PBFVMC: A New Pseudo-Boolean Formulation to Virtual-Machine Consolidation

Bruno Cesar Ribas[1,3], Rubens Massayuki Suguimoto[2], Razer A. N. R. Montaño[1], Fabiano Silva[1], 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

BRACIS, 2013

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 Work

- 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 Work(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 introduced an artificial intelligence solution based on *Pseudo-Boolean formulation* to solve the problem of optimal VM consolidation and this work refines this method.

Pseudo-Boolean Optimization

- A Pseudo-Boolean function in a straightforward definition is a function that maps Boolean values to an integer 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

Pseudo-Boolean

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

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

Problem Description

- The goal of our problem is to deploy K VMs $\{vm_1 \dots vm_K\}$ inside N hardwares $\{hw_1 \dots hw_N\}$ 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.

First PB formulation to Optimal VM consolidation

- 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.

First PB formulation to Optimal VM consolidation

- 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.

First PB formulation to Optimal VM consolidation

- 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.

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

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

First PB formulation to Optimal VM consolidation

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.

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

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

First PB formulation to Optimal VM consolidation

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

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

First PB formulation to Optimal VM consolidation

- 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.

Main Issues with this Approach

- Slow on bigger problems
  - Based on Bin Packing Formulation
- Equality Constraints Hard to Solve
  - Replaceable by two constraints, $\leq$ and $\geq$
- $\leq$ constraints not always good for a solver

PBFVMC

- Based on Pigeon Hole formulation
- Rework to be faster than previous formulation
- Merged variables
  - hw_i^r and hw_i^p to hw_i
  - vm_j^r and vm_j^p to vm_j
- All constraints in PosiForm
  - Only $\geq$
  - Non-negative coefficients

PBFVMC Variables

- N : Total number of available hardware (hw);
- K : Total number of virtual machines (VM);
- hw_i : Hardware $i \in N$;
- $vm_j^{hw_i}$: Virtual Machine $j \in K$ that runs in hw_i;
- R_{hw_i} and P_{hw_i} : Physical RAM and Processor count per hardware;
- R_{vm_i} and P_{vm_i} : RAM and Processor count needed per VM.

PBFVMC

- Objective Function is the summation of the ON servers

$$minimize : \sum_{i=1}^{N} hw_i \tag{7}$$

- Summation of memory and processing power of ON server is enough to all virtual machines

$$\sum_{i=1}^{N} R_{hw_i} \cdot hw_i \geq \sum_{j=1}^{K} R_{vm_j} \tag{8}$$

$$\sum_{i=1}^{N} P_{hw_i} \cdot hw_i \geq \sum_{j=1}^{K} P_{vm_j} \tag{9}$$

PBFVMC

- Upper limit on the total resources each hardware may provide in relation to the virtual machines that may run on this hardware

$$\forall\, i \in 1..N \left(\sum_{j=1}^{K} (R_{vm_j} \cdot \neg vm_j^{hw_i}) + R_{hw_i} \cdot hw_i \geq \sum_{j=1}^{K} R_{vm_j} \right) \quad (10)$$

$$\forall\, i \in 1..N \left(\sum_{j=1}^{K} (P_{vm_j} \cdot \neg vm_j^{hw_i}) + P_{hw_i} \cdot hw_i \geq \sum_{j=1}^{K} P_{vm_j} \right) \quad (11)$$

PBFVMC

- Virtual Machine must be running in some hardware

$$\forall j \in 1..K \left(\sum_{i=1}^{N} vm_j^{hw_i} \geq 1 \right) \tag{12}$$

- A Virtual Machine runs in exactly one hardware

$$\forall j \in 1..K \left(\sum_{i=1}^{N} \neg vm_j^{hw_i} \geq N - 1 \right) \tag{13}$$

Highlights

- Total Variables: $(N + N \times K)$
- Total Constraints: $(2 + 2 \times N + 2 \times K)$
- Algebraic Manipulation of constraints (4) and (5) to generate (10) and (11)
- Remove all non-linear constraints

First PB formulation to Optimal VM consolidation

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.

