

Idle virtual machine detection in FermiCloud

Giovanni Franzini

September 21, 2012

Scientific Computing Division

Grid and Cloud Computing Department



FermiCloud & FermiGrid

- Cloud Computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet).
- A Cloud creates Virtual Machines (VM) upon request. A VM mimics a physical computer in all functionalities (e.g. it is accessible remotely using ssh connections, etc.).
- Here at Fermilab, the Grid and Cloud Computing department manages two distributed systems:
 - **FermiCloud** A private Cloud providing Infrastructure-as-a-Service for the Fermilab scientific stakeholders to manage dynamically allocated services, interactive and batch processing.
 - **FermiGrid** A distributed campus infrastructure that manages statically allocated compute and storage resources for batch processing.

Problem Statement

- FermiCloud and FermiGrid can share resources, so it is important to optimize their individual utilization to maximize available computing cycles.
- My work focussed on identifying and minimizing idle virtual machines on FermiCloud. An idle VM is an existing machine that it not currently providing computing services.
- If the resources occupied by idle Virtual Machines on FermiCloud were reclaimed, they could be used by FermiGrid to process scientific data.
- Problem:

How can we identify idle VMs in FermiCloud?

Project overview

- Find a way to identify idle VMs
 - Investigate the reuse of existing workload management systems for compute intensive jobs, such as Condor.
 - Develop a software product to address the requirements of the problem.
- Develop a mechanism to
 - Inform the FermiCloud management system (OpenNebula) about the idleness of the machine.
 - Suspend idle VMs.
 - Replace the suspended VMs with FermiGrid “worker nodes” to execute jobs in queue.

Identify an Idle VM

- In order to identify an idle VM, some components and activities of the machine must be monitored periodically, especially:
 - CPU idleness
 - Keyboard / Mouse usage
 - Network
 - I/O
 - Virtual Memory
- In Unix, some useful information can be found in special files (like `/proc/uptime`), or can be obtained using particular commands (such as `vmstat`) offered by the shell.
- Starting from these data, we can create several indexes to detect the VM status.

Identify an Idle VM

For monitoring the VMs, six indexes were defined:

1) CPU Idle Percentage

Based on `/proc/uptime` file.

2) Keyboard / pseudo-terminal idle time (from Condor code)

Based on `utmp` file.

3) Bytes tx and rx by the network interface

4) Iowait ticks

Percentage of time the CPU is idle and there is at least one I/O in progress (local disk or remotely mounted disk, NFS).

5) Context switches

6) Memory Paged in/out

Computing the Indexes

- CPU idle example.

- At step n:

