Introduction to Performance Monitoring and Analysis of Linux

Oliver Benke, Jack Holloway

August 29, 2001

Contents

1 What you should know about the available metrics **2**
1.1 General RMFPMS/Linux background information 2
1.1.1 UNIX and Linux 2
1.1.2 Difference between RSS and virtual memory size . . . 3
1.1.3 s/390 Linux caveats 3
1.2 Metrics: Introduction 3
1.3 Resource `LINUX_SYSTEM` 4
1.3.1 rate of context switches (per second) 4
1.3.2 rate of processes created (per second) 4
1.4 Resource `LINUX_MEMORY` 4
1.4.1 % swap space used 4
1.4.2 cache memory in MB 4
1.4.3 free swap space in MB 4
1.4.4 major page fault rate by process 5
1.4.5 major page fault rate by process (including children) 5
1.4.6 memory used for buffers in MB 5
1.4.7 memory used in MB 5
1.4.8 minor page fault rate by process 5
1.4.9 minor page fault rate by process (including children) 5
1.4.10 number of pages swapped in per second 6
1.4.11 number of pages swapped out per second 6
1.4.12 resident set size (RSS) in MB by process 6
1.4.13 shared memory in MB 6
1.4.14 size of swap space in MB 6
1.4.15 total memory size in MB 6
1.4.16 used swap space in MB 6
1.4.17 virtual memory size by process 7

1.5 Resource LINUX_NETWORK 7
1.5.1 bytes received per second 7
1.5.2 bytes received per second by network device 7
1.5.3 bytes transmitted per second 7
1.5.4 bytes transmitted per second by network device . . . 7
1.5.5 packets received per second 7
1.5.6 packets received per second by network device 8
1.5.7 packets transmitted per second 8
1.5.8 packets transmitted per second by network device . . 8
1.5.9 receive errors per second 8
1.5.10 receive errors per second by network device 8
1.5.11 transmit errors per second 8
1.5.12 transmit errors per second by network device 8
1.6 Resource LINUX_CPU 8
1.6.1 % CPU idle time 8
1.6.2 % CPU idle time by processor 9
1.6.3 % CPU time in kernel mode 9
1.6.4 % CPU time in kernel mode by process 9
1.6.5 % CPU time in nice mode 9
1.6.6 % CPU time in user mode 9
1.6.7 % CPU time in user mode by process 9
1.6.8 % CPU time in user mode by processor 10
1.6.9 % CPU time total by process 10
1.6.10 % CPU total active time 10
1.6.11 % CPU total active time by processor 10
1.6.12 accumulated CPU time in kernel mode by process . . 10
1.6.13 accumulated CPU time in user mode by process . . . 10
1.6.14 load average 10
1.6.15 nice value by process 10
1.7 Resource LINUX_FILESYSTEM 11
1.7.1 % free by file system 11
1.7.2 % used by file system 11
1.7.3 % of space free 11
1.7.4 % of space used 11
1.7.5 available (in 1 MB blocks) by file system 11
1.7.6 size (in 1 MB blocks) by file system 11
1.7.7 total size of all file systems (in 1 MB blocks) 11
1.7.8 total space available (in 1 MB blocks) 11

Chapter 1

What you should know about the available metrics

1.1 General RMFPMS/Linux background information

This document assumes some level of UNIX-like operating system knowledge. Here, we have attempted to present a number of difference between generic UNIX operating systems and Linux in particular. We also try to enumerate differences in kernel versions with respect to metric value collection.

1.1.1 UNIX and Linux

Linux is a UNIX-like operating system. That is, Linux complies (the word "complies" being a bit of a misnomer in the case of Linux) with "UNIX standards" – these being either de facto standards or genuine standards. However, there are a few implementation difference between Linux and other UNIX OSes.

- Threads are processes : In Linux, threads are basically processes.
- Buffer cache : All the free memory on a Linux machine is allocated into the buffer cache. If memory is needed by a process, memory is released from the buffer cache.

1.1.2 Difference between RSS and virtual memory size

A Linux process has a resident set size (`RSS`) and virtual memory associated with it. The `RSS` is defined as the memory occupied by the process code, data, and shared libraries.

