Energy Efficient and Scalable Resource Allocation in SDN Based Cloud Data Centers

Giuseppe Portaluri, Stefano Giordano, Davide Adami, Gregorio Procissi

University of Pisa

January 15th, 2015

Contents

Contents

Introduction

Cloud computing data centers provide high performance, scalability and access to virtually unlimited computational power to application providers.

For optimal operation, data center services providers need to:

- perform hardware maintenance;
- provide redundancy;
- optimize power consumption;
- manage task execution and network traffic.

Introduction

Related Works

VMPlanner is a Virtual Machine (VM) and traffic flows allocator able to reduce data center power cost by putting in sleep mode network elements.

VMPlanner solves the Virtual Machine (VM) allocation problem with three different algorithms which use approximation and are not scalable due to only a limited number of switches and racks that could be considered.

Introduction

Related Works

Green SLA uses best effort scheduling, which minimizes task execution time and energy-performance trade off.

This approach implements a number of advanced power management strategies such as Dynamic Voltage and Frequency Scaling (DVFS) and supports parallel execution.

Introduction

Related Works

Another approach maps VMs to physical resources using genetic algorithm improved with fuzzy multi-objective optimization.

It tries to reduce the amount of power consumed by the servers, optimize CPU and memory utilization, and minimize peak temperatures inside the facility.

Introduction

Paper proposal

We propose a new resource allocation approach for cloud computing data centers that performs joint allocation of computational and network resources.

The objective is finding trade-off solutions between tasks completion time and system power consumption avoiding network link congestion.

This resource allocation is meant to be performed by a Software Defined Networking (SDN) centralized controller.

Introduction

Paper proposal

This algorithm is:

- designed using Genetic Algorithms and Simulated annealing meta-heuristics that allow both to explore solutions space and to search for the optimal solution in an efficient manner;
- based on a model developed to capture specifics of the data center network topology and device power consumption.

Contents

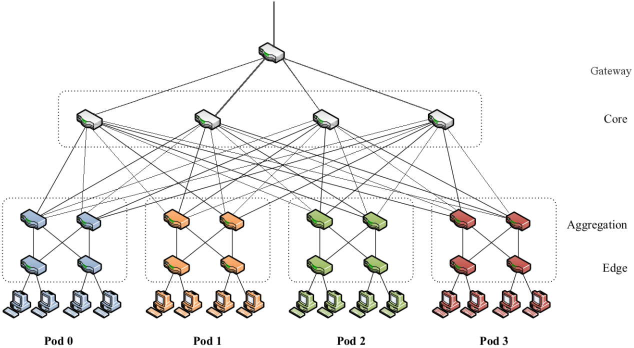
System Model

Data center

We model data center with currently the most widely used three-tier fat-tree architecture.

IT is composed of the access (or edge), aggregation, and core layers.

Figure 1: Fat-tree topology in data centers.



System Model

Data center

Access layer provides connection to servers which are arranged into racks with each rack being served by a single Top of the Rack (ToR) switch.

Each ToR switch is connected to a pair of Aggregation switches.

A group of ToR switched connected to the same pair of Aggregation switches is called Pod.

Links interconnecting server and ToR switches have capacity of 1 Gb/s.

Links between racks and aggregation switches are 2 x 10 Gb/s.

System Model

Task

The proposed algorithm allocates in this data center, which is empty at the beginning, a set of independent tasks which are characterized by:

- a number of instructions to be executed;
- a constant amount of bandwidth required to perform the execution.

All tasks are independent and do not require to communicate during their execution.

Bandwidth requirement is only related to communication between computing servers and data center gateway.

System Model

Server

Computing servers are modeled with single-core processors.

Server offers a fixed computational power expressed in instructions per second.

Tasks allocated for execution on the same server will share servers processing power equally.

System Model

Server

The power consumption of computing servers (P_{SR}) is modeled using binary law, assuming that each processor is either executing a certain number of tasks (n) at the full speed or is turned off.

$$P_{SR} = \begin{cases} P_{SRP} & \text{if } n \geq 1; \\ 0 & \text{if } n = 0. \end{cases} \tag{1}$$

System Model
Switch

Power consumption of a network switch is linearly proportional to the traffic load and stays between two values P_{SWP} (link full utilization) and P_{SWI} (switch idle).

Network switches left idle can be turned off to optimize their power consumption.

System Model

Switch

Denoting:

- l the load factor of a switch
- $P_{SW}(l)$ the switch power consumption

network switch power consumption can be computed as follows:

$$P_{SWk}(l) = \begin{cases} P_{SWI} + (P_{SWP} - P_{SWI})l & \text{if } 0 < l \leq 1; \\ 0 & \text{if } l = 0. \end{cases} \tag{2}$$

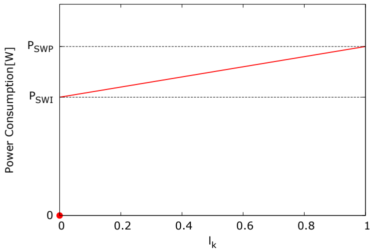


Figure 2: Network switch power consumption.