Take times from `/proc/uptime` → `total(n)`, `idle(n)`

$$\text{delta_idle}(n) = \text{idle}(n) - \text{idle}(n-1)$$

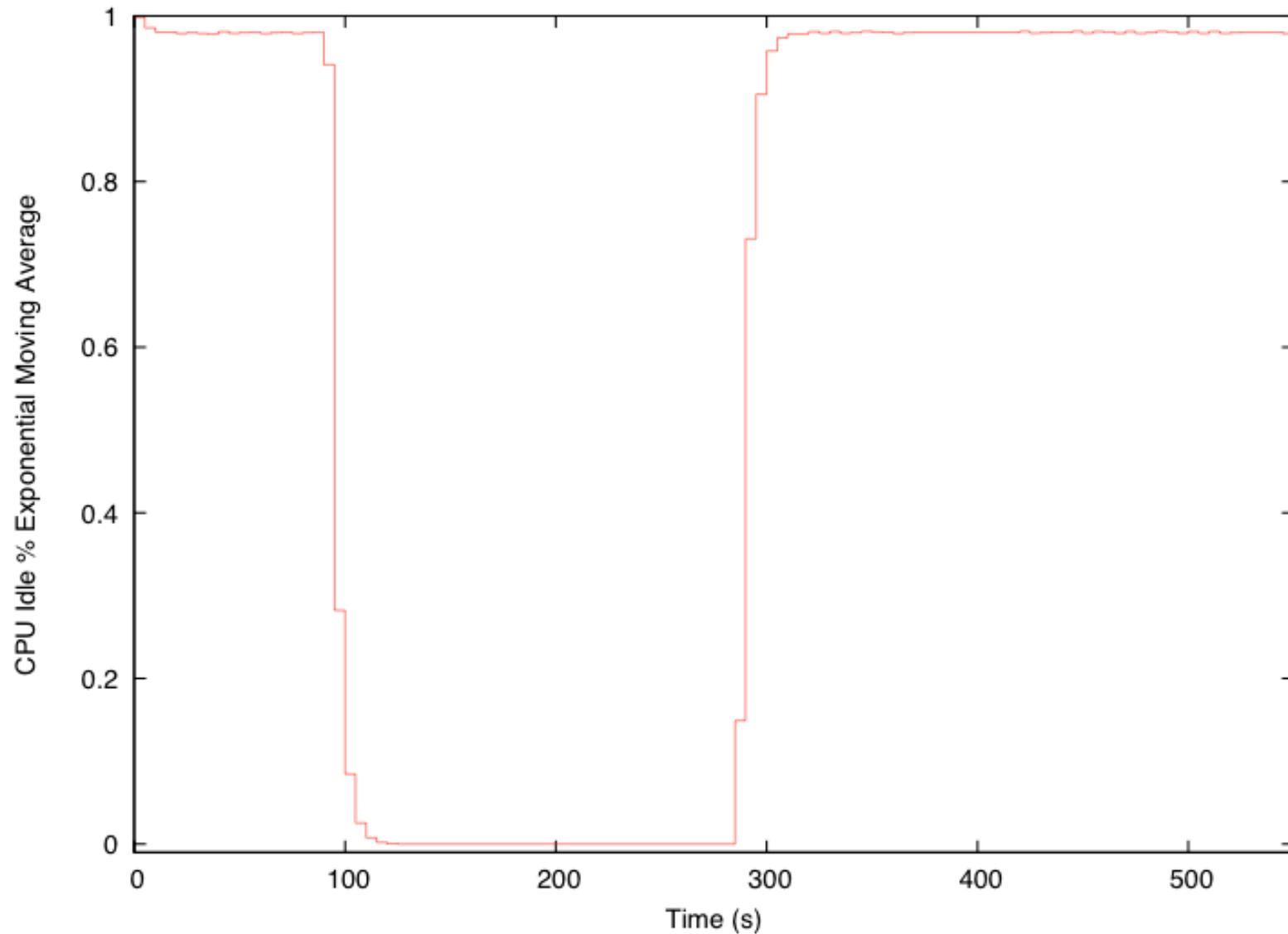
$$\text{delta_total}(n) = \text{total}(n) - \text{total}(n-1)$$

$$\text{idle_perc}(n) = \text{delta_idle}(n) / \text{delta_total}(n)$$

$$\text{avg_idle}(n) = (1-\alpha) * \text{idle_perc}(n) + \alpha * \text{avg_idle}(n-1)$$

