Continuous Resources Allocation in Internet Data Centers

Youssef Hamadi
Microsoft Research Ltd.
7 J J Thomson Avenue
Cambridge CB3 0FB,
United Kingdom
youssefh@microsoft.com

Abstract

Internet data centers (IDCs) perform multi-customer hosting on a virtualized collection of resources while Grid computing generalizes distributed computing by focusing on large scale resource sharing [8]. When we consider the problem of resource allocation, the connection between Grid and IDC is obvious. Indeed, both systems are using similar resource reservation patterns. Mainly, service level agreement for IDC, advanced resource reservation in Grid systems [11]. In each world, those reservation patterns are deemed to precisely represent resources requirements. This work presents an autonomous system for online resource allocations. The simulated architecture represents some IDC but our problem solving concepts are applicable to Grid infrastructures. Our system takes advantage of monitoring information to reconsider its mathematical modelling of the components. This results in a continuous adaptation of the allocated resources.

Keywords: Resource Management and Scheduling

1 Introduction

A data center infrastructure consists of a ”farm” of massively parallel, densely packaged servers interconnected by high-speed, switched LANs. Current data centers contain tens of thousands of servers; projected infrastructures are even larger [2]. Typically, these computing farms have to host a large set of e-commerce applications which raises a set of important issues. Besides management and security considerations we find the important problem of resource allocations. The combinatorial nature of this problem makes it hard to solve. Moreover, hosted customers increasingly require support for peak loads that are orders of magnitude larger than what they experience in their normal state. Thus, a hosting environment needs a fast turnaround time in adjusting the resources (bandwidth, servers, and storage), assigned to each customer.

Our work has two main contributions to solve the previous problem. First of all, it presents what we think is the first Constraint Programming [5] solution for resource allocation in large-scale IDCs. Second, it embeds this solution in an online architecture which can autonomously adapt to its moving environment.

We start from a pre-defined reservation pattern (represented by any SLA or advanced reservation item) which represents the founding of our CP modelling. This pattern is then repaired to cope with observed variations and then to perform on-the-fly adaptation to resource selections. In the following, section two presents some previous work. Section three gives the details of our CP modelling for resource allocation. Then, section four presents the components of the online problem solving architecture. Finally, before giving a general conclusion in section six, experimental results are presented in section five.

2 Related work

The Oceano [7] project has designed and developed a pilot prototype of a scalable, manageable infrastructure for a large scale ”computing utility power plant”. Oceano’s goal was to introduce high levels of automation to dynamically adjust web sites to actual traffic demands over a massively parallel array of shared and distributed servers. Via Oceano a group of servers can be automated to handle the IT needs of many users, including on-the-fly changes in the load requirements. The adaptation levels of Oceano are very close to our proposal. However, Oceano embraces a large scope (redundancy, reliability, etc.) while our work is focused on efficient resource allocation.

The work of [12] presents the allocation of multi-tier e-commerce applications. Authors use mathematical integer programming (MIP) mixed with dedicated heuristics to solve this hard problem. Our approach is much more versatile and adaptive since we perform successive reallocations through monitoring.

[4] discusses several approaches to define and prototype a data model devoted to IDC configuration. Their main objective is to alleviate the management of large scale Internet data center. The authors are interested in the modelling of the information model of these centers. This work is important and helpful to define a realistic CP modelling for this problem.

The goal of the Eole project [9] was to build an online optimization framework dedicated to telecom applications. The framework can consider environmental events, temporal constraints and resource constraints. This work was able to enhance the Quality of Services of network providers, by increasing their overall flexibility.

3 Constraint programming modelling

3.1 Internet data center

A data-center infrastructure consists of a "farm" of massively parallel, densely packaged servers interconnected by high-speed, switched LANs. We present here the formalization of each element of such farm.

