MFHS : Modular Framework for Heterogenous Systems

> Full Title :

MFHS : A **M**odular scheduling **F**ramework for **H**eterogeneous **S**ystem : From theoretical analysis to Real Cloud Testbed

> Collaboration with KHIAT Abdelhamid.

A PhD student in Algeria (CERIST)

> Going to be published in «Software: Practice and Experience" Journal

Contribution Highlighted

- Propose a framework which integrates a set of modules, and easier adaptable in any heterogeneous distributed environment,
- Propose a framework which can be used in both virtual and real distributed computing platforms,
- Propose a heuristic based and Min-Min and Max-Min, called MMin,
- Use of MFHS as theoretical tool using single personal computer,
- Basic evaluation of MFHS using Emulab test-bed,
- Expose wide experimentation using MFHS in a real Cloud environment based on OpenStack which compare the efficiency of a heuristic and algorithms though the computation of a set of metrics and parameters.

MFHS : Modular Framework for Heterogenous Systems

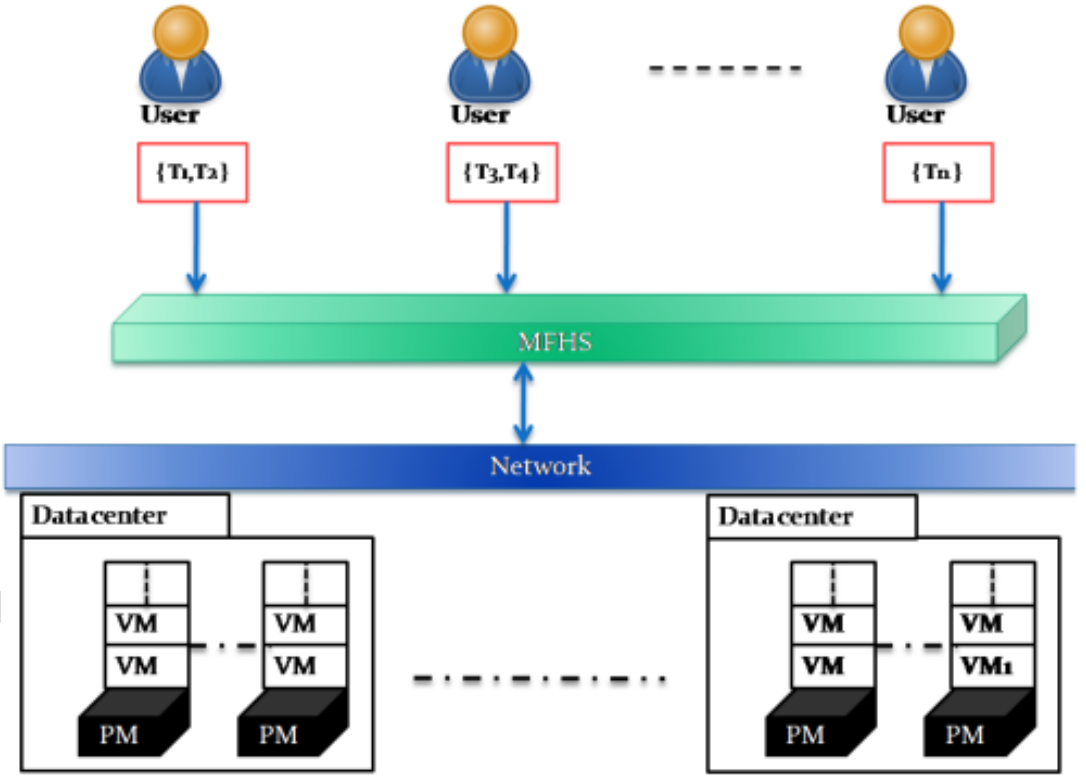
Outline

- Architecture, Modules description and QoS metrics
- Testbed environment
- Some experiments highlights
- Future Extension : Open discussion