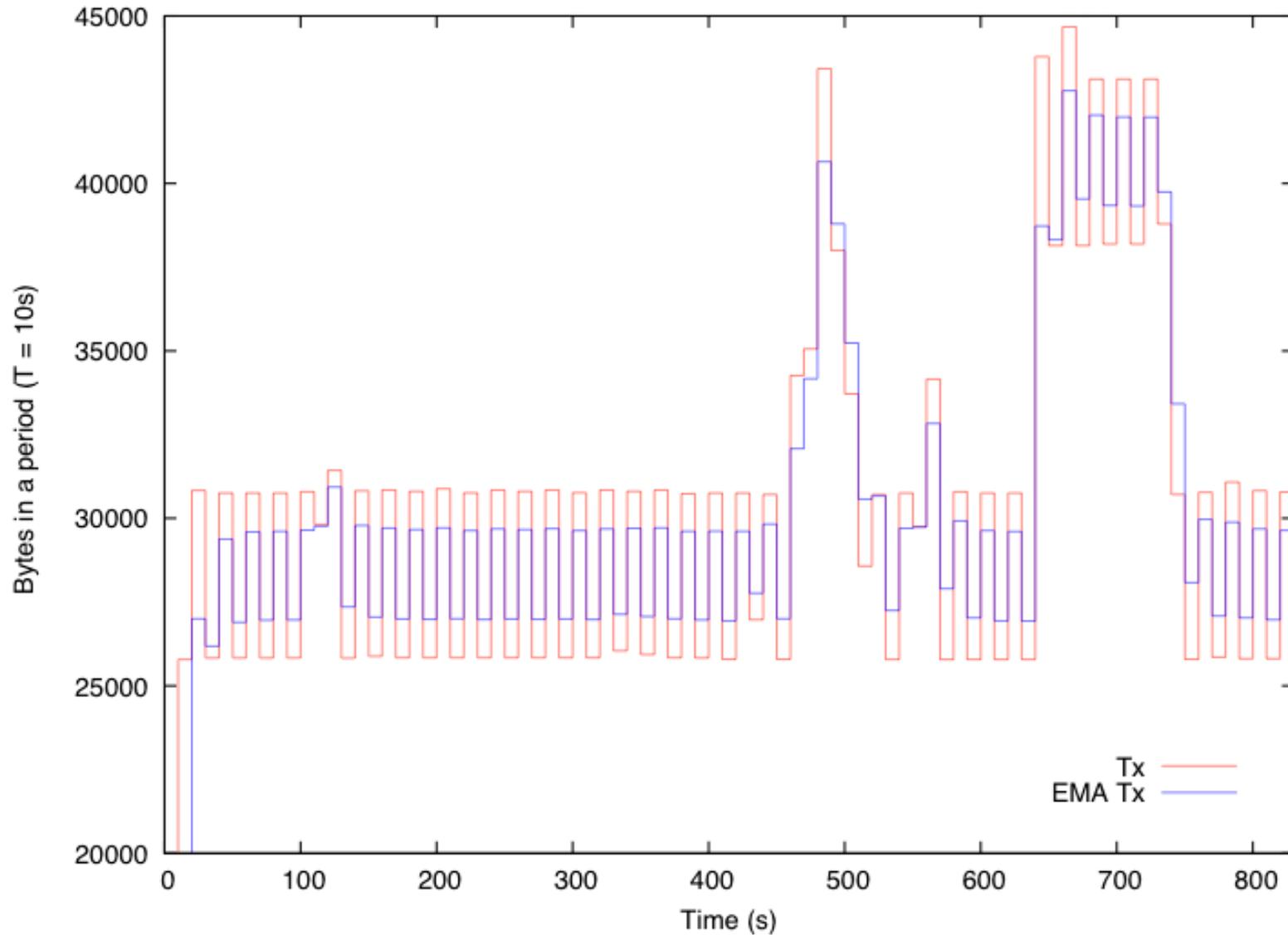
- `avg_idle(n)` is an EMA (Exponential Moving Average). It keeps track of the old values of the index, giving more importance to new values ($\alpha < 1/2$).
- The sample time for all the indexes is 60 seconds.

Index in Action: CPU Idle Prime Number Test ($T = 5s$)



Index in Action: Network Activity

Ping Test ($T = 10s$)