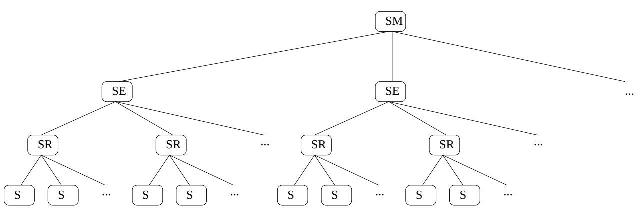

Figure 1. Internet data center topology

Figure 1 presents an abstraction of an IDC. It is composed by a set of interconnected resources (computation nodes and storage nodes) and by networking components (switches, routers, etc). They have a tree-like structure organized in three layers of switches [12].

The switch mesh (SM) is the root of the IDC; this component is connected to the outside world and to a collection of edges switches (SE). These switches are connected to a set of rack switches (SR) that are connected to a set of servers (S). Duplex links are used for the interconnection of the different switches/servers. Each link has a fixed bandwidth limit. The presented topology has three layers but the present work can be generalized to any tree-like architecture[1].

[1]The tree structure gives a unique path between resources; this feature is used for efficient solving.

3.1.1 Topology

We detail here the constraint formalization of the previous architecture. Each resource is represented by its related limitations/capacities.

An IDC is delimited by its size:

- S_e number of SE switches
- S_r number of SR switches per SE switch
- S number of Servers per SR switch

With the previous definitions, the size of an IDC is $S_e \times S_r \times S$.

3.1.2 Switches

The delay for communication in the different switches is ignored . But each switch has some bandwidth limits.

- BS_{m_i}/BS_{m_o} represent the input/output bandwidth limits of the SM switch
- $BS_{e_{i_k}}/BS_{e_{o_k}}$ represent the input/output bandwidth limits of the k^{th} SE switch
- $BS_{r_{i_k}}/BS_{r_{o_k}}$ represent the input/output bandwidth limits of the k^{th} SR switch

3.1.3 Servers

Each server node S_k has several attributes to express its hosting capacities.

- S_{Cpu_k} represents the number of CPUs for the server
- S_{Speed_k} frequency of the server's Cpu(s)[2]
- S_{Mem_k} memory size
- $S_{Storage_k}$ storage capacities
- $S_{DiskSpeed_k}$ hard drive speed
- BSS_{i_k}/BSS_{o_k} represent the input/output bandwidth limits of the server

3.2 Multi-tier application

Figure 2 presents the typical structure of a classical e-commerce application. In these applications, clients send their requests via the Internet. At the top level, some load balancing mechanism routes this traffic to a set of n_1 web servers. Each web server is able to satisfy requests for static resources. If a web server cannot satisfy a request,

[2]In a multi-processor architecture we assume the same speed for each CPU.

it forwards it to one of the n_2 application servers. Application servers run scripts and make use of the n_3 data base servers to support information retrieval, transactional order management and personalization [1]. They prepare html responses, which are addressed to customers via the web servers.

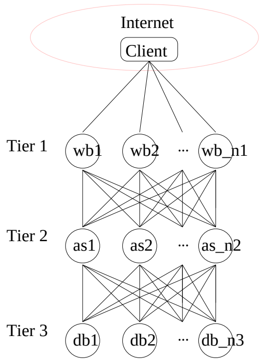

Figure 2. Typical e-commerce application

We model an application A with a graph $A = (X, E)$, where X is the set of processes required by the application ($|X| = n_1 + n_2 + n_3$) and E the set of duplex connection among processes. Each process P_k of the application has a set of lower bounds requirements.

- P_{Cpu_k} represents the number of CPUs required by the process
- P_{Speed_k} frequency of the required CPUs
- P_{Mem_k} size of the required memory
- $P_{Storage_k}$ required storage space
- $P_{DiskSpeed_k}$ hard drive speed

The bandwidth requirements for duplex connections are represented by the following values:

- bc_{01} expresses the required bandwidth between a client and a web server.
- bc_{12}, required bandwidth between a web server and an application server.
- bc_{23}, required bandwidth between an application server and a data base server.

3.3 Resource allocation

From the previous modelling, we can build a constraint programming solution to our problem. We first define a set of constrained variables and then we connect these variables with relevant constraint relations in order to compute correct allocations.

