Bin Repacking Scheduling in Virtualized Datacenters - Back to Work -

Fabien Hermenier,
OASIS Team, INRIA - CNRS - I3S, Univ. Nice Sophia-Antipolis,
fabien.hermenier@inria.fr

Sophie Demassey,
TASC project, Mines Nantes-INRIA, LINA CNRS UMR 6241,
sophie.demassey@mines-nantes.fr

Xavier Lorca,
TASC project, Mines Nantes-INRIA, LINA CNRS UMR 6241,
xavier.lorca@mines-nantes.fr

September 26, 2011

Because the acceptation of a paper does not necessarily means that its objectives were achieved, we made some progress in the model and the implementation of the VRSP since the acceptation of *Bin Repacking Scheduling in Virtualized Datacenters* [3] in June 2011. In this report, we present our modifications and the new results using the same evaluation protocol.

1 Modifications

Profiling and testing our code in various situations revealed a few flaws which reduced the performance of the solving process and the quality of the computed reconfiguration plans. In this section, we describe the modifications that were made.

In the VRSP. The implementation of the bin-packing constraint in Choco partially relies on set variables. The constraint has been reimplemented to directly maintain the candidate items for a bin as internal data, using bitsets, instead of using the external set abstraction. This reduces the propagation time of the constraint by a significant order of magnitude.

In the side constraints. Profiling the solving process of instances having numerous side constraints shown bottlenecks in `lonely` and `capacity` that were responsible of a high computation time.

The implementation of `lonely` has been modified to substitute the set-based *disjoint* constraint by an equivalent that directly manage two lists of integer variables. As a result, the set variables and the channeling constraints to link the sets with the integer variables have been removed.

The `capacity` constraint has been remodeled to also avoid the use of set variables to store the number of VMs running on servers. `capacity` is now modeled with one *among* [1] constraint that directly counts the VMs assigned to any server in the designated group of servers. For any subset of servers $R \subsetneq \mathcal{R}$ and value $n \in \mathbb{N}$, $n > 0$, $capacity(R, n)$ is modeled by:

$$\texttt{among}([0, n], \langle F_j^r \mid j \in \mathcal{J} \rangle, R).$$

where variable F_j^r models the final server assigned to VM $j \in \mathcal{J}$. In the specific case $n = 0$, constraint $capacity(R, n)$ is instead directly modeled using domain constraints: $F_j^r \notin R, \forall j \in \mathcal{J}$.

In the search heuristics. A bug was found in the search heuristic that prevented it to use a *worst-fit* approach to place VMs on servers. Fixing this bug improves its efficiency to guide the solver to a solution when the consolidation ratio was high. A second optimization was performed to reduce the scheduling delay of the actions and the number of migrations of the reconfiguration plan. In the *repair mode*, the branching heuristic now tries first to place VMs on servers that only host candidate VMs and VMs that will not be suspended nor stopped. No resource will then be freed on these servers during the reconfiguration process so any action that will place a VM on them is ensured to be scheduled without any delay. This also avoids the creation of additional migrations to liberate resource on servers when too many VMs are temporary assigned to them during the reconfiguration.

2 Evaluations

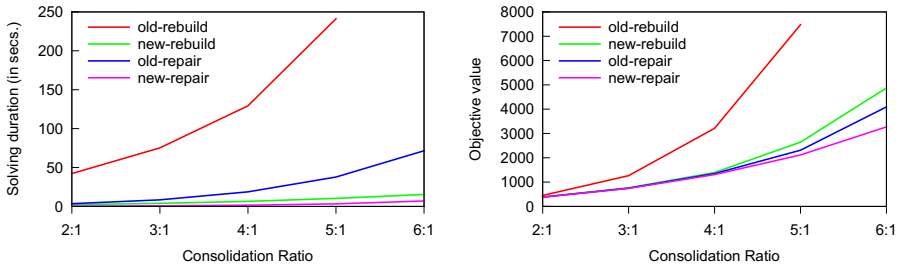
Our modifications reduce the running time of the solving process and improve the quality of the computed solutions. For an accurate comparison of the gains with regards to the published results, we have rerun our experiments using the same set of instances and the same computing servers in the Grid'5000 testbed [2]. Details about the instances and the experimental environment are available in the original paper [3].

