at a time t a task i is being migrated from machine j to machine j' if there exists a time windows $t \in [t_1 - mt_i; t_1]$ of length mt_i such that the migration starts at time $(t_1 - mt_i)$ and task i changes of machine at time t_1(Figure 3). Constraints (G) imply that at each time point and for each

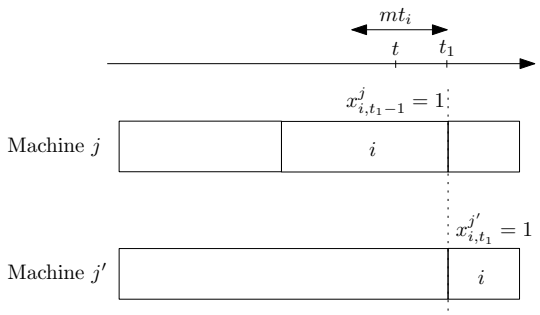

Fig. 3 Illustration of constraints (F)

non preemptable task it must be processed on a machine even if it is being migrated through the network. Constraints (H) impose that a preemptable task may be interrupted even if planned to be processed. Constraints (I) are the pre-assignment constraints. Then, constraints (J) define the bandwidth

capacity constraint that must be answered by both communications and migrations. Constraints (K) and (L) define the number of machines which are turned on at any time t. Constraints (M) define the minimum time to answer before a suspended task can be resumed. From that constraints we can derive that between the suspend operation done at time $(t_1 + 1)$ and the resume at time t_2, a preemptable task i on machine j must not be processed and we must have at least $(t_2 - t_1) = rt_{i,j} + 1$ (Figure 4). Constraints (N) define the

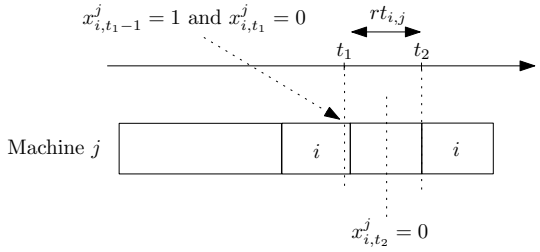

Fig. 4 Illustration of constraints (M)

duration of *resume* operations started at time t_1 (Figure 5) and constraints (O) the duration of *migrate* operations completed at time $(t_1 + 1)$ on any task i.

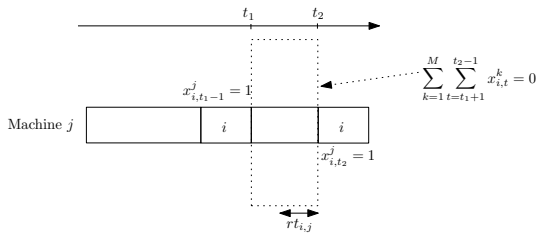

Fig. 5 Illustration of constraints (N)

4 Computational experiments

In order to evaluate the efficiency of the mathematical formulation we have tested it on a bunch of randomly generated instances. Before providing the obtained results, we first describe the way the instances are generated.

The random generation of instances is drived by the wish of being as close as possible to real-life situations. As a starting point, we have collected a set of information about *Amazon Web Service* (AWS) infrastructure (see `huanliu.`

wordpress.com/2012/03/13/amazon-data-center-size/) and the Types of Virtual Machines proposed by Amazon (see aws.amazon.com/fr/ec2/).

The Types of Virtual Machines is synthetized in Table 2 with the values of the expected data in the IP formulation provided. The first column indicates the name of the type of VMs (with into parenthesis the acronym used later on in this paper). The second column contains the CPU requirements in Ghz. Notice that this requirement, for any type of Virtual Machines, is provided by Amazon as a number of EC2 units and a number of cores. As one EC2 unit is equivalent to a requirement of 1.0-1.2Ghz of a core, then we translated directly in that table the required amount of CPU Ghz by each type of Virtual Machines; we consider that one EC2 unit is equivalent to a requirement of 1.0 Ghz. For instance, VMs of type XL requires 2 EC2 units and 4 virtual cores which yields to a global 2×4 Ghz$= 8$ Ghz requirement. The third column contains the number of required GPU. The fourth column contains the quantity of RAM required in Gb, multiplied by 10: thus, a value of 17 indicates a real requirement of 1.7 Gb of RAM. We did this in order to get integer values. The fifth column contains the quantity of harddrive required in Gb, divided by 10: thus, a value of 16 indicates a real requirement of 160 Gb. At last, the sixth column indicates the bandwidth allocated to a VM when needing to comunicate over the network. The maximum bandwidth of any link in the network is equal to 10 Gb: in order to get integer values for the bandwidth allocated to VMs, we multiply this value by 10. Then, a value of 20 in the sixth column indicates that the corresponding type of VMs requires 2 Gb on the network. If two VMs i and j of different types have affinities (and thus need to comunicate) we assume that the corresponding value of $b_{i,j}$ is equal to the minimum value between the allocated bandwidth of i and j. We also make the assumption that the bandwidth allocated for a migration is equal to 10.