3.3.1 Variables

Before defining the constrained variables of this problem, we need to define some notation (see definition 3.3.1).

DEFINITION 3.3.1
$x : Var : [lb..ub]$, represents a constraint variable x composed by the integer lb to ub.

Internet data center
The SM switch uses two constrained variables to express respectively its input/output bandwidth load.

$S_{m_i} : Var : [0..BS_{m_i}]$, $S_{m_o} : Var : [0..BS_{m_o}]$

The k^{th} SE switch uses two constrained variables to express respectively its input/output bandwidth load.

$S_{e_{i_k}} : Var : [0..BS_{e_{i_k}}]$, $S_{e_{o_k}} : Var : [0..BS_{e_{o_k}}]$

The k^{th} SR switch uses two constrained variables to express respectively its input/output bandwidth load.

$S_{r_{i_k}} : Var : [0..BS_{r_{i_k}}]$, $S_{r_{o_k}} : Var : [0..BS_{r_{o_k}}]$

Each server k uses the following set of constrained variables

- $Process_k : Var : [0..n_1 + n_2 + n_3]$ which represents the identification of the hosted process. Remark that among these $n_1 + n_2 + n_3 + 1$ values, the last one represents a special value expressing that the server is not hosting anything.
- $Tier_{1_k} : Var : [0..1]$ boolean variable set to 1 if the hosted process is part of the first tier of the multi-tiered application, i.e., $Process_k$ between 0 and $n_1 - 1$.
- $Tier_{2_k} : Var : [0..1]$ boolean variable set to 1 if the hosted process is part of the second tier of the multi-tiered application.
- $Tier_{3_k} : Var : [0..1]$ boolean variable set to 1 if the hosted process is part of the third tier of the multi-tiered application.

The application
Each process k of the multi-tiered application has one constrained variable.

- $Server_k : Var : [0..S_e * S_r * S - 1]$ represents the hosting server for the k^{th} process.

3.3.2 Constraints

In order to limit $Process_k$ and $Server_k$ to possible allocation sets we start with a static pruning of their possible values:

$\forall S_k, \forall P_{k'},\ k' \in Process_k$ iff $S_{Cpu_k} \geq P_{Cpu_{k'}}$, $S_{Speed_k} \geq P_{Speed_{k'}}$, $S_{Mem_k} \geq P_{Mem_{k'}}$, $S_{Storage_k} \geq P_{Storage_{k'}}$, $S_{DiskSpeed_k} \geq P_{DiskSpeed_{k'}}$

$\forall P_k, \forall S_{k'},\ k' \in Server_k$ iff $P_{Cpu_k} \leq S_{Cpu_{k'}}$, $P_{Speed_k} \leq S_{Speed_{k'}}$, $P_{Mem_k} \leq S_{Mem_{k'}}$, $P_{Storage_k} \leq S_{Storage_{k'}}$, $P_{DiskSpeed_k} \leq S_{DiskSpeed_{k'}}$

In order to correctly compute the required bandwidth at each server, we define three boolean vectors.

DEFINITION 3.3.2
$tier1[k] = 1$ if P_k is in the first tier of the application, 0 otherwise. $tier2[k] = 1$ if P_k is in the second tier of the application, 0 otherwise. $tier3[k] = 1$ if P_k is in the third tier of the application, 0 otherwise.

These vectors are used to define the values of the $Tier_{i_k}$ variables in relation with the hosted process $Process_k$:

- $element(tier1, Process_k, Tier_{1_k})$
- $element(tier2, Process_k, Tier_{2_k})$
- $element(tier3, Process_k, Tier_{3_k})$

The operational semantic of an $element$ constraint can be seen as an indirection between constrained variables. $element(T, X, Y)$ enforces T[X] = Y. In our case, the $Tier_{i_k}$ variables will receive a correct value 0 or 1 corresponding to the hosted process's tier.

