On Modelling Virtual Machine Consolidation to Pseudo-Boolean Constraints

Bruno Cesar Ribas[1,3], Rubens Massayuki Suguimoto[2], Razer A. N. R. Montaño[1], Fabiano Silva[1], Luis C. E. de Bona[2], Marcos Castilho[1]

[1]LIAMF - Laboratório de Inteligência Artificial e Métodos Formais

[2]LARSIS - Laboratório de Redes e Sistemas Distribuídos
Federal University of Paraná

[3]Universidade Tecnológica Federal do Paraná - Campus Pato Branco

IBERAMIA, 2012

Summary

Introduction

- Cloud Computing is a new paradigm of distributed computing that offers virtualized resources and services over the Internet.
- One of the service model offered by Clouds is Infrastructure-as-a-Service (IaaS) in which virtualized resource are provided as virtual machine (VM).
- Cloud providers use a large data centers in order to offer IaaS.
- Most of data center usage ranges from 5% to 10%.

Introduction(2)

- In order to maximaze the usage, a IaaS Cloud provider can apply server consolidation, or VM consolidation.
- Consolidation can increase workloads on servers from 50% to 85%, operate more energy efficiently and can save 75% of energy.
- Reallocating VM allow to shutdown physical servers, reducing costs (cooling and energy consumption), headcount and hardware management.

Related Works

- Optimal VM consolidation has been explored and solved using Linear Programming formulation and Distributed Algorithms approaches.
- Marzolla et al. presents a gossip-based *distributed algorithm* called V-Man. Each physical server (host) run V-Man with an *Active* and *Passive* threads. Active threads request a new allocation to each neighbor sending to them the number of VMs running. The Passive thread receives the number of VMs, calculate and decide if current node will pull or push the VMs to requested node. The algorithm iterate and quickly converge to an optimal consolidation, maximizing the number of idle hosts.

Related Works(2)

- Ferreto et. al. presents a Linear Programming formulation and add constraints to control VM migration on VM consolidation process. The migration control constraints uses CPU and memory to avoid worst performance when migration occurs.
- Bossche et. al. propose and analyze a *Binary Integer Programming* (BIP) formulation of cost-optimal computation to schedule VMs in Hydrid Clouds. The formulation uses CPU and memory constraints and the optimization is solved by *Linear Programming*.
- We introduce an artificial intelligence solution based on *Pseudo-Boolean formulation* to solve the problem of optimal VM consolidation.

- A Pseudo-Boolean function in a straightforward definition is a function that maps Boolean values to a real number;
- PB constraints are more expressive than clauses (one PB constraint may replace an exponential number of clauses)
- A pseudo-Boolean instance is a conjunction of PB constraints

- **PBS (Pseudo Boolean Satisfaction)**
  - decide of the satisfiability of a conjunction of PB constraints
- **PBO (Pseudo Boolean Optimization)**
  - find a model of a conjuction of PB constraints which optimizes one objective function

$$\begin{cases} \textit{minimize}, & f = \sum_i c_i \times x_i \text{with } c_i \in \mathbb{Z}, x_i \in \mathbb{B} \\ \text{subject to} & \text{the conjunction of constraints} \end{cases}$$

- The goal of our problem is to deploy K VMs $\{vm_1 \ldots vm_K\}$ inside N hardwares $\{hw_1 \ldots hw_N\}$ while minimizing the total number of active hardwares. Each VM vm_i has an associated needs such as number of VCPU and amount of VRAM needed while each physical hardware hw_j has an amount of available resources, number of CPU and available RAM.

- In order to create the PB Constraints each hardware consists of two variables:

  hw_i^{ram} tha relates the amount of RAM in hw_i

  hw_i^{proc} that relates to the amount of CPU in hw_i

- Per hardware, a VM has 2 variables:

  $vm_j^{ram \cdot hw_i}$ to relate the VM vm_j required amount of VRAM vm_j^{ram} to the hardware hw_i amount of RAM hw_i^{ram}

  $vm_j^{proc \cdot hw_i}$ relate the required VCPU vm_j^{proc} to the amount of CPU available hw_i^{proc}

- The total amount of VM variables is $2 \times N$ variables.