Name	CPU (Ghz) n_i^c	GPU (Number) n_i^g	RAM (/10 Gb) n_i^r	HDD (x10 Gb) n_i^h	Bdw (/10 Gb)
Standard Virtual Machines					
Small (S)	1	0	17	16	10
Intermediate (I)	2	0	37	41	10
Large (L)	4	0	75	85	20
Extra large (XL)	8	0	150	196	20
Virtual Machines with High Memory requirement					
Extra large (XLM)	7	0	171	42	10
Double Extra large (DXLM)	13	0	342	85	20
Quad Extra large (QXLM)	26	0	684	196	20
Virtual Machines with High CPU requirement					
Medium (MC)	5	0	17	35	10
Extra large (XLC)	20	0	70	196	20
Cluster Compute Virtual Machines					
Quad Extra large (QXLCC)	36	0	230	196	30
Oct Extra large (OXLCC)	88	0	605	337	30
Cluster GPU Virtual Machines					
Quad Extra large (QXLG)	34	2	220	196	30

Table 2 Types of generated Virtual Machines

Concerning the physical machines, we elaborate on the assumptions raised on `huanliu.wordpress.com/2012/03/13/amazon-data-center-size/` to define a set of physical machines similar to that of the Amazon's cloud. Those machines are servers from the HP Proliant series as synthetized in Table 3. The first column indicates the series' name. The second column indicates the total available CPU capacity in Ghz which is computed as in Table 2: a machine with 4 processors, 16 cores per processor with a frequency of 2.3Ghz lead to a total capacity of 147Ghz. The third column contains the number of GPU. The fourth column contains the availabe RAM multiplied by 10, to be consistent with the values given in Table 2 for Virtual Machines. Similarly, the fifth column contains the available harddrive capacity divided by 10. At last, the sixth column contains the speed of the processor when resuming a suspended Virtual Machines. We have assumed that this speed is dependent from the CPU speed and in the same order of values than the bandwidth allocated to the Virtual Machine to communicate over the network.

Name	CPU (Ghz) m_j^c	GPU (Number) m_j^g	RAM (/10 Gb) m_j^r	HDD (x10 Gb) m_j^h	Speed Cont. v_j
DL380pG8 (D3)	12	0	160	164	10
DL585G7 Conf. 1 (D51)	147	0	1280	491	20
DL585G7 Conf. 2 (D52)	333	0	2560	614	30
DL585G7 Conf. 3 (D53)	147	6	1280	614	20

Table 3 Types of generated Physical Machines

Regarding the costs of the physical machines' usage, Table 4 provides the considered values. They have been built by taking account the seeling price of each kind of resource (CPU, GPU, RAM, HDD). These values have to be read by comparison to each other: for instance, for physical machine D3 the costs of using, for a given time unit, the CPU is ten times coster than storing data in the RAM. We set the cost β_t of turning on a machine at time t equal to 14.

Name	α_j^c	α_j^g	α_j^r	α_j^h
DL380pG8 (D3)	10	0	1	7
DL585G7 Conf. 1 (D51)	14	0	8	22
DL585G7 Conf. 2 (D52)	34	0	17	22
DL585G7 Conf. 3 (D53)	14	18	8	22

Table 4 Costs of using the Physical Machines

We now turn to the way instances are randomly generated.
First, regarding the number N of Virtual Machines and M the number of Physical Machines, we make the assumption that the ratio $\frac{N}{M} = 6$. Conversely, not all Physical Machines can process all Virtual Machines: for instance, an

OXLCC Virtual Machine cannot be processed by a D3 Physical Machine. Therefore we present, in Table 5, the preassignements $q_{i,j}$ where a cell with value 1 indicates that the corresponding Virtual Machine (row) can be processed by the Physical Machine (column).