The virtual memory associated with the process has a slightly more complicated definition. The Linux kernel has an address space which is mapped onto physical memory (`RAM`). The Linux kernel allocates a chunk of this virtual memory address space to the process. When the process in question wishes to access a piece of its code or data, a kernel request for a piece of virtual memory is made. The kernel then takes care of mapping the request virtual memory address into physical memory. Here is the real point: the data that now resides in the physical memory space was not there until a request for the corresponding virtual memory area was made. In this way, the process code, etc. doesn't have to be completely loaded into physical memory all the time. Therefore, Linux has a large amount of virtual memory space marked as available and allocated to a process, but much less actual memory in use.

1.1.3 s/390 Linux caveats

There are a few metrics which simply cannot be collected with Linux for s/390 running a 2.2.x version Linux kernel. Most notably, the `DASD` and network metrics are unavailable under a 2.2.x kernel.

1.2 Metrics: Introduction

This section provides a more in-depth look at the various metrics available in RMFPM/RMFPMS. Besides describing what the metrics explicitly measure, we also discuss why one would want to use these metrics in one's performance analysis.

1.3 Resource `LINUX_SYSTEM`

1.3.1 rate of context switches (per second)

This metric measures the number of context switches per second. A context switch happens when the operating system's scheduler stops (pauses) execution of one proc. and begins (continues) execution of another process.

1.3.2 rate of processes created (per second)

This metric measures the number of processes created per second. If this number is high, then a large number of processes are being started. Each time a process is created, there is some amount of overhead associated with this creation; this overhead can become a performance problem if the rate of process creation become large.

1.4 Resource `LINUX_MEMORY`

1.4.1 % swap space used

The percentage of a machine's swap space used. If swap space is being heavily used, then the operating system is increasingly having to read and write from the disk to execute memory reading and writing routines. This can have a significant impact on performance.

1.4.2 cache memory in MB

This metric measures the memory used for cache in MB.

1.4.3 free swap space in MB

Again, this is related to virtual memory. To achieve the best performance, we would like to have all the processes memory segments in physical memory (RAM) and not being swapped to/from disk. Therefore, a large value here is usually good in terms of machine performance.

1.4.4 major page fault rate by process

This metric gives us the number of major page faults by process per second. We define a major page fault to simply be any page fault where disk access is involved. Again, we would like to have as little disk access as possible. A large value here usually corresponds to poor performance.

1.4.5 major page fault rate by process (including children)

Similar to the above metric, except the inclusion of major faults by all children processes. Again, a large value usually corresponds to poor performance.

1.4.6 memory used for buffers in MB

The Linux kernel maintains a disk cache designed to relieve processes from waiting on relatively slow disk access. When free memory becomes low, buffer frames are released. Ideally, we would like the amount of free memory to stay high. Therefore, a small value here corresponds to buffer frames being released. Perhaps there isn't a performance problem yet, but swapping to/from virtual memory could start soon.

1.4.7 memory used in MB

This metric measures the number of MB of memory used (including the buffer caches). If the amount of memory used is near the total amount of memory (RAM) available on the machine, the machine may have poor performance.

1.4.8 minor page fault rate by process

The number of page faults *without* disk access per second by process.

1.4.9 minor page fault rate by process (including children)

Again, like the above metric, but including all children processes.

1.4.10 number of pages swapped in per second

This metric reports the number of pages of virtual memory swapped to disk per second. Generally, any non-zero value here indicates poor performance.

1.4.11 number of pages swapped out per second

Like the above metric, only referring to pages swapped out.

1.4.12 resident set size (RSS) in MB by process

The resident set size refers to the amount of memory used by the process. This includes the code, data, and shared libraries. If a process's RSS is growing rapidly, this could be problematic – the machine may run out of physical memory and begin using swap space.

1.4.13 shared memory in MB

This metric measures the amount of memory in MB that can be used by more than one process. This isn't usually a good indication of poor performance – more information is needed. For instance, a database server may have several processes running with one very large shared memory area. If the total amount of memory collectively used by the processes and this shared memory area is larger than the physical memory size (RAM), then there may be a performance problem.