- Our main objective is to minimize the amount of active hardware. This constraint is defined as:

$$minimize : \sum_{i=1}^{N} hw_i \tag{1}$$

- Each hw_i is a Boolean variable that represents one hardware that, when *True*, represents that hw_i is powered on and powered off otherwise.

- To guarantee that the necessary amount of hardware is active we include two more constraints that implies that the amount of usable RAM and CPU must be equal or greater than the sum of resources needed by VM.

$$\sum_{i=1}^{N} RAM_{hw_i} \cdot hw_i^{ram} \geq \sum_{j=1}^{K} RAM_{vm_j} \cdot vm_j^{ram} \tag{2}$$

$$\sum_{i=1}^{N} PROC_{hw_i} \cdot hw_i^{proc} \geq \sum_{j=1}^{K} PROC_{vm_j} \cdot vm_j^{proc} \tag{3}$$

To limit the upper bound of hardwares, we add two constraints per host that limit:

available RAM per hardware: This constraint dictates that the sum of needed ram of virtual machines must not exceed the total amount of ram available on the hardware, and it is illustrated in constraint 4;

available CPU per hardware: This constraint dictates that the sum of VCPU must not exceed available CPU, and it is illustrated in constraint 5.

$$\forall\, hw_i^{ram} \in hw_N^{ram} \left(\sum_{j=1}^{K} RAM_{vm_j} \cdot vm_j^{ram \cdot hw_i} \leq RAM_{hw_i} \right) \quad (4)$$

$$\forall\, hw_i^{proc} \in hw_N^{proc} \left(\sum_{j=1}^{K} PROC_{vm_j} \cdot vm_j^{proc \cdot hw_i} \leq PROC_{hw_i} \right) \quad (5)$$

- Finally we add one constraint per VM to guarantees that the VM is running in exactly one hardware.

$$\forall\, vm_i \in vm_K \left(\sum_{j=1}^{N} vm_i^{proc \cdot hw_j} \cdot vm_i^{ram \cdot hw_j} \cdot hw_j^{proc} \cdot hw_j^{ram} = 1 \right) \tag{6}$$

- With this model we have $(2 \times N + 2 \times N \times K)$ variables and $(2 + 2 \times N + K)$ constraints with one more constraint to minimize in our PB formula.

Host	RAM	CPU
hw1	30	4
hw2	18	4
hw3	10	8
hw6	10	8
hw5	30	4
prd3b	125	32
prd3d	125	32
prd3c	125	32
tesla1	62	16
SUM	535	140

(a) Hardware description.

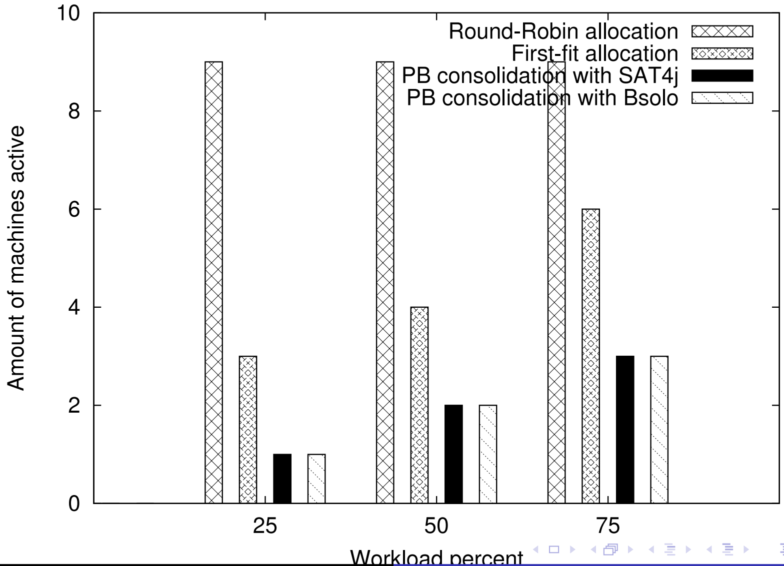
VM	VRAM	VCPU	VM	VRAM	CPU
planetmon	12	4	db	2	1
vc3-blanche	8	4	devel	4	2
alt	10	8	salinas	5	2
dalmore	10	8	vc3-colombard	8	2
mumm	10	8	vc3-educacional	2	2
priorat	5	8	vc3-newcastle	4	2
talisker	32	8	vc3-qef1	2	2
bowmore	20	12	vc3-qef2	2	2
alt-marcadle	80	16	vc3-qef3	2	2
alt-murphy	93	24	vc3-qef4	2	2
caporal	18	4	alt-guinness	120	32
			SUM	451	155