Architecture and Modules description

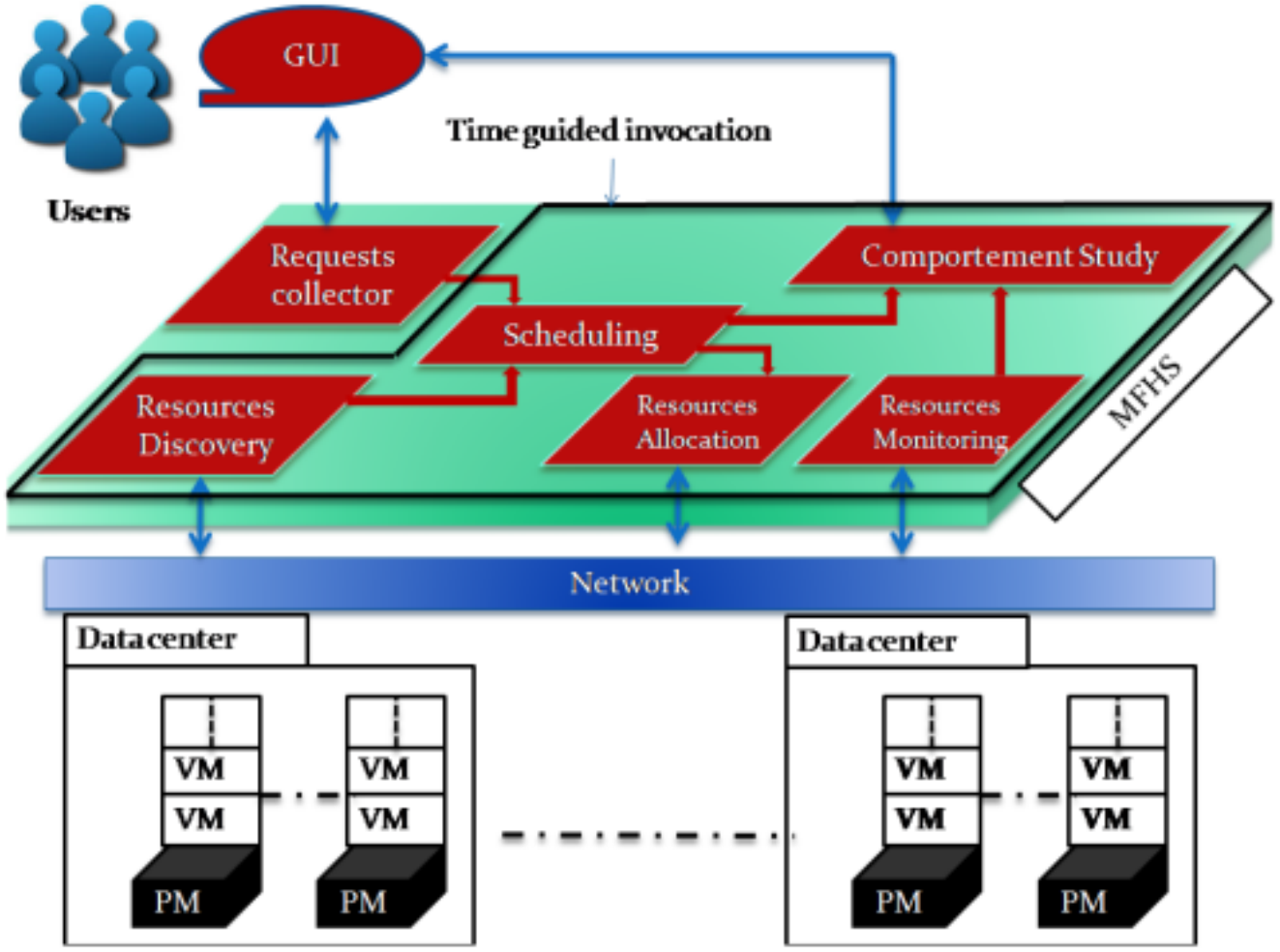
Features :

- Service provider point of view
- MFHS deployed as a Middleware
- VM can be hosted in different Cloud (Heterogeneous & Geographically distributed)
- Considered User tasks to be allocated can be heterogeneous as well
- MFHS supports any Task scheduling policy to optimize any QoS objective



Architecture and Modules description

- Each module fulfils a specific function
- Sequential and parallel are done to follow each task lifetime
- From request to end
- While motoring the platform and QoS objectives



Architecture and Modules description

Resources Discovery measures the Network and Disk throughput : upload & download

Request Collector collects data about the tasks execution requests

Scheduling module filters the computing resources regarding Ram, Disk and Cpu capacities. Computes some expected values (exec time, cost & energy)

Resource Allocation sends tasks and data, to compute node, waits task end, retrieves output data from node

Resource Monitoring stores measured execution metrics from nodes. Anomaly like connection loss can be detected.