In order to verify that $Process_k = k' \Rightarrow Server_{k'} = k$ we need another $element$ constraint. This time, the vector Tab is made by the set of $Process_k$ variables:

$\forall\ Server_k, element(Tab, Server_k, k)$

Now, since two servers cannot host the same process, we put an $alldiff$ constraint between them. Such a constraint ensures that a set of variables are using different values. However since some server can be unallocated which for us is interpreted by the hosting of the extra process ranked $n_1 + n_2 + n_3$, this peculiar value is not considered by our alldiff.

alldiff$(Process_k)$

To respect the bandwidth limitations of each SR switch, we define the ingoing traffic $S_{r_{i_k}}$ as the traffic addressed by the external processes toward processes hosted[3] by S_{r_k}:

$\forall S_{r_k}, S_{r_{i_k}} = (\sum_{\forall S_{k'} \in S_{r_k}} Tier_{1_{k'}}) \times bc_{01} + (n_1 - \sum_{\forall S_{k'} \in S_{r_k}} Tier_{1_{k'}}) \times (\sum_{\forall S_{k'} \in S_{r_k}} Tier_{2_{k'}}) \times bc_{12} + (n_2 - \sum_{\forall S_{k'} \in S_{r_k}} Tier_{2_{k'}}) \times (\sum_{\forall S_{k'} \in S_{r_k}} Tier_{1_{k'}} + \sum_{\forall S_{k'} \in S_{r_k}} Tier_{3_{k'}}) \times (bc_{12} + bc_{23}) + (n_3 - \sum_{\forall S_{k'} \in S_{r_k}} Tier_{3_{k'}}) \times (\sum_{\forall S_{k'} \in S_{r_k}} Tier_{2_{k'}}) \times bc_{23}$

The previous equation is decomposed in four products. The first one represents the traffic routed from the Internet clients toward the hosted web servers. The second represents the traffic routed from the external web servers, i.e., the web server not hosted by the switch; toward the hosted application servers. The third product represents the traffic incoming from the external application servers to the hosted web servers and data base servers. Finally, the fourth product represents the traffic upcoming from the external data base servers toward the hosted application servers. Since we assumed a symmetric bandwidth between related tiers, $S_{r_{i_k}} = S_{r_{o_k}}$.

Bandwidth limitations at the edge level (SE) are expressed similarly, $S_{e_{i_k}} = S_{e_{o_k}}$:

$\forall S_{e_k}, S_{e_{i_k}} = (\sum_{\forall S_{k'} \in S_{e_k}} Tier_{1_{k'}}) \times bc_{01} + (n_1 - \sum_{\forall S_{k'} \in S_{e_k}} Tier_{1_{k'}}) \times (\sum_{\forall S_{k'} \in S_{e_k}} Tier_{2_{k'}}) \times bc_{12} + (n_2 - \sum_{\forall S_{k'} \in S_{e_k}} Tier_{2_{k'}}) \times (\sum_{\forall S_{k'} \in S_{e_k}} Tier_{1_{k'}} + \sum_{\forall S_{k'} \in S_{e_k}} Tier_{3_{k'}}) \times (bc_{12} + bc_{23}) + (n_3 - \sum_{\forall S_{k'} \in S_{e_k}} Tier_{3_{k'}}) \times (\sum_{\forall S_{k'} \in S_{e_k}} Tier_{2_{k'}}) \times bc_{23}$

We use a similar formulation for the SM switch.

3.3.3 Optimization

The previous set of equations gives correct solutions for the allocation of a multi-tiered application in an IDC. However, the hierarchical structure of the hosting infrastructure allows us to distinguish between these solutions. The optimal solution to this resource allocation problem minimizes communication latency. We express this optimization function with the following constraint:

$$min(\sum_{\forall Server_k, Server_{k'}} dist(k, k') \times band(k, k'))$$

In the previous equation, $dist(k, k')$ represents the distance (links) within the IDC (2, 4, or 6) between the two servers hosting processes k and k'. This value is weighted by the required bandwidth (bc_{12} or bc_{23}). The solver will have to minimize the previous cost function to find out the optimal allocation. A lower bound can easily be computed

[3] A switch "hosts" the processes hosted in the switch's subtree.

by relaxing bandwidth constraints. The obtained value represents the weighted addition of minimal distances between communicating processes.