$q_{i,j}$	D3	D51	D52	D53
S	1	1	1	1
I	1	1	1	1
L	1	1	1	1
XL	0	1	1	1
XLM	0	1	1	1
DXLM	0	1	1	1
QXLM	0	1	1	1
MC	1	1	1	1
XLC	0	1	1	1
QXLCC	0	1	1	1
OXLCC	0	1	1	1
QXLG	0	0	0	1

Table 5 Preassignements of VMs to PMs

The different scenarii considered for Virtual Machines and Physical Machines generation are given in Table 6. To each scenario is defined:

- N_1: the number of Standard Virtual Machines, randomly selected between S, I, L and XL,
- N_2: the number of Virtual Machines with High Memory requirement, randomly selected between XLM, DXLM and QXLM,
- N_3: the number of Virtual Machines with High CPU requirement, randomly selected between MC and XLC,
- N_4: the number of Cluster Compute Virtual Machines, randomly selected between QXLCC and OXLCC,
- N_5: the number of Cluster GPU Virtual Machines (QXLG),
- M_1: the number of $D3$ Physical Machines,
- M_2: the number of $D51$ Physical Machines,
- M_3: the number of $D52$ Physical Machines,
- M_4: the number of $D53$ Physical Machines.

Therefore, we have $N = N_1+N_2+N_3+N_4+N_5$ and $M = M_1+M_2+M_3+M_4$.

Scenario	N_1	N_2	N_3	N_4	N_5	M_1	M_2	M_3	M_4	N	M
Sc1	5	3	0	0	0	1	1	0	0	8	2
Sc2	6	5	0	0	0	2	1	0	0	11	3
Sc3	6	4	2	2	1	2	0	1	1	15	4
Sc4	6	5	3	2	2	2	1	1	1	18	5
Sc5	9	5	4	2	1	2	0	2	1	21	5
Sc6	7	5	4	5	3	2	1	2	1	24	6

Table 6 Different scenarii for the generation of Virtual Machines and Physical Machines

For each scenario, we randomly generate 20 instances. The time horizon T is equivalent to 1 hour discretized by steps of 5mn, leading to a value of $T = 12$. As the banwidth indicated in Table 2 and the speed of loading a context in Table 3 are expressed in Gb per second, they are translated into GB per

slot of 5mn: thus, they are multiplied by 300. As a consequence, the resuming time $rt_{i,j}$ and the migration times mt_i are all equal to 1. The unitary penalty ρ_i is drawn at random in the interval $[1000; 22000]$. For each Virtual Machine we also randomly determine its value $R_i \in \{0; 1\}$. The matrix of affinities is determined as follows: each couple (i, i') of Virtual Machines has a probability of 20% of having an affinity. The values $u_{i,t}$ are determine in a similar way: each value $u_{i,t}$ has a probability of 80% of being equal to 1.

The last part which needs to be described is the way the network is generated. This is easily done by generating a binary tree in the following way: at the root node, we generate a switch. Its left child node in the network is one of the Physical Machine (randomly selected), whilst its right child node is another switch. This process is repeated with the right child node, generating at each level a left child node correspondint to a Physical Machine and a right child node corresponding to another switch. This is done until all Physical Machines are placed in the binary tree (network). The maximum bandiwidth of the network is assumed to be equal to 1Gb which we multiply by 100 to be consistent with the bandwidth requirements of the VMs. The bandwidth allocated for a migration is equal to 10.

The testings have been done a PC with an Intel i5-2540M CPU with 4 cores at 2.6Ghz and 8 Gb of RAM. The IP formulation has been solved by CPLEX 12.2 with parallel solve (3 threads) and a time limit of 1800s and a memory limit of 1Gb of RAM. When one of these two limits is reached, ona given instance, this one is declared *unsolved* and CPLEX may return its best solution, if one has been found.
Table 7 presents, for each scenario, the number of instances found to be infeasible by CPLEX (column $\#InFeas$), the number of instances solved to optimality (column $\#Solved$), the number of instances on which CPLEX stops due to the memory limit (column Mem), and the number of instances on which CPLEX stops due to the time limit (column Tim). For these two columns, the instances may be known to be feasible if CPLEX has found a feasible solution. The results show that for scenarii Sc1 and Sc2, CPLEX was able to solve all the instances whilst for Sc3 only one instance was unsolvable due to the time limit. Scenario Sc4 seems to be of an average difficulty since only 7 instances out of 20 were not solved. This is not the case for scenarii Sc5 et Sc6 which are hard for the solver since, respectively, 15 and 17 instances were not solved. It is important to notice that despite the large size of the IP formulation (as expected due to the high number of variables and constraints), only one instance out of 120 leads to exceeding the memory allocated to the solver (1Gb).