Indexes & Idleness Rules

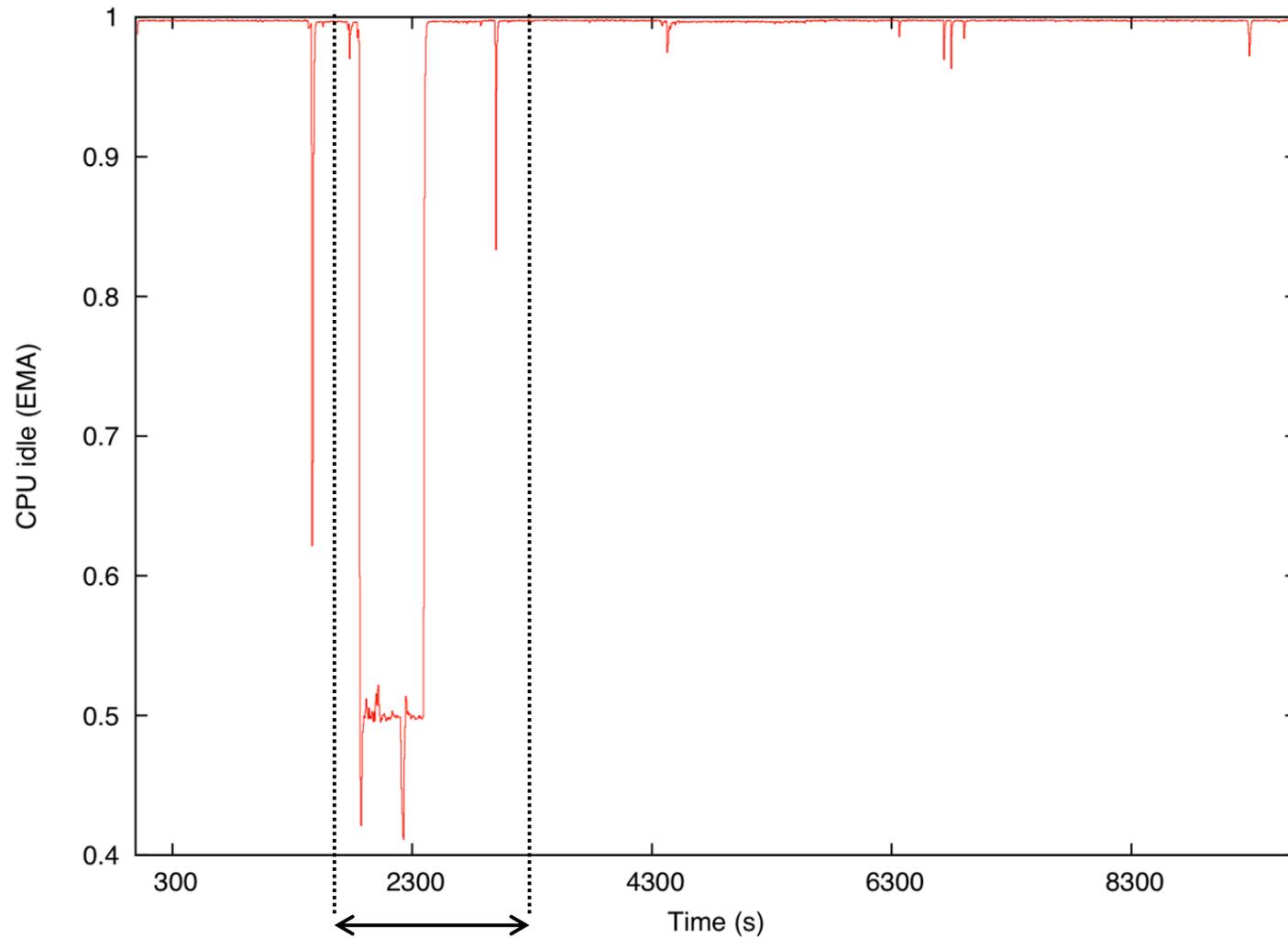
- Rules based on index values can define if a VM is currently idle. These rules are evaluated periodically (e.g. every hour).

- An example of rule could be:

```
vm_status = ( CPU_idle > 0.98 && pty_idle > 2*60  
             ) ? IDLE : NOT_IDLE
```

- In order to define these rules, “idle” thresholds for the defined indexes must be found.
- I considered two ways to define these thresholds:
 - experimentally (24 hours VM usage tests);
 - using ANFIS (Adaptive Network Fuzzy Interference System).