Behavior study analyses all information monitored, compares real & theoretical values,

QoS Metrics

$$ET_i = \frac{SizeDTW_i}{DebitI_j} + \frac{SizeDTR_i}{DebitO_j} + \frac{SizeDTD_i}{DebitDk_j} + \frac{SizeDTU_i}{DebitUk_j} + TimeCPU_i$$

- Completion time :

$$CT_i = TStart_i + ET_i$$

- Average Resource utilization
- Cost

$$P = P_{min} + (P_{max} - P_{min})u$$

$$E_j = \frac{\sum_{t=1}^{n}(P_j)}{n} * T$$

$$E = \sum_{j=1}^{NbrR}(E_j)$$

Testbed Environment

MFHS allows both theoretical and real deployment

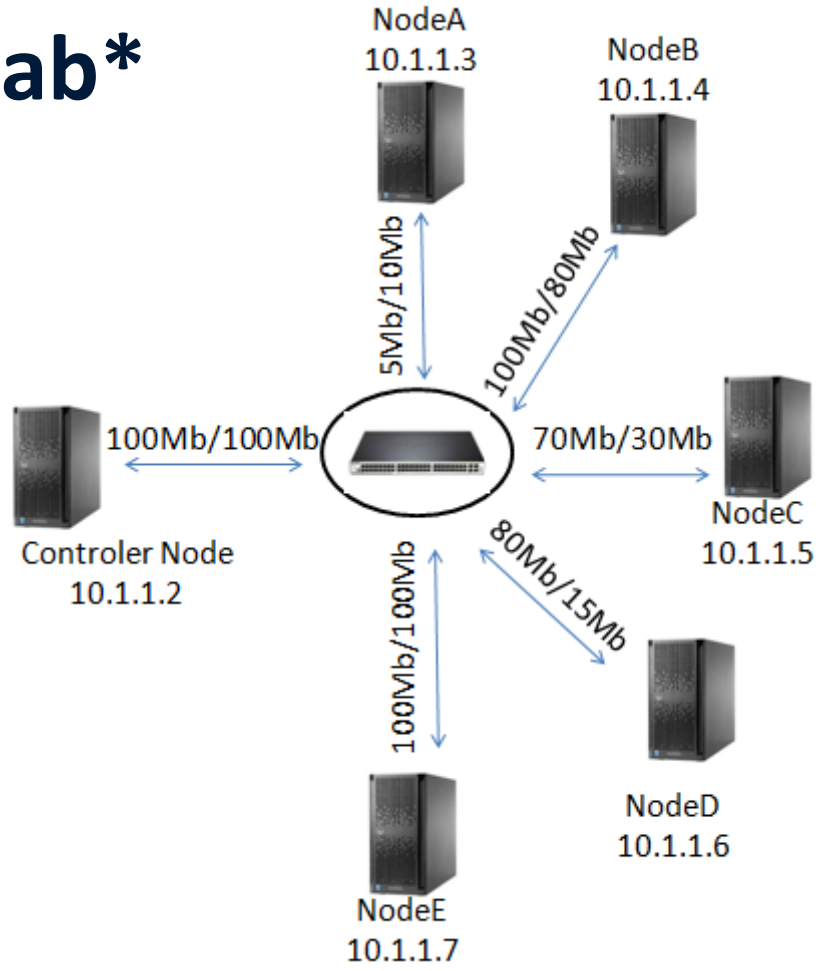
- **Theoretical analysis** is based on :
  - tasks needs, capacities nodes (real measures)
  - and Scheduling you want to study
  - Real needs, Real node capacities, Real Scheduling algorithm execution, BUT No real deployment
  - Allows to compute expected QoS value in real condition

Testbed Environment

EmuLab*

- Considered as one of the more traditional computing cluster environment

- Allows specifying the network topology and link characteristics



* [REF] Mike Hibler, Robert Ricci, Leigh Stoller, Jonathon Duerig, Shashi Guruprasad, Tim Stack, Kirk Webb, and Jay Lepreau. Large-scale virtualization in the emulab network testbed. In USENIX 2008 Annual Technical Conference, ATC'08, pages 113–128, Berkeley, CA, USA, 2008

