ALC 2016/17
3^{rd} Project – VMC with CP

18/Nov/2016, version 1.0

Overview

The 3^{rd} ALC project is to develop an encoding for solving the Virtual Machine Consolidation (VMC) problem using a Constraint Programming (CP) solver (e.g. minizinc). The problem specification is the same as in the previous project.

Problem Specification

With the blooming of cloud computing, the demand for data centers has been rising greatly in recent years. Their energy consumption and environmental impact has become much more significant due to the continuous growth of data center supply. It is possible to reduce the amount of energy consumed by a data center by shutting down unnecessary servers and maintaining only a subset running that is enough to fulfill the resource demand. With recent advances in virtualization technology, it even became possible to consolidate the workload of multiple under-utilized servers into a single server. The VMC problem consists of determining a placement of a set of VMs among a set of servers that minimizes energy consumption.

Consider a set $J = \{j_1, j_2, ..., j_m\}$ of m jobs in a data center, where each job is composed of several virtual machines (VMs). For each job $j \in J$, let $V_j = \{v_1, v_2, ..., v_{k_j}\}$ denote the set of k_j VMs in job j. Furthermore, let $V = \bigcup_{j \in J} V_j$ denote the set of all VMs and for each VM $v \in V$, $cpu\_req(v)$ and $ram\_req(v)$ are the CPU and memory requirements of VM v, respectively.

Let $S = \{s_1, s_2, ..., s_n\}$ denote the set of n servers in the data center. For each server $s \in S$, we denote as $cpu\_cap(s)$ ($ram\_cap(s)$) the CPU (memory) capacity of server s. A server s is said to be active if at least one VM is placed in s. Otherwise, it is inactive. In the VMC problem the goal is to determine a placement of all VMs of V in the servers of S that minimizes the number of active servers. This placement is subject to the following constraints:

- each VM $v \in V$ is placed in exactly one server;
- for each server $s \in S$, the sum of the CPU (memory) requirements of the VMs placed in s cannot exceed its CPU (memory) capacity;

- if two VMs v and v' are part of the same job, i.e., $v \in V_j$ and $v' \in V_j$ for some $j \in J$, then v and v' may be required to be placed in different servers.

Project Goals

You are to implement a tool (**in Linux or MacOS**), or optionally a set of tools, invoked with command `proj3`, that should take one command line argument representing the scenario / characteristics of a problem instance.

This set of tools should use a CP solver to compute the makespan given the target graph and scenario, and output the VMs to be put in each server.

The tool **must** be executed as follows:

```
proj3 <scenario-file-name> > solution.txt
```

The tool **must** write to the standard output, which can then be redirected to a file `solution.txt`.

The programming language to use can either be C/C++, Java or Python.

File Formats

The input file has the following format:

```
<number of servers>
<server id> <CPU capacity> <RAM capacity>
...
<server id> <CPU capacity> <RAM capacity>
<number of VMs>
<job id> <VM index> <CPU requirements> <RAM requirements> <anti-collocation?>
...
<job id> <VM index> <CPU requirements> <RAM requirements> <anti-collocation?>
```

The output file has the following format:

```
o <optimum value>
<job id> <VM index> -> <server id>
...
<job id> <VM index> -> <server id>
```

Example

The input scenario is given by:

```
4
0 6 6
1 7 8
2 8 6
3 7 10
8
0 0 1 2 True
0 1 2 1 True
1 0 3 3 False
2 0 1 1 True
2 1 1 4 True
2 2 1 2 False
3 0 3 2 True
3 1 1 3 True
```

An example of a valid output is the following:

```
o 2
0 0 -> 3
0 1 -> 1
1 0 -> 1
2 0 -> 1
2 1 -> 3
2 2 -> 3
3 0 -> 3
3 1 -> 1
```

Note that VMs of the same job with the `anti-collocation?` constraint set to `True` cannot be placed on the same server; other VMs with `anti-collocation?` set to `False` can be placed on any server.

Additional Information

The project is to be made by groups of two students or one student only.

The project is to be submitted through the Fenix system. No submissions will be accepted using other methods such as email.

The project submission must be a zip file with all your code. The zip must also contain a text file with information on to build and execute your program, as well as a short text file describing the main features of your project. Finally, if your group number is `Y`, the name of the zip file should be `groupY.zip`.

The evaluation will be made taking into account correctness (80%) and efficiency (20%).

Th output described in this document should be strictly followed.

Project Dates

- Project published: 18/11/2016.
- Project due: 02/12/2016, 3PM.

Omissions & Errors

Any detected omissions or errors will be added to future versions of this document. Any required clarifications will be made available through the course's official website.

Versions

18/Nov/2016, version 1.0: Original version.