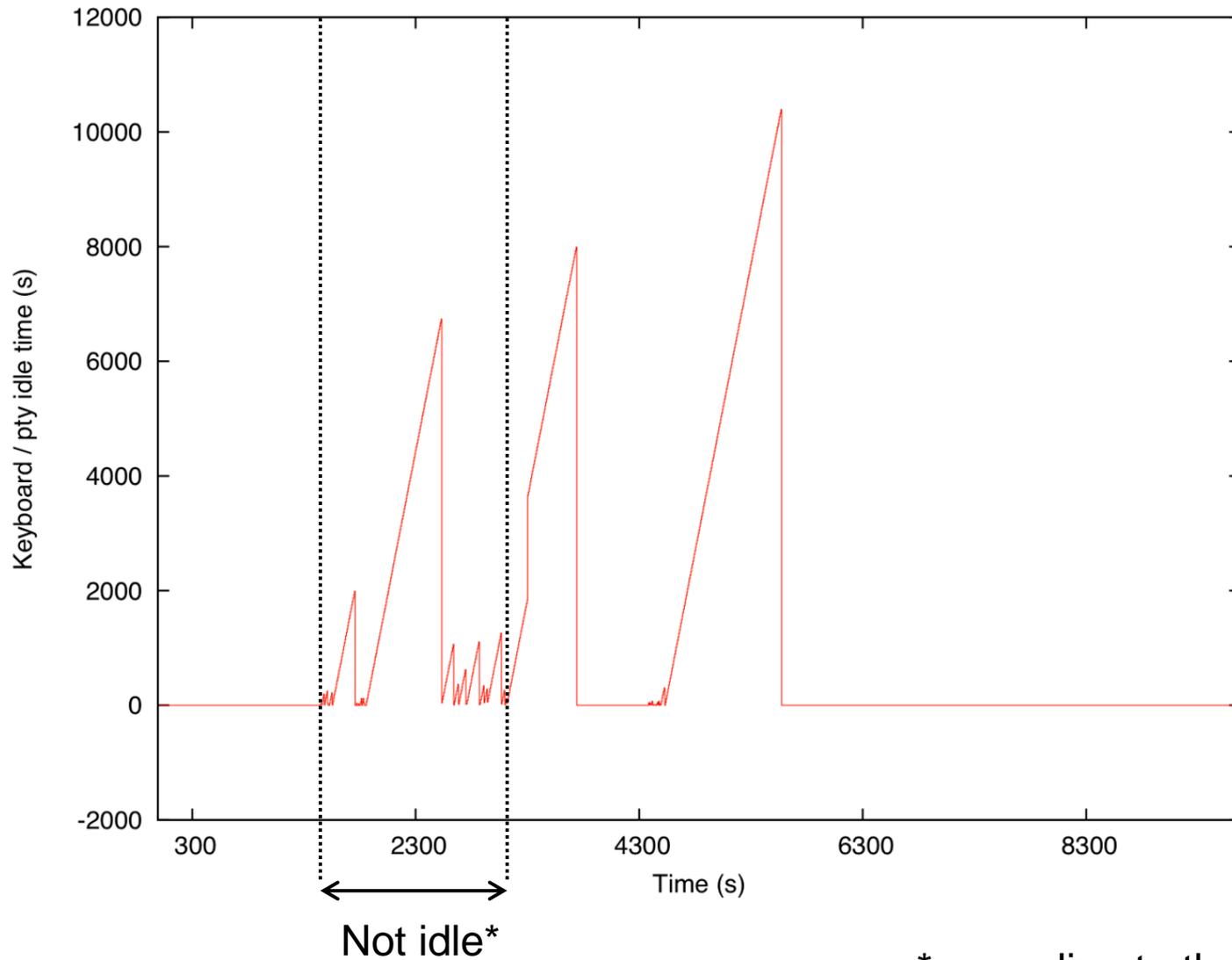
24 Hours VM Test: CPU Idle



Not idle*

* according to the tester report