System Model

Objective

The following two objectives must be minimized:

1. Maximum total completion time of all tasks (*makespan*);
2. Power consumption of servers and network switches.

The optimization problem is the subject to two different constraints:

- is not possible to allocate on a single server tasks demanding higher bandwidth than the available link capacity (1Gb/s);
- the 20 Gb/s uplink capacity of ToR switches cannot be exceeded by the sum of the traffic.

Contents

Optimal Allocation of Resources

Problem Description

In our problem we encode solutions as an integer vector of size equal to the number of tasks (N).

The value assigned to every component of the vector represents a server where the corresponding task is allocated to (this value ranges from 1 to the number of servers M).

This problem is a *constrained multi-objective bin packing problem* which is NP-Hard.

Optimal Allocation of Resources

Problem Description

Solution space has cardinality M^N and optimum research is computational expensive because:

- the solution space size becomes very large also for low values of servers and tasks;
- solution set is non-convex;
- no domain relaxation could be adopted because the objective functions are defined only onto the integer domain.

For these reasons, we adopted two heuristics to perform the research of allocation vectors which minimize as much as possible the two objectives at the same time.

Optimal Allocation of Resources

Multi-objective optimization

In multi-objective algorithms, solutions are not comparable because of the problem of incommensurability.

The main task of a multi-objective optimizator is to find a set of *non-dominated* solutions (or Pareto-optimal solutions) optimizing the current problem.

Optimal Allocation of Resources

Genetic Algorithms

Genetic algorithms (GAs) are iterative stochastic optimization methods based on the principles of natural selection and evolution.

A population of candidate solutions is evolved toward better solutions with the application of genetic operators.

Each solution has a set of properties which can be:

- recombined
- randomly mutated
- altered

to form a new set of solutions.

In each iteration the population of solutions is evaluated considering the fitness of every individuals (usually related to the objective function).

The more fit individuals are selected and survive for consequent iterations.

Optimal Allocation of Resources

Simulated Annealing

Simulated Annealing (SA) is a fast and robust technique used in single and multi-objective optimization problems inspired to the metal cooling down mechanism.

This heuristic performs a local search from an initial point moving towards a random solution present in its neighbourhood.

If the neighbour solution minimizes the current solution we move in it, otherwise with a certain probability the neighbour solution is rejected or accepted in order to avoid local minimum trap.

Optimal Allocation of Resources

Simulated Annealing

Iterations number and probability of acceptance depend both on a parameter named *temperature* which usually monotonically decreases reaching lower values at the end.

The implemented acceptance policy is:

$$P_a = exp\left(-\frac{d}{T}\right) \tag{3}$$

denoting d the 2-norm distance between current and neighbour solution objectives and T the temperature.

If the temperature value is high, the probability of acceptance of a non optimizing solution is higher providing sufficient energy to leave the minimum local area.

Optimal Allocation of Resources

Simulated Annealing

Simulated annealing itself does not provide any specifications about constrained problems.

During neighbour generation phase if the generated neighbour does not violate any constraint it is chosen otherwise this neighbour is considered as starter point for a new phase of neighbour research.

This procedure terminates if:

- a non violating constraint solution is found;
- the number of neighbours explored is equal to the variable size (i.e. task number)

In this second case at the end is generated randomly a new solution and is compared with the less constraint violating solution found during the neighbourhood traversal phase and the best one (i.e. the one which less violates the sum of all constraints) is chosen.

Optimal Allocation of Resources
SDN

Software Defined Networking (SDN) is a dynamic and a flexible network architecture in which the network control plane logic is programmable and decoupled by the forwarding plane.

In SDN, the control plane is logically centralized and can be programmed through a well-defined API, exercising direct control over the state in the network data plane elements.

Our idea is to modify slightly SDN controller to execute our algorithm and to perform resource allocation automatically.

Contents

Experimental Setup

Scenario

Parameters related to data center topology, tasks, servers and hardware power consumption are described in Tables 1 and 2

Table 1: Topology, server and task related parameters

Parameter	Value
Server per rack	24
Rack per pod	8
IPS	$90 * 10^9 [instr/sec]$
Task instruction distribution	Uniform in the range $[18;72] * 10^{13}$ $[instr]$
Task bandwidth distribution	Uniform in the range $[1;20] * 10^6$ $[bps]$

Table 2: Power consumptions

Parameter	Value [W]
Server P_{SRP}	300
ToR P_{SWP}	200
ToR P_{SWI}	160
Aggregation P_{SWP}	2500
Aggregation P_{SWI}	2000

Experimental Setup

Scenario

We used the original parameters proposed for NSGA-II which are described in Table 3.

Table 3: Genetic algorithm parameters

Parameter	Value
Population size	100
Evaluations number	25000
Crossover operator	One point
Crossover probability	0.9
Mutation operator	Random mutation
Mutation probability	$\frac{1}{\text{task number}}$
Selection operator	Binary tournament

Experimental Setup

Scenario

In Table 4 are described Simulated Annealing parameters.

As cooling scheduling is implemented a decremental procedure: in each iteration the temperature value decrease of one unit.