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

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

PBFVMC

- Upper limit on the total resources each hardware may provide in relation to the virtual machines that may run on this hardware

$$\forall\, i \in 1..N \left(\sum_{j=1}^{K} (R_{vm_j} \cdot \neg vm_j^{hw_i}) + R_{hw_i} \cdot hw_i \geq \sum_{j=1}^{K} R_{vm_j} \right) \quad (10)$$

$$\forall\, i \in 1..N \left(\sum_{j=1}^{K} (P_{vm_j} \cdot \neg vm_j^{hw_i}) + P_{hw_i} \cdot hw_i \geq \sum_{j=1}^{K} P_{vm_j} \right) \quad (11)$$

Highlights - Pigeon Restrictions

- Pigeon Hole formulation is easily done with clauses leading to $(n+1)$ clauses, where n is the number of holes, saying that a pigeon has to be placed in some hole
- The problem is that the number of clauses increases rapidly as the number of pigeons grow

$$\forall j \in 1..K \left(\sum_{i=1}^{N} vm_j^{hw_i} \geq 1 \right) \tag{14}$$

$$\forall j, i, k \in 1..K, 1..N, i+1..K \ (\neg vm_j^{hw_i} + \neg vm_k^{hw_i}) \geq 1 \tag{15}$$

Size of the Formulae

		Previous		PBFVMC	
HW	VMS	Vars	Constr	Vars	Constr
hw32-vm25p	98	6336	164	3168	262
hw32-vm50p	173	11136	239	5568	412
hw32-vm75p	278	17856	344	8928	622
hw32-vm85p	320	20544	386	10272	706
hw32-vm90p	325	20864	391	10432	716
hw32-vm95p	348	22336	414	11168	762
hw32-vm98p	364	23360	430	11680	794
hw32-vm99p	366	23488	432	11744	798

Size of the Formulae

		Previous		PBFVMC	
HW	VMS	Vars	Constr	Vars	Constr
hw512-vm25p	1432	1467392	2458	733696	3890
hw512-vm50p	2771	2838528	3797	1419264	6568
hw512-vm75p	4035	4132864	5061	2066432	9096
hw512-vm85p	4431	4538368	5457	2269184	9888
hw512-vm90p	4745	4859904	5771	2429952	10516
hw512-vm95p	5068	5190656	6094	2595328	11162
hw512-vm98p	5319	5447680	6345	2723840	11664
hw512-vm99p	5402	5532672	6428	2766336	11830

Experiments Setup

- Google Cluster DATA
- Using a range of hardware: 32, 64, 128, 256 and 512
- A range of workloads, in percentage of resource needed: 25%, 50%, 75%, 85%, 90%, 95%, 98%, and 99%
- All experiments ran in a Intel Xeon 2.1GHz, 256GB of memory running GNU/Linux.

Experiments - Decide SAT

Formula	Previous	PBFVMC
hw32-vm25p	92.756	0.433
hw32-vm50p	35.643	0.542
hw32-vm75p	3.43	0.588
hw32-vm85p	4.516	0.911
hw32-vm90p	6.795	9.716
hw32-vm95p	3442.92	8.129
hw32-vm98p	TLE	45.589
hw32-vm99p	TLE	5566.28
hw64-vm25p	3118.029	0.706
hw64-vm50p	18.306	0.892
hw64-vm75p	50.687	1.15
hw64-vm85p	60.38	1.365
hw64-vm90p	121.006	1.423
hw64-vm95p	TLE	7.512
hw64-vm98p	TLE	4135.757
hw64-vm99p	TLE	240.538

Experiments - Decide SAT

Formula	Previous	PBFVMC
hw128-vm25p	TLE	1.731
hw128-vm50p	4015.592	2.753
hw128-vm75p	5975.386	4.026
hw128-vm85p	7676.653	7.984
hw128-vm90p	13491.676	7.904
hw128-vm95p	TLE	65.916
hw256-vm25p	TLE	4.379
hw256-vm50p	TLE	14.244
hw256-vm75p	TLE	33.259
hw256-vm85p	TLE	48.298
hw256-vm90p	TLE	126.506
hw256-vm95p	TLE	389.329
hw256-vm98p	TLE	7737.502

Experiments - Decide SAT

Formula	Previous	PBFVMC
hw512-vm25p	TLE	28.436
hw512-vm50p	TLE	162.289
hw512-vm75p	TLE	508.322
hw512-vm85p	TLE	287.437
hw512-vm90p	TLE	5604.022
hw512-vm95p	TLE	4222.892