2.1 Impact of the consolidation ratio

Ratio	Rebuild Mode					Repair Mode				
	solved	obj	nodes	fails	time	solved	obj	nodes	fails	time
2:1	100	387	1972	0	2.3	100	380	162	0	0.3
3:1	100	766	2952	0	3.9	100	742	393	0	0.8
4:1	100	1393	3932	1	6.5	100	1309	830	0	1.5
5:1	100	2644	4921	9	10.3	100	2117	1574	4	3.2
6:1	100	4873	5958	71	15.3	100	3271	2691	1	7.0

Table 1: Impact of the consolidation ratio on the solving process.

Table 1 shows the impact of the consolidation ratio on the solving process in the *rebuild* and *repair* modes using the new implementation. Figure 1 shows the performance of our new implementation with regards to the original one. Figure 1(a) shows the average solving time to compute the first solution. Figure 1(b) shows the average cost of these solutions.



(a) Computation time for the first solution

(b) Cost of the solution

Figure 1: Gains on the solving process for instances having a variable consolidation ratio

We first observe on Table 1 that all the instances are now solved. This is mostly explained by the bug fix in the search heuristic. Figure 1(a) shows the gain from the optimization of the pack constraints. In the *rebuild* mode, instances with a consolidation ratio of `5:1` are now computed in 17 seconds; 14 times faster. In the *repair* mode, instances with a consolidation ratio of `6:1` are now computed in 7.6 seconds; 9.5 times faster. This gain is slightly less important than the gain in the *rebuild* mode as the number of managed VMs, so the number of items handled by the pack constraint, is reduced. Finally, Figure 1(b) shows the improvements on the computed reconfiguration plans. This cost was reduced by a factor of 2.8 in the *rebuild* mode and by a factor of 1.15 in the *repair* mode.

Our optimizations reduce the original gap between the *rebuild* and the *repair* mode for these instances. The gain is however still appreciable as the *repair* mode still provides better solutions in a fewer amount of time.

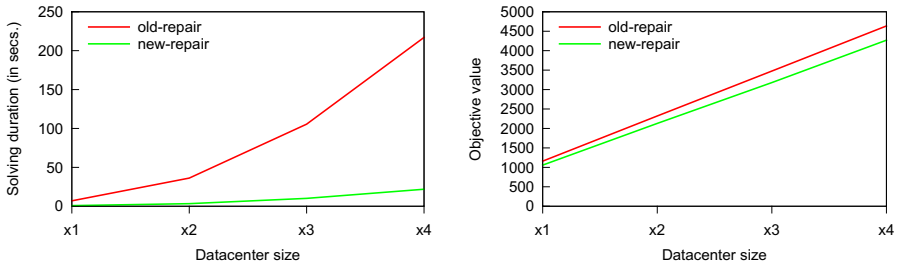
2.2 Impact of the datacenter size

Set	#servers	#VMs	Rebuild Mode					Repair Mode				
			solved	obj	nodes	fails	time	solved	obj	nodes	fails	time
`x1`	500	2,500	100	1326	2457	0	2.0	100	1058	798	0	0.8
`x2`	1,000	5,000	100	2653	4914	0	10.3	100	2130	1578	0	3.3
`x3`	1,500	7,500	100	3950	7370	0	36.5	100	3179	2346	0	10.1
`x4`	2,000	10,000	100	5317	9828	0	88.9	100	4269	3143	0	21.9

Table 2: Impact of the datacenter size on the solving process.

Table 2 shows the impact of the datacenter size on the computation in the *rebuild* and the *repair* modes. Contrary to the old implementation that failed at solving one instance for sets `x2` and `x3`, we observe here that every instances are now solved using the new implementation. The performance gap between the *rebuild* mode and the *repair* mode is more visible in this experiment where instances are getting bigger.

Figure 2 shows the performance of our new implementation with regards to the original one. Figure 2(a) shows the average solving time to compute the first solution while Figure 2(b) shows the average cost of these solutions. We first observe that the gap between the old and the new implementation increases with the size of the problems. The accelerator factor using the new implementation in the *repair* mode stays however over 9. Figure 2(b) shows a constant 10% improvement on the quality of the computed solutions.



