MFHS : Modular Framework for Heterogenous Systems

> Full Title :

**MFHS : A Modular scheduling Framework for Heterogeneous System : 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.
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

- Completion time:

- Average Resource utilization

- Cost

- Energy consumption

\[ ET_i = \frac{\text{SizeDTW}_i}{\text{Debit}I_j} + \frac{\text{SizeDTR}_i}{\text{Debit}O_j} + \frac{\text{SizeDTD}_i}{\text{Debit}Dk_j} + \frac{\text{SizeDTU}_i}{\text{Debit}Uk_j} + \text{TimeCPU}_i \]

\[ CT_i = T_{\text{Start}_i} + ET_i \]

\[ P = P_{\text{min}} + (P_{\text{max}} - P_{\text{min}})u \]

\[ E_j = \frac{\sum_{i=1}^{n}(P_j)}{n} \times T \]

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

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 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

<table>
<thead>
<tr>
<th>Host</th>
<th>Upload (Kb/S)</th>
<th>Download (Kb/S)</th>
<th>Write Disk (Mb/S)</th>
<th>Read Disk (Mb/S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM0</td>
<td>256</td>
<td>512</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>VM1</td>
<td>256</td>
<td>512</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>VM2</td>
<td>512</td>
<td>1024</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>VM3</td>
<td>128</td>
<td>1024</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>VM4</td>
<td>256</td>
<td>256</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>VM5</td>
<td>2048</td>
<td>128</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 7: Resources characteristics

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

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. \textbf{(Algo: MMin, MinMin and MaxMin)}

Theoretical values

Cloud values

Ratio Theo/Cloud

Same comparisons for Energy \textit{("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?