Testbed Environment

Deployment & execution on real experimental Cloud Platform

- LAASNetExp Cloud platform (E35 server room)
  - OpenStack Kilo – 2015
  - Use of “extra_specs module” allowing to set up network throughput & disk R/W speed
  - 6 heterogeneous VM deployed where to tasks will be executed
  - MFHS deployed on the OpenStack controller node (iot server)

Deployment on Experimental Cloud

VM characteristics set up using "extra_specs" module are discovered through the Res. Discovery module

Host	Upload (Kb/S)	Download (Kb/S)	Write Disk (Mb/S)	Read Disk (Mb/S)
VM0	256	512	5	7
VM1	256	512	2	2
VM2	512	1024	8	4
VM3	128	1024	6	8
VM4	256	256	5	7
VM5	2048	128	3	3

Table 7: Resources characteristics

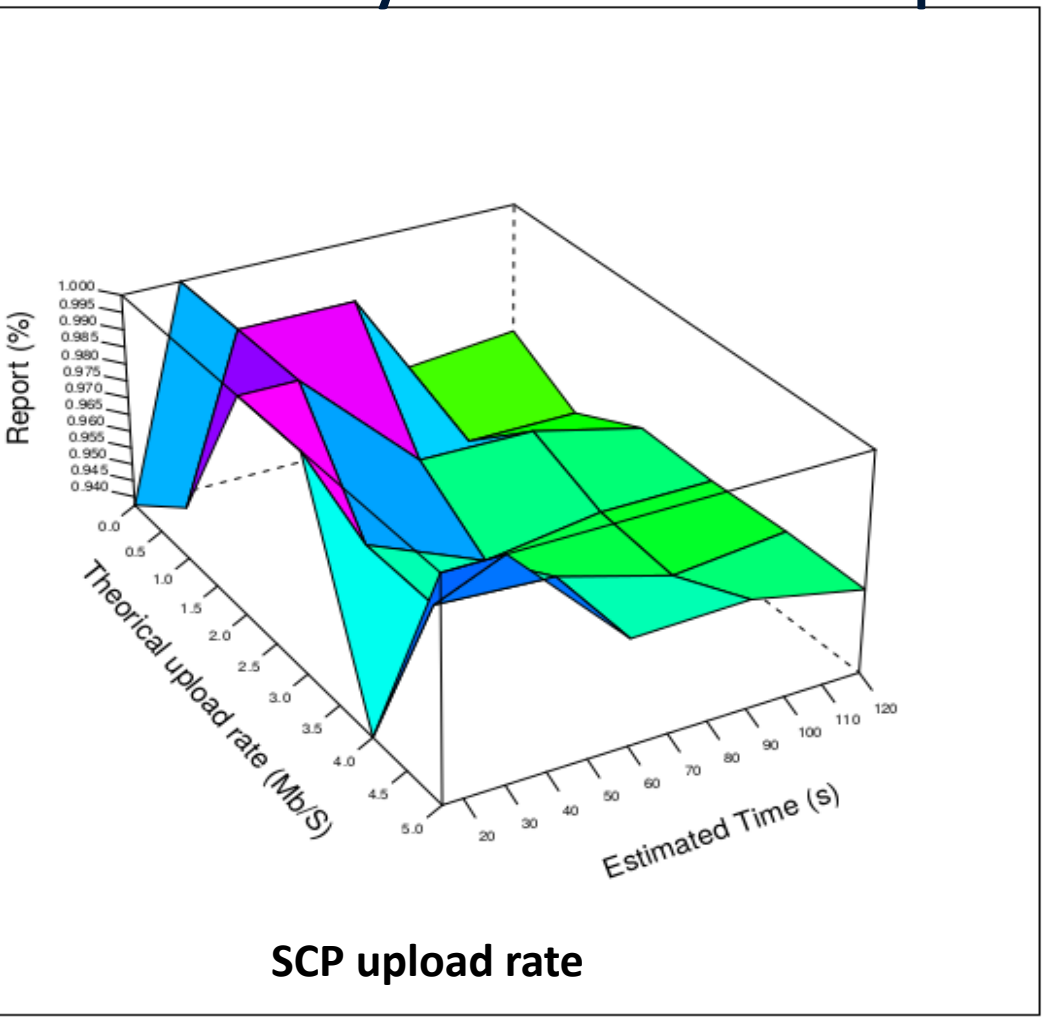
List of Tasks to schedule:

