

Idle virtual machine detection in FermiCloud

Giovanni Franzini

Fermi National Accelerator Laboratory

Scientific Computing Division

Grid and Cloud Computing

Abstract—FermiCloud is a private cloud providing Infrastructure-as-a-service services to Fermilab employees and users, to manage dynamically allocated services, interactive and batch processing. At Fermilab, this distributed computing system can share resources with FermiGrid, a distributed campus infrastructure that manages, conversely, statically allocated compute and storage resources for batch processing. In particular FermiGrid is used to run compute-intensive jobs related to experiments conducted here at Fermilab. The FermiGrid resources can be extended launching Virtual Machines configured as "batch nodes" running on FermiCloud. A VM mimics a physical computer in all its features. In order to maximize available computing cycles both for FermiCloud and FermiGrid, optimization of their individual utilization is required.

This work is focused on identifying and minimizing idle virtual machines in FermiCloud. An idle virtual machine is a machine that is not currently providing computing services. To increase the overall batch capacity of FermiCloud, these virtual machine can be retired and substituted with "batch nodes" VM accessible by FermiGrid.

To detect idle virtual machines, a set of indexes were created, to monitor and expose the activity of a virtual machine. After that, a rule to identify the status of the machine was defined. This last point was supported by experimental data coming from tests on real working virtual machines in FermiCloud. All the VMs in FermiCloud run Unix-like operative system.

I. INDEXES FOR IDLE DETECTION

In order to identify an idle virtual machine, some of its components and activities must be monitored periodically. Keyboard usage is among the most important activities. If someone is using the keyboard, we will be quite sure that the machine is currently in use. But when the keyboard has not been used for a long time (e.g. one hour), other indicators are

needed to understand if the machine is really providing computing services or not. The CPU idleness is a good parameter to figure out if there is some compute-intensive job running. A low value of this parameter suggests an active usage of the CPU by one or more process. But sometimes the CPU is idle because a process in execution is waiting for data coming from an I/O interface, or for gaining access to a file locked by someone else. The iowait index shows its potentialities in these cases. Iowait is one of the possible Unix machine status. A virtual machine enters into this status when the CPU is idle and there is at least one I/O operation in progress (it can involve both local disk and remotely mounted disk). Furthermore, parameters about virtual memory activity can be used to understand the degree of idleness of a machine. In particular, context switches and memory paging activity can tell us if the machine is loading memory pages for a new process, or other programs are resuming their execution. A virtual machine could also provide services to the network, it can be for example a web server. So, network activity of the VM must be observed, to properly define the status of the machine.

These considerations lead to the definition of six indexes (actually eight) for idle detection:

- 1) CPU idleness.
- 2) Keyboard / pseudo-terminal idle time.
- 3) Bytes received and transmitted by the VM's network interface.
- 4) Iowait ticks.
- 5) Context switches.
- 6) Memory paging in/out.

For all of them but one, keyboard / pty index, an exponential moving average (EMA) is computed. It is

a special weighted average defined as follows

$$\begin{aligned} S_1 &= I_1 \\ S_n &= (1 - \alpha) \cdot I_n + \alpha \cdot S_{n-1}, \quad n > 1 \end{aligned}$$

where S_n is the EMA and I_n is the index value, at step n . Generally $\alpha < 0.5$ in order to give more importance to the new index value. This average keeps track of its old values, influencing the current measure. In this way we try to do not miss activity occurred before the sampling instant. As a matter of fact, a value taken at the sampling instant, give us information about that activity in that specific instant. Otherwise, a weighted average that associates an heavier weight to current measure, like the EMA, is influenced by the old values of the index, taking in account past activity. How much the old values affect the average depends on the coefficient α . During the test it was set $\alpha = 0.3$, with a sampling period $T = 60$ s.

The defined indexes are discussed in the following sections.

A. CPU idle

The CPU idle index relies on the `uptime` Unix file (which can be usually found into the `/proc` directory). This file contains two values:

- 1) the total number of seconds the system has been up ($total_n$);
- 2) how much of that time the machine has spent idle ($idle_n$).

Starting from these information, it is quite simple to compute a percentage (actually, a number between 0 and 1) of system idleness, during a sample period.

$$\begin{aligned} \Delta idle_n &= idle_n - idle_{n-1} \\ \Delta total_n &= total_n - total_{n-1} \\ idlePerc_n &= \Delta idle_n / \Delta total_n \end{aligned}$$

After these steps, the EMA of $idlePerc$ is computed. A note about the idle time and the `uptime` file. Some Unix distributions (like Scientific Linux Fermi 5) intend this value as the amount of time the machine has spent idle. Thus, the idle time is always less than the total time. Other distributions (e.g. Scientific Linux

Fermi 6) count the idle time of every single CPU, and then put the sum of all of these into the uptime's idle time field. In this case, the idle time could be greater than the total time. Therefore it is necessary to know how the VM's operative system implements this counter.

B. Iowait, context switches and memory paged in/out

The Unix command `vmstat -s` shows to the user a series of counters updated by the kernel. Within this list, we can find information about the number of paging in and out, context switches and iowait ticks (i.e. the number of ticks the system has spent in the iowait status). A tick is an arbitrary unit for measuring internal system time. In Linux, the number of clock ticks per second can be obtained using `sysconf(_SC_CLK_TCK)`.

From the counters values, four of the eight indexes are computed. Context switches, paging in and out share the same formula. They are computed as the difference between the new and the old value at every sampling time. In this way we know how many context switches (or paging in and out) occurred in the previous period.