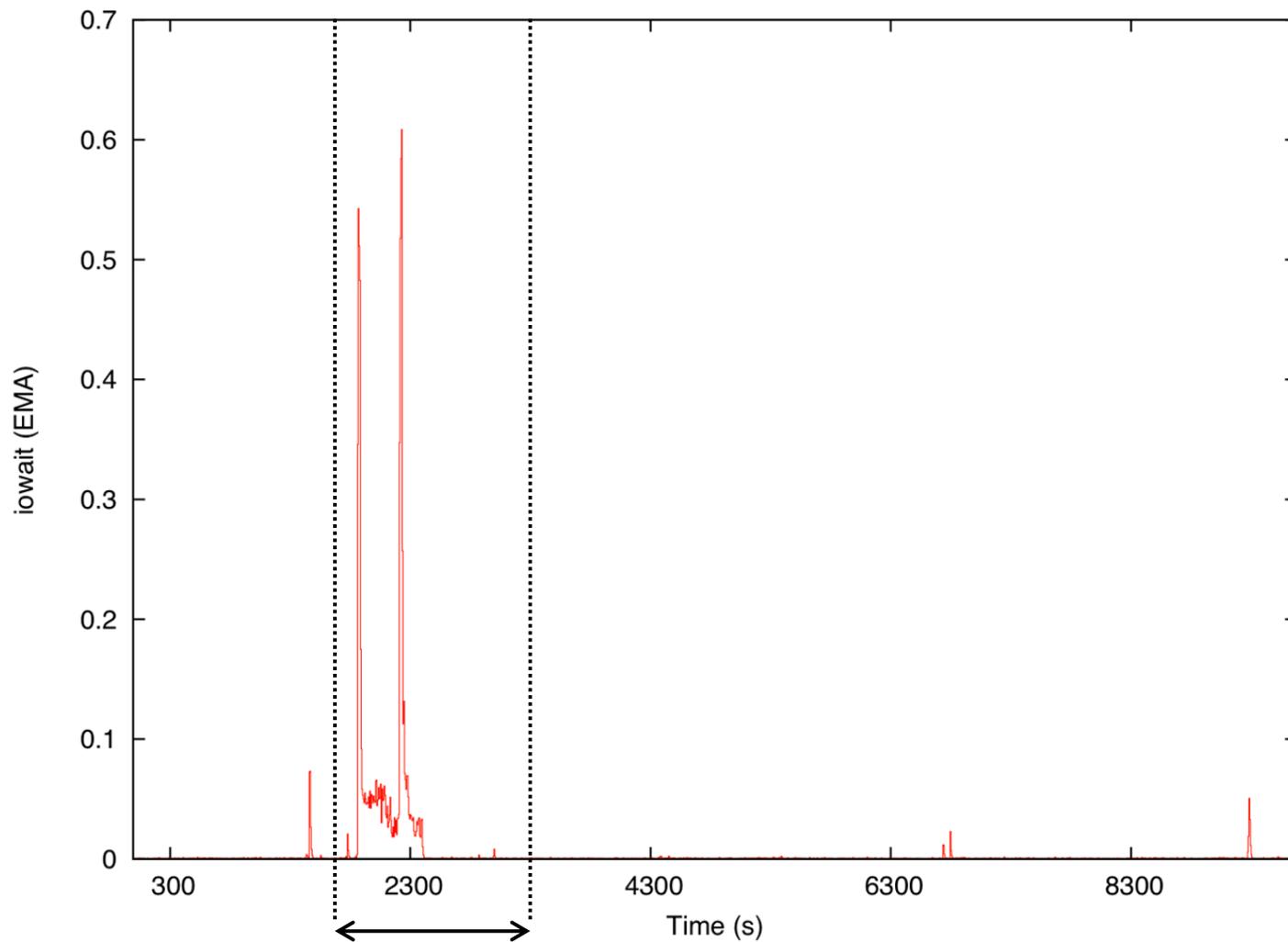
24 Hours VM Test: Keyboard / pty Idle Time