Task id	Upload FS (KB)	Download FS (KB)	Read FS (KB)	Write FS(KB)	vCPUs number	CPU Time(S)
T0	1000	2000	18000	11000	6	440
T1	2000	5000	15000	40000	6	180
T2	1000	15000	25000	12000	6	40
T27	4000	5000	75000	77000	6	180
T28	15000	15000	92000	56000	6	160
T29	8000	8000	20000	40000	6	90

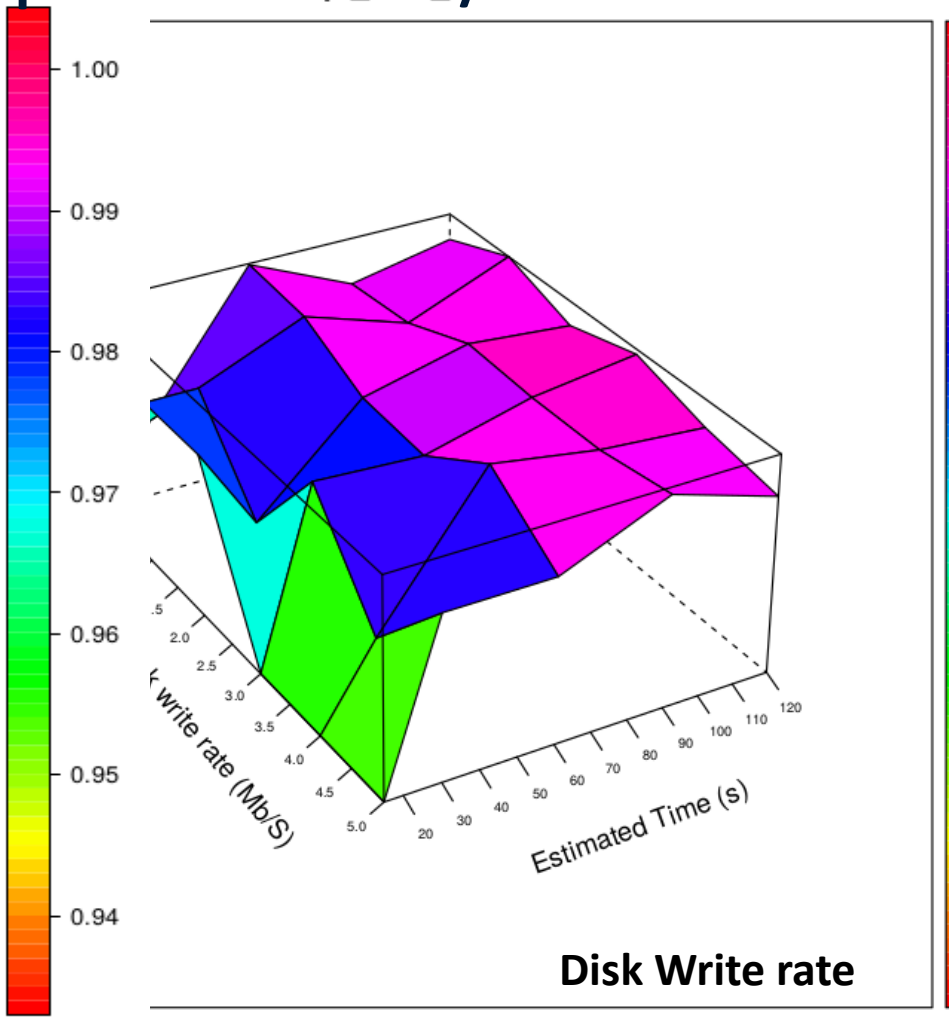
Table 8: openstack:Requests Description

Some experiments highlights

Res. Discovery module does multiple scp transfert and R/W on disk



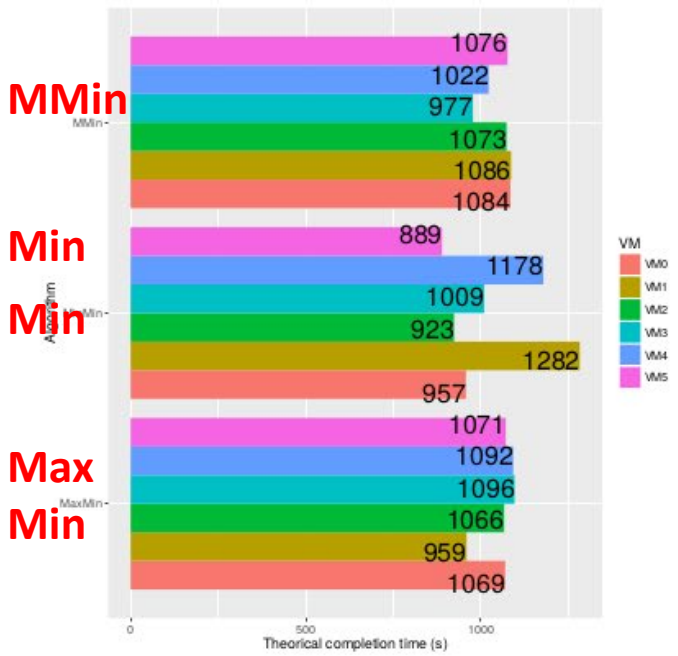
SCP upload rate