3.3.4 Breaking symmetrical solutions

The high level of symmetries occurring in the network infrastructure and within the applications raises a large set of equivalent solutions. If we consider that the size of the search space is $O((n_1+n_2+n_3)^{S_e \times S_r \times S})$, it becomes crucial to remove symmetries. We decided to remove the vast majority of symmetries by the addition of some arithmetic constraints.

At the infrastructure level

We can break symmetries within each S_{r_k} switch iff the following proposition (checked after the initial filtering) holds:

PROPERTY 3.3.1
$\forall S_i, S_j \in S_{r_k}, Process_i \equiv Process_j$

That means that the two servers are equivalent, i.e., they can host the same subset of processes. Moreover, since they are connected to the same SR switch any combination of hosting between them has the same impact on the cost function (see 3.3.3). E.g., $(Process_i = a, Process_j = b) \equiv (Process_i = b, Process_j = a)$. In order to break these symmetries, we add the following new constraint when the property holds:

$$InfEqual(Process_i, Process_j)$$

We cannot use a tighter condition (Inf, i.e., $<$ instead of $\leq$) since IDC's servers express their availability by hosting a fake process (see above).

Between SE switches (and within the SM switch) the calculation of an equivalence condition between servers becomes harder since their routing costs are different.

At the application level

Within each tier of the application, the processes are equivalent. We can directly remove some symmetry with the following constraint applied to any equivalent pair of processes P_i and P_j:

$$Inf(Server_i, Server_j)$$

4 Online Optimization in Internet Data Centers

The traffic of a given application can greatly vary over time and similarly, the IDC topology can vary according to failures or to maintenance operations. In this section we first show how to integrate these variations in our CP modelling. We then detail an online optimization architecture which constantly monitor the environment and efficiently update the solution.

4.1 Changing traffic

E-commerce applications usually experience very large traffic variations. In figure 3, we present the annual day-load of a successful commercial website [1]. In this figure, several large variations can be observed. Among these variations, some can be predicted and some cannot. The advertising campaign can be predicted[4]. The typical variation upcoming from classical Christmas and bank holiday can also be predicted. However, the sudden failure of a competitor could report some of its customers to your application. This sudden raise of traffic cannot be predicted.

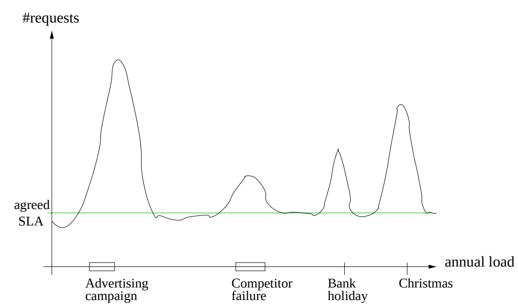

Figure 3. Web site annual load

During these sudden peaks, the current resource allocation which is based on some agreed service level agreement (SLA) will not be able to cope with this new traffic. The obvious solution is to raise the agreed SLA but this over sizing is wasteful.

Another possible variation for an application is the change in traffic classes. This can happen with the use of personalization technology. Initially, the knowledge on a given user is empty. You then provide basic (static) content to this user. With time, you can learn user's profile and then provide dedicated (dynamic) content. The previous changes traffic classes (from static to dynamic) without changing the amount of external traffic. With successful personalization technologies the second and third tiers of your application will become more solicited. As said previously the current allocation based on some agreed SLA ($n2$ and $n3$) could be too weak to support these variations.

[4] Of course you must bet on a bit of communication between marketing and IT.

4.2 Changing infrastructure

Large computing infrastructures like Internet data centers are prone to component failures. Moreover such large systems involve important maintenance and update operations. These breakdowns and updates can greatly jeopardize the hosted applications. An IDC provider needs an efficient mechanism in order to maintain an acceptable service level while performing essential maintenance operations.