Table 8 presents the complementary results. The three first columns, N_{min}, N_{avg} and N_{max}, provide the minimum, average and maximum number of nodes explored by CPLEX in its Branch& Cut algorithm while solving the problem. The fourth to sixth columns, T_{min}, T_{avg} and T_{max}, provide the minimum, average and maximum CPU time (in seconds) taken by CPLEX to solve instances. Notice that, here, we do not take account of those instances for which either

	$\#InFeas$	$\#Solved$	Mem	Tim
Sc1	2	18	0	0
Sc2	9	11	0	0
Sc3	1	18	0	1
Sc4	0	13	0	7
Sc5	4	1	1	14
Sc6	2	1	0	17

Table 7 Computational results - part 1 (20 instances per scenario)

the time or memory limits are reached: only the instances shown to be infeasible or solved to optimality are considered. The three last columns,M_{min}, M_{avg} and M_{max}, provide the minimum, average and maximum number of Physical Machines used in any solution returned by CPLEX: we consider both instances solved to optimality and instances for which a limit is reached but a feasible solution has been found (the best solution found by the solver).
The results show that, concerning the CPU times, there is large disparity since some instances can be solved quickly (most of the time when the instance in infeasible) whilst other take time. Despite the reasonnable average CPU times reported in that table, scenarii Sc5 and Sc6 seem to be intractable for most of the instances. At last, regarding the number of Physical Machines used in the solution returned by CPLEX, it is most of the time very close to the number of available machines in the instances.

	N_{min}	N_{avg}	N_{max}	T_{min}	T_{avg}	T_{max}	M_{min}	M_{avg}	M_{max}
Sc1	0	6.70	103	0.00	0.15	1.00	1.67	1.86	2.00
Sc2	0	823.5.50	6109	0.00	3.75	28.00	2.50	2.75	3.00
Sc3	0	2070.101	12734	1.00	21.33	88.00	3.06	3.65	3.92
Sc4	1684	16049.53	53160	44.00	421.62	1003.00	3.83	4.51	5.00
Sc5	0	5132.00	20528	0.00	340.75	1361.00	4.50	4.82	5.00
Sc6	0	4596.00	13788	1.00	537.33	1597.00	4.67	5.64	6.00

Table 8 Computational results - part 2 (20 instances per scenario)

5 The on-line version of the SCP

The on-line version of the SCP can be introduced as follows. Assume we are given a schedule s of the tasks on the physical machines (for example, computed by solution of the off-line problem). At any unknown time t a set of unforecasted events when computing s, can occur. These events imply a rescheduling of the tasks on the machines and can be:

- *Arrival of a new task*. A new task i arrives and has to be scheduled,
- *Evolution of an existing task*. One or several characteristics of a task i vary:
  - *Disappear*: task i has no longer to be processed,
  - *Profil modification*: the values of the $u_{i,t}$'s change,
  - *Resource requirement modification*: the resource requirements (n_i^c, n_i^g, n_i^r, n_i^h) change,

- *Machine breakdown.* A machine j is no longer available, and all the assigned tasks have to be reallocated.
- *Network failure.* Some links in the network become unavailable, disabling some paths.

6 Particular subproblems of the SCP

The aim of this section is to highlight particular subproblems of the SCP.

6.1 The Fixed Independents Task SCP with Instantatenous Migrations (FIT-IM)

We assume that we have only CPU as ressources on the machines and tasks. No preemptable task ($R_i = 0, \forall i$). No affinity ($a_{i,j} = 0, \forall i, j$). No pre-assignement ($q_{i,j} = 1, \forall i, j$). All $\beta_t = 0$. The duration of the migration The aim is to schedule tasks in order to minimize the number of migrations (*e.g.* $M_1 = \sum_i \#Migration_i$, $M_2 = \max_i \#Migration_i$ or $M_3 = \max_t \#Migration_{t,t+1}$ with $\#Migration_i$ the number of migrations for task i and $\#Migration_{t,t+1}$ the number of migrations on all the machines between t and $t+1$) and the number of tasks that cannot be scheduled at each time t.
Criterion M_3 is a criterion that has already been used in the literature. Criterion M_1 is "good for the system" (to clarify).