* according to the tester report

24 Hours VM Test

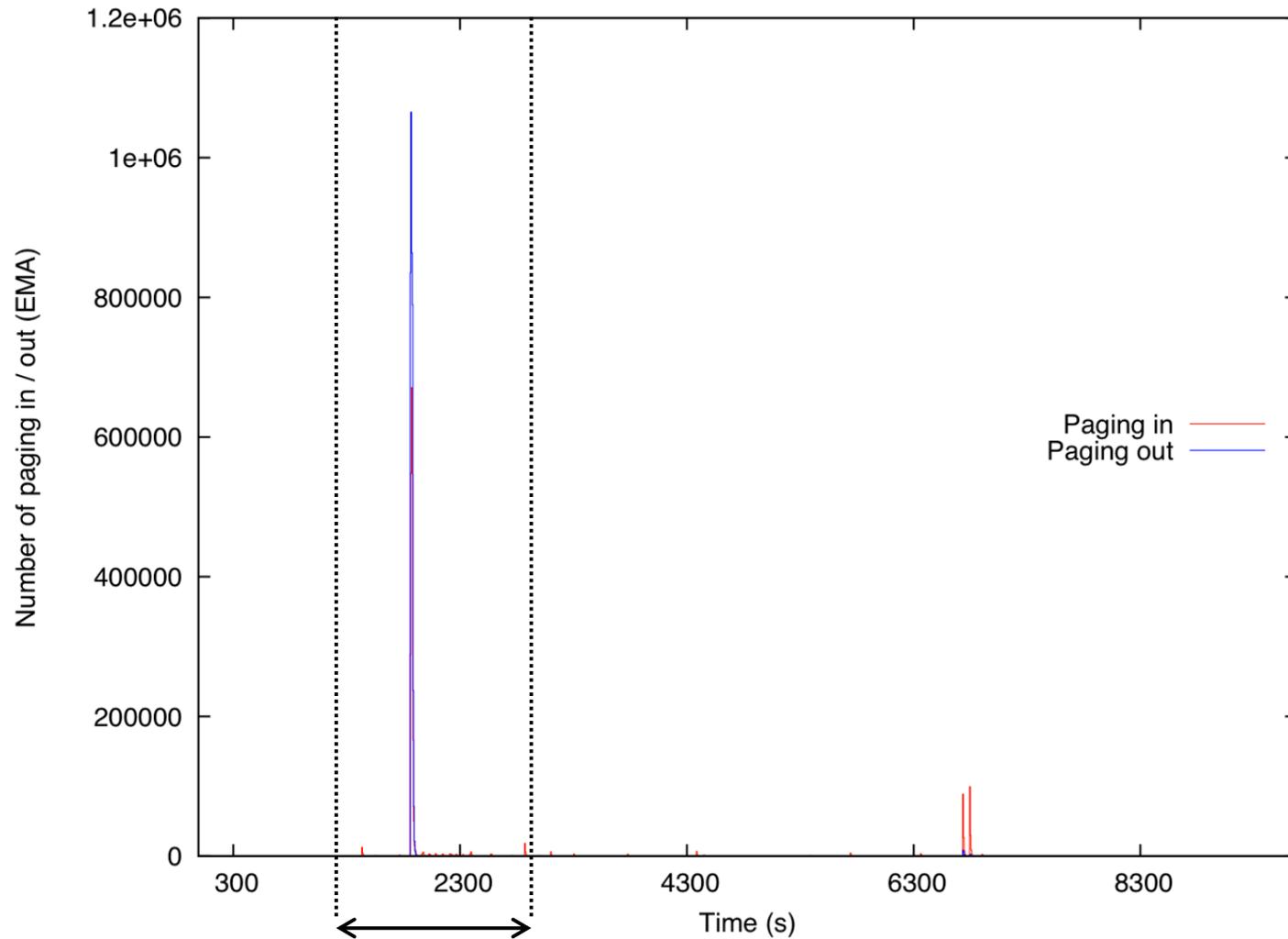
iowait



Not idle*

* according to the tester report

24 Hours VM Test Memory Paging in/out



Not idle*

* according to the tester report

Idleness rule

- An average of the indexes values recorded while the VM was idle, was performed. This gave us an idea of typical values for the indexes, when the VM is idle. The “idle” thresholds were defined starting from these averages.
- These values show an high standard deviation. Different VMs so may have very different values for the same index.

	CPU idle	Keyboard/ pty idle time	Bytes tx per period	Bytes rx per period	lowait	Context switches per period	Paging in per period	Paging out per period
Avg	0.996	3600	2089906	408922	0.0026	3012	62.43	449.19
Std.Dev.	0.0053	~	4147777	2151731	0.0015	3378	313.01	454.68

- Idleness rule used for the final tests is shown in the next slide (th_index_A is the idle threshold for index_A).

Idleness rule

```
SOMEONE_LOGGED = ( keyboard_pty != -1 );
```

```
KEYBOARD_USED = ( SOMEONE_LOGGED &&  
    keyboard_pty < th_keyboard_pty );
```

```
CPU_IDLE = ( cpu_idle > th_cpu_idle );
```

```
IOWAIT_IDLE = ( iowait < th_iowait );
```

```
PI_IDLE = ( paging_in < th_paging_in );
```

```
rule_A = ( SOMEONE_LOGGED && !KEYBOARD_USED && CPU_IDLE &&  
    IOWAIT_IDLE && PI_IDLE );
```

```
rule_B = ( !SOMEONE_LOGGED && CPU_IDLE && IOWAIT_IDLE && PI_IDLE );
```

```
vm_status = ( rule_A || rule_B ) ? IDLE : NOT_IDLE;
```

Final test results

- During the final test, the detector identified idle VMs with 90% accuracy.
- This is only a first encouraging result. Deeper analysis and improvements of the idle detector are needed.
- In particular:
 - The final test results come from the analysis of a very little pool of VMs (30). The majority of them were always idle, only a few of them were actually used during the test.
 - Idle thresholds must be improved, processing data coming from VMs providing different services (servers, worker nodes, etc.).
 - New ways to use the recorded indexes values may be investigated (use of different averages, recording of the last 60 minute values and processing of these data, etc.).

Conclusions

- Goal: optimize the usage of computing resources at Fermilab by detecting idle Virtual Machines (VM).
- Idle FermiCloud resources can be reclaimed and used by FermiGrid to run data processing jobs.
- I have implemented six indexes to expose the activity of a VM. These indexes are sampled periodically.
- An “idleness” rule was defined, to properly identify an idle VM.
- Running 24 hour tests: friendly users mark down when their VM is used, while my software record index values and define the VM status.
- A first prototype of the detector was created, with an accuracy of 90%.