Table 4: Simulated Annealing parameters

Parameter	Value
Initial temperature	100; 500; 1000; 1500
Acceptance procedure	Distance-based
Cooling scheduling	Decremental

Experimental Setup

Results

To solve allocation problem we implemented three different strategies using:

1. genetic algorithms only;
2. simulated annealing only;
3. genetic algorithms combined with simulated annealing.

For each strategy we evaluate three aspects:

- best solutions taking into account the two objectives separately;
- an example of Pareto front obtained;
- the experimental time complexity as a function of the number of tasks to allocate.

Experimental Setup

Results

The platform used to execute these experiments is an Intel I7 3630QM 2.4 GHz equipped with 8 GB RAM with OS Ubuntu 12.04, the jMetal framework for GA and a self-programmed tool for SA.

Each experiment was executed with input values presented in Table 5.

Table 5: Input values

Parameter	Value [W]
Server number	1536
Task number	[500:20000] with steps of 500
Same Experiment repetition	40

Experimental Setup

Results

Figures 3 and 4 represent the best time and power performant solutions found by the algorithm.

Figure 3: Best task completion times.

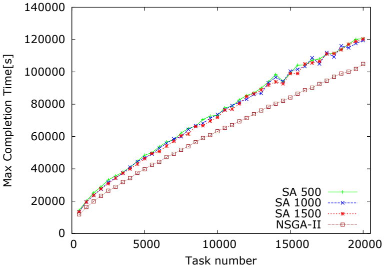
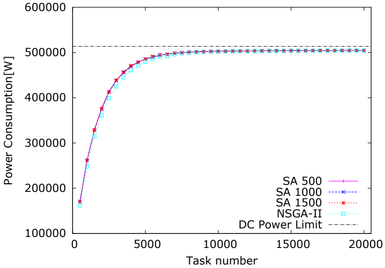


Figure 4: Best power consumption values.



Genetic Algorithm finds better solutions for the first objective and it is approximately the same for the second objective on average respect to Simulated Annealing.

Experimental Setup

Results

We describe now the relation among obtained solutions.

Figure 5 shows the Pareto front obtained by the algorithm from a single run computed on 8000 tasks.

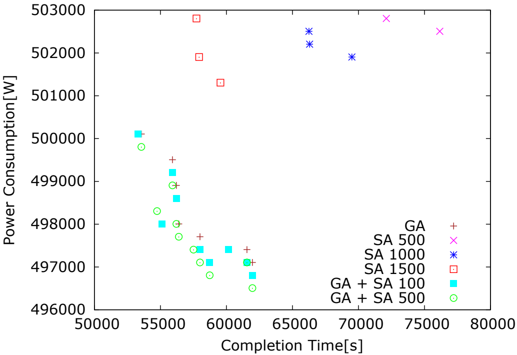


Figure 5: Pareto front obtained by a single experiment.

Experimental Setup

Results

Figures below show algorithm time complexity.

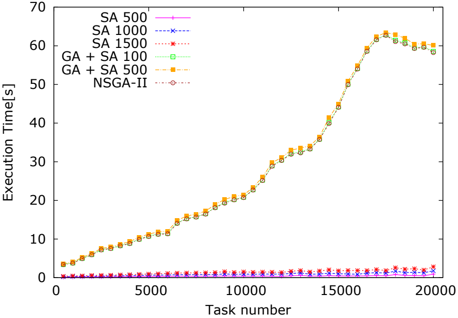


Figure 6: Execution time of the algorithm.

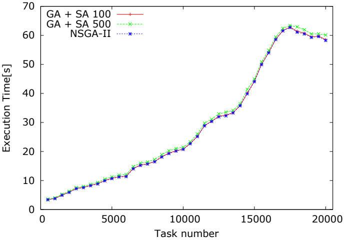


Figure 7: GA and GA + SA execution time

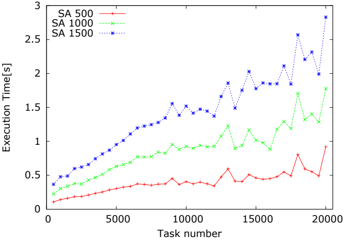


Figure 8: SA execution time

Contents

Conclusions and Future works

In this paper we introduce a power efficient resource allocation algorithm for cloud computing data centers which is based on genetic heuristics and simulated annealing.

Our problem is dealt such as a variant of the multi-objective constrained bin packing problem which is NP-Hard, solution set is non-convex and grows exponentially with the number of tasks to be allocated.

Furthermore it is not possible to apply real relaxation because objective functions are meaningful only into the integer domain. For these motivations we adopted two heuristics able to provide fast computed solutions.

Conclusions and Future works

When the execution of the algorithm is completed it becomes is possible to fine tune the trade-off between power consumption and execution time.

As main future work, our algorithm will be implemented directly on the SDN controller in order to perform both task allocation and network connection establishment and maintenance exploiting SDN centralizated controller functionalities and traffic probe to detect and monitor links state.

Other future works will focus on model adaptation to dynamic task allocation, account for internal communications, electricity cost and data center load factor.