(a) Computation time for the first solution

(b) Cost of the solution

Figure 2: Gains on the solving process for variable datacenter sizes.

2.3 Impact of the side constraints

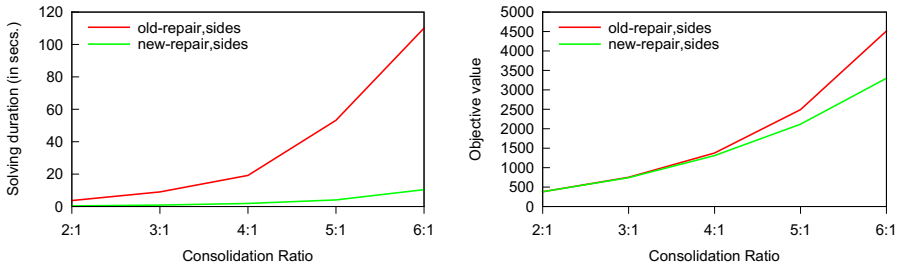
Table 3 shows the impact of the side constraints on the instances with a variable consolidation ratio (left) and with a variable datacenter size (right) using the new implementation. Contrary to the previous imple-

variable consolidation ratios						variable datacenter sizes					
Set	solved	obj	nodes	fails	time	Set	solved	obj	nodes	fails	time
`2:1`	100	381	163	0	0.4	`x1`	100	1059	798	0	0.9
`3:1`	100	742	393	0	0.9	`x2`	100	2131	1578	0	4.2
`4:1`	100	1310	830	0	1.9	`x3`	100	3179	2346	0	13.8
`5:1`	100	2117	1570	0	4.1	`x4`	100	4269	3143	0	30.7
`6:1`	100	3298	3172	0	10.4						

Table 3: Impact of the side constraints on the solving process (*repair* mode).

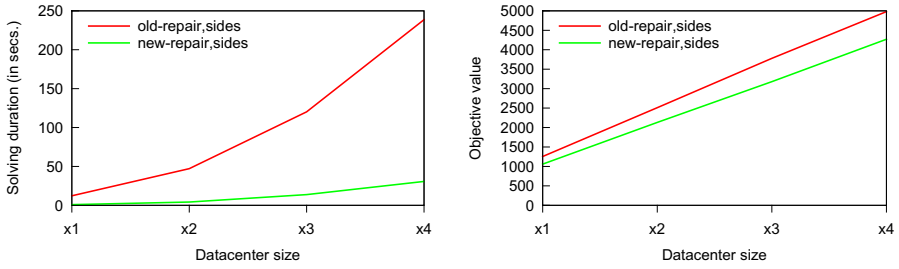
mentation where the additional constraints forbade to solve some of the biggest instances, all the instances can now be solved.

Figure 3 and Figure 4 shows the impact of the side constraints on the solving process for instances having a variable consolidation ratio and a variable datacenter size, respectively. Similar to the previous experiments, we observe that the optimizations performed on our model and our implementation improve the performance by a significant order of magnitude.



(a) Solving duration (b) Cost of the solutions

Figure 3: Impact of the side constraints with variable consolidation ratios



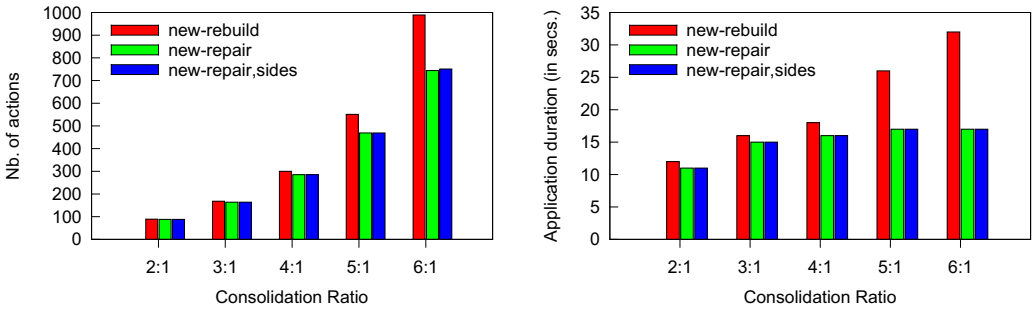
(a) Solving duration (b) Cost of the solutions

Figure 4: Impact of the side constraints with variable datacenter sizes

2.4 Practical quality of our solutions

The objective value denotes the quality of the computed reconfiguration plan. In practice, this quality is mostly characterized by the amount of actions to execute and the estimated application duration of the plan. Such a representation of the solution was not included in the original paper for space constraints.