Experiments - Optimize

Formula	Previous	PBFVMC
hw32-vm25p	**249.897/7**	**191.912/7**
hw32-vm50p	35.696/16	4.134/16
hw32-vm75p	23.628/25	**772.657/24**
hw32-vm85p	1175.103/29	159.86/28
hw32-vm90p	108.361/31	**948.924/29**
hw32-vm95p	3442.92/32	**319.041/31**
hw32-vm98p	TLE	**45.651/32**
hw32-vm99p	TLE	**5566.491/32**
hw64-vm25p	4248.893/17	8.541/16
hw64-vm50p	6477.271/33	200.261/33
hw64-vm75p	8784.933/50	8608.38/47
hw64-vm85p	603.393/59	490.656/55
hw64-vm90p	1272.89/62	869.421/58
hw64-vm95p	TLE	679.719/62
hw64-vm98p	TLE	4135.757/64
hw64-vm99p	TLE	**240.642/64**

Experiments - Optimize

Formula	Previous	PBFVMC
hw128-vm25p	TLE	10319.859/29
hw128-vm50p	14661.134/75	4856.869/64
hw128-vm75p	16209.656/105	12538.628/98
hw128-vm85p	11203.456/122	1117.772/115
hw128-vm90p	13491.676/128	11295.761/117
hw128-vm95p	TLE	65.916/128
hw256-vm25p	TLE	12381.653/68
hw256-vm50p	TLE	3576.626/136
hw256-vm75p	TLE	11468.942/204
hw256-vm85p	TLE	10537.747/230
hw256-vm90p	TLE	2704.592/243
hw256-vm95p	TLE	2003.068/255
hw256-vm98p	TLE	7737.502/256

Experiments - Optimize

Formula	Previous	PBFVMC
hw512-vm25p	TLE	4471.005/140
hw512-vm50p	TLE	5406.047/281
hw512-vm75p	TLE	4378.66/408
hw512-vm85p	TLE	4919.328/461
hw512-vm90p	TLE	14426.6/487
hw512-vm95p	TLE	6864.151/510

Conclusion and Future Work

- With PBFVMC solvers spend most the running time dedicated to optimize the formula while in the previous formulation most of the time are spent trying to decide whether the formula is satisfiable;
- This approach is solved by a generic solver, no need to write specific solver;
- PB solvers were not able to optimize bigger instances;
- Work on some aspects of the formulation to achieve better results;
- We can use these formulas as a good benchmark to improve PB solvers;
- Add some important restrictions such as network dependency of VMs and create classes of VMs to make better use of network interfaces of hosts.

Thank You

- Bruno Ribas
- ribas@inf.ufpr.br

Strengthening

- Strengthening is a method where a literal, or a set of literals are fixed a value and then a propagation is applied to the formula. Some assumptions will cause some constraints to become oversatisfied, i.e. suppose that after setting a literal, l_0 to true, we discover that a constraint c is given by $\sum w_i l_i \geq r$ becomes oversatisfied by an amount s in that the sum of the left hand side is greater (by s) than the amount required bu the right hand side of the inequality. The oversatisfied constraint c can now be replaced by the following:

$$s \cdot \neg l_0 + \sum w_i l_i \geq r + s \tag{16}$$

- if l_0 is true, we know that $\sum w_i l_i \geq r + s$, so (16) holds. If l_0 is false, then $s \cdot \neg l_0 = s$ and we still must satisfy the original constraint $\sum w_i l_i \geq r$, so (16) still holds. The new constraint implies the original one, so no information is lost in the replacement. The power of this method is that it allows us to build more complex axioms from a set of simple ones. The strengthened constraint will often subsume some or all of the constraints involved in generating it.

Strengthening

- In the case of a pigeon hole formulation, constraint (15) will be strengthened and will subsume all constraints (15), which will be replaced by constraint (13), generating a smaller and richer set of constraints, taking advantage of all the power pseudo-boolean provides and yet keeping with linear and normalized constraints.

Strengthening - Example

$$a + b \geq 1 \quad (17)$$

$$a + c \geq 1 \quad (18)$$

$$b + c \geq 1 \quad (19)$$

- With $P = \{\neg a\}$, $\{b, c\}$ will be propagated
- Restriction 19 is super satisfied and can be replaced by:

$$a + b + c \geq 2$$

- This new constraint subsumes the three original constraints, then 17 and 18 can be removed.