4.3 Dynamic Constraint Programming modelling

In this section we show how to extend our initial problem modelling in order to cope with the previous variations. The idea is to take advantage of the versatility of Constraint Programming. Indeed, in this formalism, any problem P can be transformed in a new problem P' by some addition/removal of constraints [6].

4.3.1 Application

To successfully face traffic variations (amount and classes), we start with a modelling which can manage the worst loads. We then select within this large set of components a subset which can satisfy some initial SLA (this set corresponds to the initial allocation). Remaining units will be selected and added to the application with respect to traffic variations. Figure 4 presents these two sets.

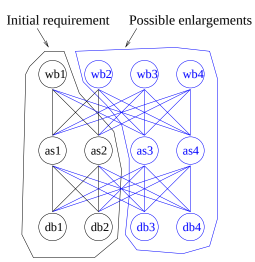

Figure 4. Dynamic modelling of an e-commerce application

In order to disconnect the remaining components from the initial set we use the following constraints:

- $Server_k = S_e \times S_r \times S$, this constraints allocate a fake server[5] to the remaining process k.
- $BSS_{i_k} = 0$, $BSS_{o_k} = 0$, in the previous modelling and for the sake of simplicity these variables were set as constant. In the dynamic modelling we have to use constrained variables to apply constraints on them. The transformation is seamless. Thanks to these two constraints process k has no impact of the cost function.

[5]Remember that allocated servers range from 0 to $S_e \times S_r \times S - 1$

The practical result of these constraints is a "logical" disconnection of the remaining part from the modelling. To add more processes we just have to change the allocated values on the previous constraints. For example to integrate process k:

- $Server_k : Var : [0..S_e * S_r * S - 1]$
- $BSS_{i_k} : Var : [BSS_{i_k}..BSS_{i_k}]$, $BSS_{o_k} : Var : [BSS_{o_k}..BSS_{o_k}]$. These two variables are initialized with the initial requirements for process k.

Thanks to the previous transformations, the CP solver can allocate the correct number of processes. When facing traffic variations our architecture will just have to add/remove components by changing these few constraints.

4.3.2 Infrastructure

Similarly, changes in the IDC topology can be integrated in the CP modelling by addition/removal of constraints. Figure 5 features a small data center where the right part represents a possible extension (second SE switch). Our CP modelling can integrate these resources from the beginning and avoid their allocation thanks to some extra constraints. In case of component failure, constraints can similarly disconnect faulty components.

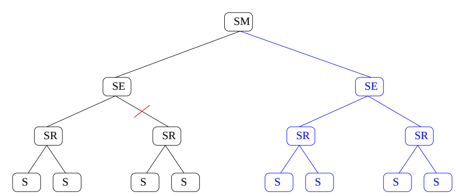

Figure 5. Dynamic modelling of an Internet data center

For instance, in order to disconnect the second SE switch:

$$BS_{e_{i_1}} = 0, BS_{e_{o_1}} = 0$$

These two constraints allocate null bandwidth capacities to the switch. The outcome of that is that related components will not be part of any feasible solution. In the same way, when a failure is detected, the same kind of constraints can be used to disconnect faulty parts. On the figure, the second SR switch becomes deficient. In order to disconnect it we apply the following constraints:

$$BS_{r_{i_1}} = 0, BS_{r_{o_1}} = 0$$

When the damaged component is replaced, the resource allocation can use it again thanks to these two constraints:

$$BS_{r_{i_1}} : Var : [BS_{r_{i_1}}..BS_{r_{i_1}}], BS_{r_{o_1}} : Var : [BS_{r_{o_1}}..BS_{r_{o_1}}]$$

The same mechanism can be applied for any server addition or removal.

4.3.3 Online Optimization

In section 3.3.3 we wanted to minimize communication latency. In this online extension, it is worthwhile to minimize both latency and turnaround time between successive allocations. We can integrate this second goal in a new cost function:

$$min(\omega \times \sum_{\forall Server_k, Server_{k'}} dist(k,k') \times band(k,k') + \omega' \times \sum_{\forall Server_k} dist(k, p(k)))$$

In the previous equation, ω and ω' are the weights associated to latency and to turnaround objectives. Thanks to these weights one can favor one of the criterions. The history is represented by the function $p(k)$ which gives the location of process k in the previous allocation. When k was not part of the previous solution, which occurs when this process is added to cope with increased traffic, the function returns 0.