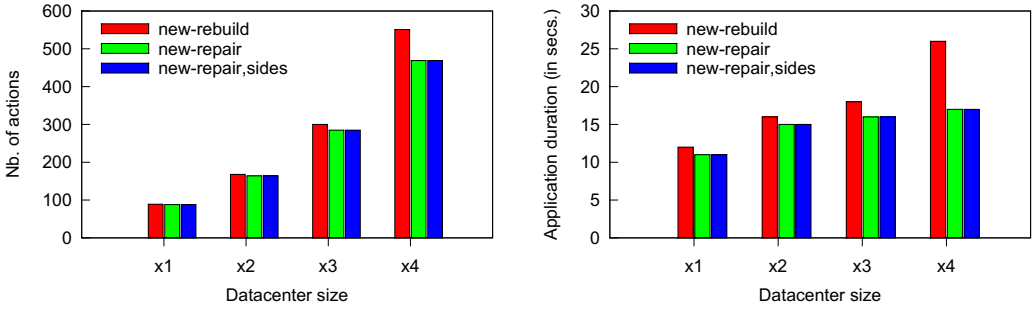
Figure 5 and 6 show for the new implementation, the practical cost of the computed plans, for the instances with a variable consolidation ratio and a variable datacenter size, respectively. We observe first



(a) Number of actions

(b) Estimated application duration

Figure 5: Practical cost of the reconfiguration plans with variable consolidation ratios



(a) Number of actions

(b) Estimated application duration

Figure 6: Practical cost of the reconfiguration plans with variable datacenter sizes

that the gain between the *rebuild* mode and the *repair* mode is very interesting in practice. Indeed, for the biggest problems, the estimated application duration is divided by up to 2. With faster reconfiguration plans, Entropy will be more reactive: it will quickly fix performance issues while being able to manage VMs with frequent resource requirement variations. We also observe a significant reduction in the number of actions to execute. In practice, a fewer amount of migrations will then be executed in the *repair* mode. This improvement reduces the temporary performance degradation for the VMs involved in migrations while saving network bandwidth.

Finally, we observe that the practical quality of the reconfiguration plans is mostly equivalent when the side constraints are considered. For the instance set `x4`, only 6 instances have a reconfiguration plan made up with more actions when the side constraints are considered. This is explained by the VM placement heuristic in the *repair* mode. Indeed, the search heuristic computes that about half of the servers are candidates to host VMs with a guarantee to have their associated action scheduled without any delay.

3 Conclusions

The modifications we performed on Entropy lead to a major improvements on our results. With regards to the published results, Entropy is now capable of solving up to ten time faster instances that can be harder to solve, while providing better reconfiguration plans.

It has to be noticed that our optimizations do not change the theoretical complexity of the problem. The solving duration of the instances is still exponential with regards to the size of the instances and the consolidation ratio. However, we have shifted the solving capability of Entropy to a point that current instances are no longer complex and big enough to show the limitation of our implementation.

References

[1] C. Bessiere, E. Hebrard, B. Hnich, and Z. Kiziltan. Among, common and disjoint constraints. In *In CSCLP: Recent Advances in Constraints, volume 3978 of LNCS*, pages 29–43. Springer, 2006.

[2] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, B. Quetier, O. Richard, E.-G. Talbi, and I. Touche. Grid'5000: A large scale and highly reconfigurable experimental grid testbed. *Int. J. High Perform. Comput. Appl.*, 20:481–494, November 2006.

[3] F. Hermenier, S. Demassey, and X. Lorca. Bin repacking scheduling in virtualized datacenters. In J. Lee, editor, *Principles and Practice of Constraint Programming, CP 2011*, volume 6876 of *Lecture Notes in Computer Science*, pages 27–41. Springer Berlin / Heidelberg, 2011.