Workload Percent	$\sum$ VRAM	$\sum$ VCPU	Amount of VMs
25%	51	23	11
50%	81	39	14
75%	138	71	18

Table: Table of workload subsets with σ equals to 25%, 50% and 75% and respectives sum of VRAM, VCPU and amount of VMs for DInf-UFPR scenario.

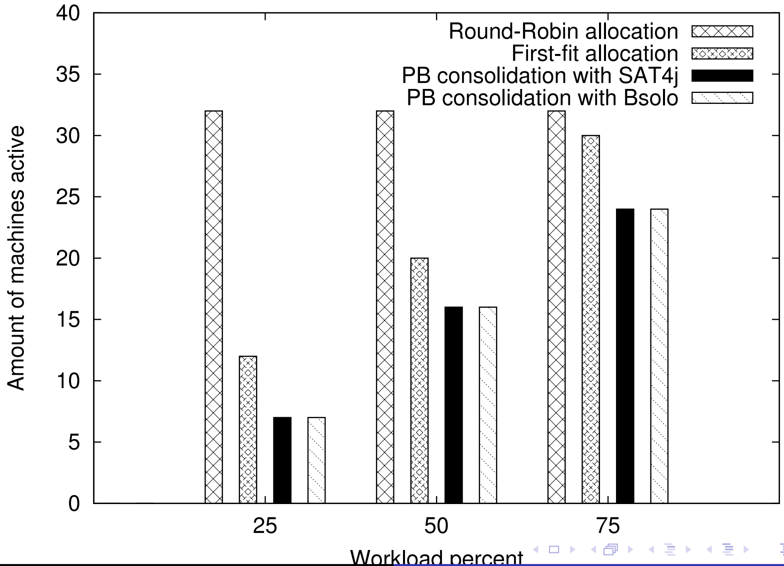
Formula	Variables	Constraints	BSOLO	Sat4j-PB
hw9-vm25p	216	31	**0.004**	**0.101**
hw9-vm50p	270	34	**0.004**	**0.109**
hw9-vm75p	342	38	**0.004**	**0.118**

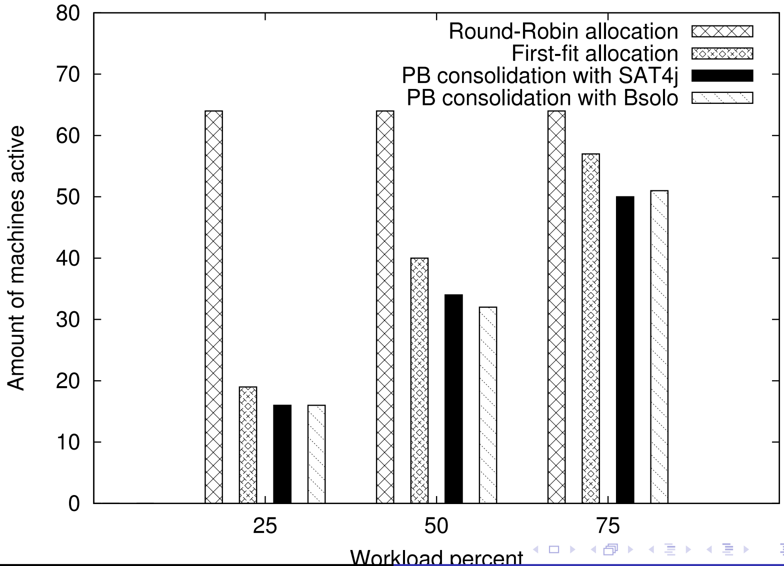
Table: Variables and constraints generated and execution time for DInf-UFPR scenario using BSOLO and Sat4j-PB solvers.