Note: We should be able to reduce the $\epsilon(Migration/Rejectedtasks)$ problem with the number of rejected tasks fixed to a (MCP) and thus derive an exponential algorithm (Sort & Search in $O^*(\sqrt{m+1}^n \frac{n}{2}^{m+2})$?). As the maximum number of rejected tasks is polynomial then this complexity results would also be true for the problem of enumerating the strict Pareto optima.

Todo:

- Modify the problem definition such that the packing subproblem is polynomial (packing means, for a given set of tasks, finding a feasible assignment is polynomial)... all $m_j^c = m, \forall j$?

6.1.1 The Time-Indexed Formulation

For the FIT-IM, we can derive a time-indexed formulation in the line of the one introduced in section 1.

$$\text{Minimize } TC = \sum_{i=1}^{N} \sum_{t=1}^{T} (u_{i,t} - \sum_{j=1}^{M} x_{i,t}^{j})$$

and

$$\text{Minimize } RE = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{j,j'=1}^{M} y_{i,i,t}^{j,j'}$$

subject to

$$\sum_{i=1}^{N} n_i^c x_{i,t}^j \leq m_j^c \qquad \forall t = 1, ..., T, \forall j = 1, ..., M$$

$$y_{i,i,t}^{j,j'} \geq (x_{i,t}^j - \sum_{k=1}^{M} \sum_{t'=t+1}^{t_2-1} x_{i,t'}^k + x_{i,t_2}^{j'} - 1) \qquad \forall i = 1, ..., N,\ \forall j, j' = 1, ..., M, j \neq j',\ \forall t = 1, ..., T-1,\ \forall t_2 = t+1, ..., T$$

$$\sum_{j=1}^{M} x_{i,t}^j \leq u_{i,t} \qquad \forall i = 1, ..., N,\ \forall t = 1, ..., T,$$

$$x_{i,t}^j \in \{0; 1\},\ y_{i,i',t}^{j,j'} \in \{0; 1\}$$

6.1.2 An exponential algorithm

6.2 The Unitary Fixed Independents Task SCP with Instantatenous Migrations (UFIT-IM)

This problem is a particular subproblem of FIT-IM in which we assume that each task i has $n_i^c = 1$ and all the machines are identical. We are allowed to turn on machines whenever necessary to make a feasible schedule. The aim is to minimize the number of migrations as in FIT-IM under the condition that at each time unit only the minimum number of machines is turned on.

6.3 The Batching Independents Task SCP with Instantatenous Migrations (BIT-IM)

This problem is the same than FIT-IM except that we have no profil $u_{i,t}$ for each task, but each task has a required processing time p_i. Preemption is allowed in addition to migration. Then the aim is to schedule tasks in order to minimize the makespan and the number of migrations.

6.4 The Unitary Time Horizon SCP (UTH)

We assume that machines (resp. tasks) are defined by their CPU, GPU, RAM and harddrive capacity (resp. requirement). We assume that the time horizon $T = 1$ which implies that their is no migration or suspend/resume operation. We only have to satisfy the constraints of affinities on the network capacity. The aim is to minimize the cost and the number of rejected tasks.

This problem is $\mathcal{NP}$-hard since it is a generalization of the bin-packing problem.

6.5 The Unitary Time Horizon SCP (UTH)

7 Conclusions and future directions

Several modifications of the above model can be planned:

1. In practice, tasks which have affinities, if preemptable, must be suspended and resumed at the time and not independently (Lionel),
2. Location constraints (Adrien, Denis): tasks having affinities may be located "not too far" in the network. This notion will be measured by a distance (number of links?) and we should add constraints stating that two tasks with affinities must not be on machines with a distance on the network greater than a given value. That value could depend on the kind of data exchanged between the tasks (remark of Denis).

References

1. Hermenier, F., Demassey, S., Lorca, X.: Bin repacking scheduling in virtualized datacenters. Proceedings of the 17th international conference on Principles and practice of constraint programming (CP'11) pp. 27–41 (2011)
2. Hermenier, F., Lorca, X., Menaud, J.M., Muller, G., Lawall, J.: Entropy: a consolidation manager for clusters. Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual Execution Environments pp. 41–50 (2009)
3. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. Proceedings of the 29th IEEE conference on Computer Communications (INFOCOM 2010) pp. 1–9 (2010)
4. T'kindt, V., Billaut, J.C.: Multicriteria Scheduling: Theory, Models and Algorithms. Springer (2006)