4.4 Global architecture

We have defined a complete CP solution to solve the problem of resource allocation in IDCs. We have also presented simple extensions to compensate environmental variations. Our extensions use a new objective function which is able to minimize both latency i.e., quality of the allocation, and turnaround, i.e., cost of a reallocation. In this section we present the integration of our work in an online problem solving architecture, figure 6. This architecture takes advantage of successive re-allocations to incrementally raise its performances. It implements two major components.

The management component This component is used to setup applications into the IDC according to search results. Beside that, this unit can use the search capabilities during pre-stage customer negotiation. During SLAs negotiation, the architecture is useful to know about the feasibility/cost of the hosting.

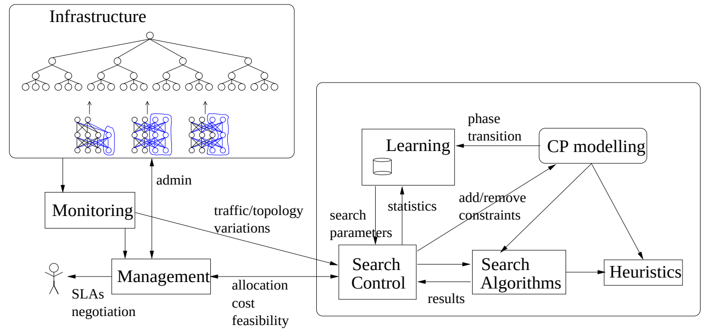

Figure 6. Online optimization architecture

The search components The search control module is principally connected to monitoring and to management. From this latter component, it receives allocation demands and returns information on feasibility (is there a solution?), cost, physical allocation results. The monitoring component addresses this module to report fluctuating traffic and infrastructure variations. This component is able to extend/reduce the amount of allocated resources in response to environmental changes which are directly reported in the CP modelling by simple additions/removals of constraints. Beside these fundamental services, the online situation of our architecture authorizes it to constantly improve its problem solving performances. Indeed, it is connected to a learning component which stores statistics on previous searches. This knowledge can be used to increase the practical performances of future searches.

5 Experiments

Projected data centers will use thousands of servers. However, infrastructures will be logically partitioned in smaller farms with accessible sizes [12]. To achieve these results we used the sequential branch & bound algorithm of [10]. We defined an IDC with 1024, ($S_e = 8, S_r = 8, S = 16$) servers. This IDC was derived in two topologies. The first one called “Symmetric” uses the following bandwidth limitations, $BS_{m_i} = BS_{m_o} = 10, BS_{e_{i_k}} = BS_{e_{o_k}} = 15, BS_{r_{i_k}} = BS_{r_{o_k}} = 25$. The second one called “Asymmetric” uses for each previous parameter a random value between 1 and the previous limitation. This is interesting to simulate allocation in partially loaded IDCs. Indeed, it corresponds to the real situation where e-commerce applications are successively added to a data center [12]. Servers were partitioned in three classes able to host respectively, the web servers, the web servers and the application servers, anything. Their bandwidth limitations were set to $BSS_{i_k} = BSS_{o_k} = 50$. We used several applications figured by their respective characteristics $((n_1, n_2, n_3), (bc_{01}, bc_{12}, bc_{13}))$.