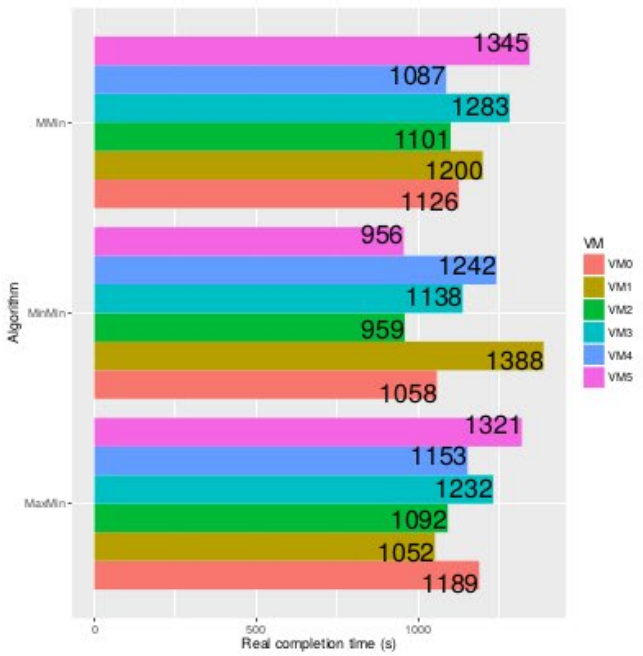
Disk Write rate

Some experiments highlights

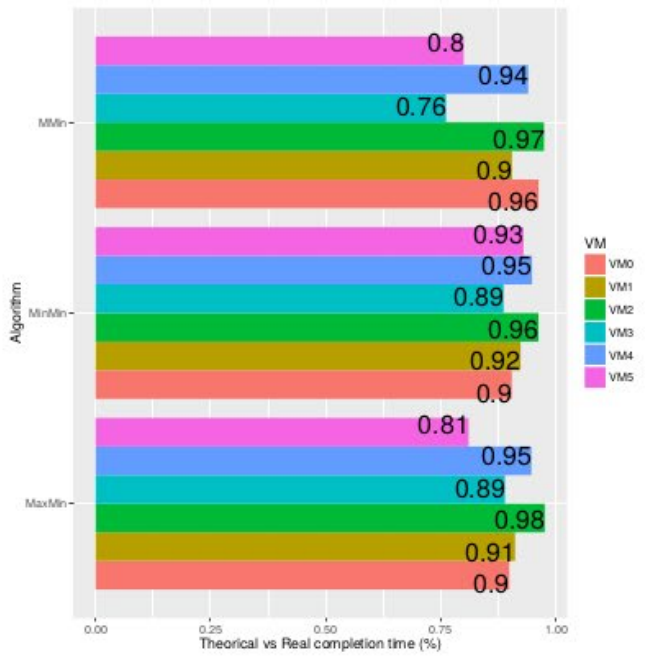
Example Completion Time analysis : Theoretical VS Cloud Exec. (**Algo : MMin, MinMin and MaxMin**)



Theoretical values



Cloud values



Ratio Theo/Cloud

Same comparisons for Energy (*"sensors" command used to get live delivered servers' Power)*

Future Extension : Open discussion

- Add on-line Task arrival
- Experimentation on the other LAAS Cloud (Feed by Renewable Energies)
- Extension to Fog Computing

Any else Idea ?