The iowait index needs a preliminary conversion of the sample period in ticks. After that, the percentage of time the machine spent in iowait status is computed (again, a number between 0 and 1). Let $iowait_n$ be the number of iowait ticks reported by the `vmstat` command at step n , and T_{ticks} the index sample period in ticks. Then the iowait index ($iowaitPerc$) is evaluated as follows.

$$\begin{aligned} \Delta iowait_n &= iowait_n - iowait_{n-1} \\ iowaitPerc_n &= \Delta iowait_n / T_{ticks} \end{aligned}$$

C. Keyboard / pty idle time

To obtain the amount of time since the keyboard was last used, some pieces of Condor source code were reused (in particular `src/condor_sysyapi/idle_time.cpp`). Condor is a workload management system for compute-intensive jobs. During the first phase of this

work, its possible use for performing the VM idle detection was investigated. It turned out that only the keyboard / pty idle time mechanism was useful for our purpose.

The extracted code uses the `utmp` file to obtain the number of seconds since the last activity detected from a keyboard or one of the pseudo-terminal (pty) associated to logged in users. In this way we know the instant of last detected interactive activity. If no one is logged in, the index value will be -1 . For this index EMA is not computed.

D. Network activity

Network activity of the VM is monitored using the information stored into `/proc/net/dev` file. Two indexes were created to count the number of bytes transmitted and received during a sample period, by the VM's network interface (usually `eth0`).

II. IDLENESS RULES

The VM status is defined processing the indexes values periodically. This period is different from the index sample period, as long as we can perform the idle detection test on a longer time scale (e.g. every hour). The idleness test is based on the evaluation of a logical rule (or a set of them) that can return only two values: idle or not idle. This rule may be very simple, and may be different from VM to VM, according to their tasks.

In order to write an "idleness rule", thresholds for the indexes must be defined. These thresholds should be related to typical indexes values when the VM is idle. Two possible ways to obtain them are:

- 1) experimentally, running 24 hours tests on VM where usage time are known (we know in every moment if the machine is idle or not);
- 2) training a neural network, in particular an ANFIS (Adaptive Network Fuzzy Interference System).

Both the solutions need data from VMs in use. The second one needs a good knowledge of the status of the machine in every moment, in order to perform a good training of the network. The trained network can

be used in two ways. The first one as an idle detector, implementing it in C++ for example. The second way need an extraction of the network weights in order to obtain the wanted thresholds. As a matter of fact, some of the parameters of the network, tuned during the training phase, are just the thresholds needed to evaluate the idleness rule. So after a first training of the network, these values may give an idea of the final "idle" threshold for the detector.

During this work, the first solution was chosen. A pool of friendly users working at the Fermilab Computing Division marked down when they use their VMs, while the detector was recording index values. Logs from these VMs were processed to obtain a first set of thresholds. The processing of these data consisted in computing an average of the indexes values recorded while the machine was idle. The results are shown in Table I. Data coming from 30 FermiCloud VMs were used. Keyboard / pty idle time threshold was set at 1 hour.

Table I
IDLE THRESHOLDS OBTAINED PROCESSING TESTS RESULTS.

	Average	Std. Deviation
CPU idle	0.996	0.0053
Keyboard / pty idle time	3600	~
Bytes tx per period	2089906.54	4147777.63
Bytes rx per period	408922.42	2151731.32
iowait	0.0026	0.0015
Context switches per period	3012.74	3378.73
Paging in per period	62.43	313.01
Paging out per period	449.19	454.68

As we can see, these values are characterized by an high standard deviation, pointing out the difference of values recorded among the different VMs. These lead me to the definition of a simple idleness rule, that uses only few indexes. In particular, observing some of the collected logs, the paging in index has a value different from zero when the user starts the execution of a process on his virtual machine. Otherwise its value is always zero, except in some isolated cases where was found a periodic pattern (periodically the paging in index showed a value different from zero).

The algorithm used to implement the idle detection test is the following.

```

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;

```

The thresholds used for the tests are listed in Table II. They were obtained combining the data contained into the logs coming from the 24 hours tests, and the index average values when the VM is idle, shown before. The rules were evaluated every 1 hour.

Table II
THRESHOLDS FOR TEST PHASE

th_keyboard_pty	3600
th_cpu_idle	0.96
th_iowait	0.01

The rule is actually divided into two sub-rules, to differentiate cases when someone is actively using the machine (keyboard activity is detected) from case where there is no interactive activity. After that, CPU idleness, iowait ticks and memory paging in are considered.

III. TEST RESULTS AND CONCLUSIONS

Test results are shown in Table III. As we can see the idle detector identifies correctly the VM status 90% of the time.

Table III
IDLE DETECTOR TEST RESULTS.

VM hostname	Hit
fermicloud010	16/27
fermicloud049	25/27
fermicloud053	26/27
fermicloud064	25/27
fermicloud089	26/27
fermicloud092	25/27
fermicloud101	25/27
fermicloud108	25/27
fermicloud130	25/27
fermicloud141	26/27

However, several things must be pointed out. The majority of the machine used for these tests were always idle. Only for a few number of virtual machines, detailed information about their usage time were available. Therefore, further tests are required, in order to understand the real performance of this algorithm.

A series of new tests may also be useful to better define “idle” thresholds. In particular a pool of virtual machine with different tasks could be an useful test group to understand the difference between the index values recorded on different machines. In this way, different rules for different kind of machines might be defined.

Moreover, the index values may be processed in a different way. In particular, instead of compute the EMA at every sampling time, index values may be recorded in a log file. Then, when it is needed to perform the idleness test for the virtual machine, data collected into this file may be processed to foresee the machine activity in the future, or to understand what the machine did during the previous period. So different use of these indexes is possible.

All tests use $\alpha = 0.3$ as coefficient for the EMA. Further tests could be performed to tune this parameter to satisfy different requirements. Different values for α changes the behavior of the index average, increasing or reducing its “decadence” time.