Symmetric IDC				
Application	*cost*	*time(sec.)*	*#btrks*	*#nodes*
((3,1,1),(1,2,2))	24*	50.64	35	16264
((3,1,1),(2,4,4))	48*	51.14	34	16138
((3,2,2),(1,2,2))	84	625.05	273	180728
Asymmetric IDC				
Appli.	*cost*	*time(sec.)*	*#btrks*	*#choices*
((3,1,1),(1,2,2))	24*	34.04	33	11917
((3,1,1),(2,4,4))	48*	4.09	14	1352
((3,2,2),(1,2,2))	84	190.40	224	79433

Table 1. Experimental results

Results are presented in table 1. Optimal cost results are labeled with a "star". Indeed, when the experiments were too time consuming, we decided to stop them roughly after 10 minutes. We can see that the allocation in the "Symmetric" IDC is much harder than in the "Asymmetric" one. Indeed, similar bandwidth capacities raise a high level of symmetry which generates a larger solution space. These symmetries come from the edge and mesh switches and were not removed in our "symmetry break" section. With the asymmetric IDC, the switches are distinguished and some of them cannot route the required traffic. The previous limits the space of feasible solutions. As we can see, the number of nodes developed by the search process in the asymmetric infrastructure is less important than in regular ones. This results implies that the solver was able to prune the space earlier thanks to bandwidth limitations. In regular IDCs the solver had to rely on the cost function to bound the search close to the leaf level.

Interestingly, a more general result comes from these experiments. It concerns the time granularity of our online re-allocation scheme. Indeed, from the previous we can see that the system computes optimal or good allocations within minutes. If we then add the time for resource migrations, we can infer the responsiveness of our architecture to sudden fluctuations in resource demands. Roughly, if we consider that data bases relocations which are the most expensive migrations are achieved within minutes, we can consider that our online architecture can easily handle intra day variations.

6 Conclusion

We have presented a Constraint Programming solution which efficiently allocates multi-tier e-commerce applications in Internet data centers. Our solution was extended to compensate environmental changes. Moreover, we showed how to integrate it in an online problem solving architecture. The experiments demonstrated the feasibility of our approach to tackle very large infrastructures. Our results are competitive with the one from [12]. We learned from these experiments that allocating applications in partially loaded infrastructure was far easier since the impact of previous allocation is to break the high level of symmetry of this problem. As a future work we are planning to apply Constraint Programming to Grid resource sharing [3].

References

[1] M. Arlitt, D. Krishnamurthy, and J. Rolia. Characterizing the scalability of a large web-based shopping system. *ACM Transactions on Internet Technology (TOIT)*, 1(1):44–69, August 2001.

[2] S. Banerjee and X. Zhu. Internet data centers: A survey of key players and market growth. Technical Report 2001-39, Hewlett-Packard lab., 2001.

[3] M. Bartlett, A. M. Frisch, Y. Hamadi, I. Miguel, and C. Unsworth. Effi cient algorithms for selecting advanced reservations. Technical Report 2004-132, Microsoft Research, Dec 2004.

[4] J. L. de Verga, J. Guijarro, P. Goldsack, and C. Todman. Modeling and developing the information to manage an Internet data center. Technical Report 2001-44, Hewlett-Packard lab., 2001.

[5] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[6] R. Dechter and A. Dechter. Belief Maintenance in Dynamic Constraint Networks. In *Proc. National Conference on Artificial Intelligence*, 1988.

[7] T. Eilam. Neptune: A dynamic resource allocation and planning system for a cluster computing utility. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, pages 57–64, May 2002.

[8] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1–??, 2001.

[9] S. Givry, Y. Hamadi, J. Mattioli, P. Gérard, M. Lemaître, G. Verfaillie, A. Aggoun, I. Gouachi, T. Benoist, E. Bourreau, F. Laburthe, P. David, S. Loudni, and S. Bourgault. Towards an on-line optimisation framework. In *CP-2001 Workshop on On-Line combinatorial problem solving and ConstraintProgramming (OLCP'01)*, pages 45–61, Paphos, Cyprus, December 1 2001.

[10] Y. Hamadi. Disolver: A Distributed Constraint Solver. Technical Report 2003-91, Microsoft Research, Dec 2003.

[11] J. MacLaren. Advanced reservation: State of the art. GWD-I, Global Grid Forum (GGF), june 2003.

[12] X. Zhu and S. Singhal. Optimal resource assignment in internet data centers. In *Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems MASCOTS'01*, pages 61–69, August 2001.