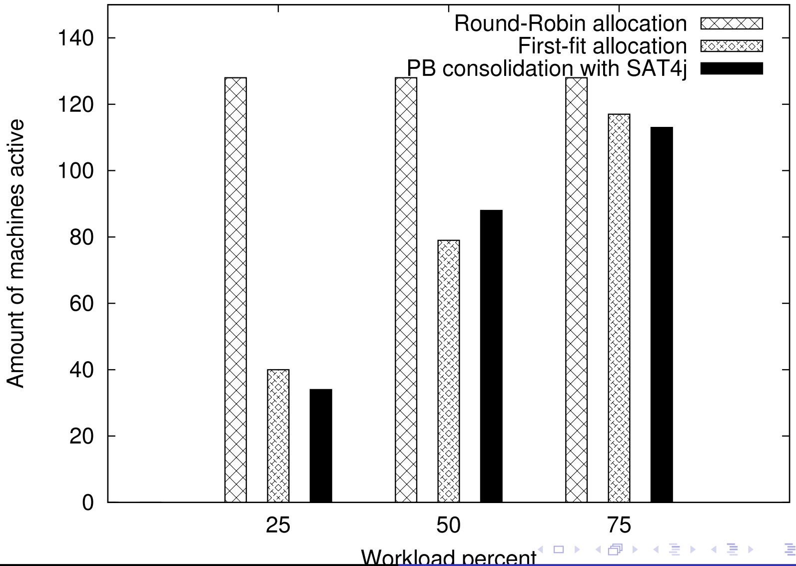

Round-Robin allocation
First-fit allocation
PB consolidation with SAT4j
PB consolidation with Bsolo
Amount of machines active
0
2
4
6
8
10
25
50
75
Workload percent

#Machines	RAM	CPU	Workload %	$\sum$ VRAM	$\sum$ VCPU	#Tasks
32	14.9813	17.0000	25%	3.7375	4.3475	98
32	14.9813	17.0000	50%	5.7048	8.5640	173
32	14.9813	17.0000	75%	9.5204	12.7674	278
64	32.2117	34.5000	25%	5.7281	8.6389	174
64	32.2117	34.5000	50%	13.8382	17.2724	371
64	32.2117	34.5000	75%	19.3733	25.8826	559
128	61.8284	68.0000	25%	13.5025	17.0473	368
128	61.8284	68.0000	50%	26.3261	34.3367	713
128	61.8284	68.0000	75%	39.0425	51.0215	1048
256	121.5035	134.5000	25%	26.2943	33.9555	712
256	121.5035	134.5000	50%	49.0585	67.2507	1407
256	121.5035	134.5000	75%	75.6842	10.08777	2119

Formula	Variables	Constraints	BSOLO	Sat4j-PB
hw32-vm25p	6336	164	7242.75	**305.277**
hw32-vm50p	11136	239	7198.01	7204.971
hw32-vm75p	17856	344	7237.44	**6417.293**
hw64-vm25p	22400	304	7198.02	7227.192
hw64-vm50p	47616	501	7198.02	7243.419
hw64-vm75p	71680	689	7198.19	7243.385
hw128-vm25p	94464	626	TLE	7244.51
hw128-vm50p	182784	971	TLE	7244.46
hw128-vm75p	268544	1306	TLE	7243.678
hw256-vm25p	365056	1226	TLE	TLE
hw256-vm50p	720896	1921	RTE	TLE
hw256-vm75p	1085440	2633	RTE	TLE


Round-Robin allocation
First-fit allocation
PB consolidation with SAT4j
PB consolidation with Bsolo
Amount of machines active
0
5
10
15
20
25
30
35
40
25
50
75
Workload percent


Round-Robin allocation
First-fit allocation
PB consolidation with SAT4j
PB consolidation with Bsolo
Amount of machines active
0
10
20
30
40
50
60
70
80
25
50
75
Workload percent


Round-Robin allocation
First-fit allocation
PB consolidation with SAT4j
Amount of machines active
0
20
40
60
80
100
120
140
25
50
75
Workload percent

- PB Constraints can be used to optimize costs
- PB solvers were not able to solve the formulas of a huge test scenario such as Google Cluster
- We can use these formulas as a good benchmark to improve PB solvers
- Extend our solution and implement it inside a Cloud Management System
- Add some important restrictions such as network dependency of VMs and create classes of VMs to make better use of network interfaces of hosts.