1.4.14 size of swap space in MB

This simply measures the size of the swap space/swap file on the machine.

1.4.15 total memory size in MB

Total amount of memory available (RAM) on the machine.

1.4.16 used swap space in MB

This metric reports the amount of swap space used at the end of each cycle time. This is usually a very good indicator of performance. A high value corresponds to large amount of swap space being used – this means that there are many disk access, and thus poor performance.

1.4.17 virtual memory size by process

Size of the virtual memory in bytes by process at the end of a time cycle. This is normally a very big number, but most parts of this virtual memory area are often left unused (not even paged in). Therefore, this metric is usually not a good performance indicator (on its own).

1.5 Resource `LINUX_NETWORK`

1.5.1 bytes received per second

This metric reports the number of bytes received per second by the host – this is the sum of all received bytes on all network devices per second.

1.5.2 bytes received per second by network device

This metric is an enumeration of the network devices and the number of bytes received per second.

1.5.3 bytes transmitted per second

Similar to *bytes received per second*.

1.5.4 bytes transmitted per second by network device

Similar to *bytes received per second by network device*.

1.5.5 packets received per second

This metric is the total of all packets received by the machine (through all network devices) per second. Though the size of a packet is not constant, this can give you an idea about the network processor and CPU utilization caused by network traffic. For instance, if you have many small network packets, you don't have a bandwidth problem, but rather a problem with network processor speed.

1.5.6 packets received per second by network device

This metric is similar to *packets received per second* above – except this metric is enumerated by the network devices.

1.5.7 packets transmitted per second

Similar to *packets received per second.*

1.5.8 packets transmitted per second by network device

This metric is similar to *packets received per second by network device.*

1.5.9 receive errors per second

Number of receive errors per second for the entire machine (over all connected network devices).

1.5.10 receive errors per second by network device

This metric is similar to the metric above, except the number of errors is broken down by network device.

1.5.11 transmit errors per second

Similar to the metric *receive errors per second.*

1.5.12 transmit errors per second by network device

This metric is similar to the *receive errors per second by network device.*

1.6 Resource `LINUX_CPU`

1.6.1 % CPU idle time

This metric measures the amount of time that the CPU is idle (averaged over all CPUs). This can be used as an effective performance metric. If this metric has a low value, then the machine is not being utilized 100%. If the value is very large, then the CPUs are being fully utilized. In some cases, one should also look at disk access/swap space usage stats during periods of high CPU utilization as disk access can lead to high CPU usage.

1.6.2 % CPU idle time by processor

This metric is simply the above *% cpu idle time* enumerated by CPUs.

1.6.3 % CPU time in kernel mode

Percentage of CPU idle time in kernel mode (averaged over all CPUs). When a machine is in kernel mode, then it is executing code inside the kernel (system calls, scheduler routines, etc.).

1.6.4 % CPU time in kernel mode by process

This metric is similar to the above *% CPU time in kernel mode* metric, enumerated by process.

1.6.5 % CPU time in nice mode

This metric measures the amount of time the machine spends in “nice” mode (averaged over all CPUs. We say a process is “nice” if its scheduling priority is lower than normal. If the superuser has increased the scheduling priority of some processes to values higher than normal, the process is no longer classified as running in “nice” mode.

1.6.6 % CPU time in user mode

This metric measures the amount of time spent executing code of user-space processes (averaged over all CPUs).

1.6.7 % CPU time in user mode by process

Similar to *% CPU time in kernel mode by process.*

1.6.8 % CPU time in user mode by processor

This metric is similar to the above – only enumerated by CPUs.

1.6.9 % CPU time total by process

This metric measures the amount of CPU time spent in user, nice, and kernel modes by process.

1.6.10 % CPU total active time

Amount of time the CPU spent not idling.

1.6.11 % CPU total active time by processor

Similar to the above metric, but broken down by CPU.

1.6.12 accumulated CPU time in kernel mode by process

1.6.13 accumulated CPU time in user mode by process

This metric is similar to *accumulated CPU time in kernel mode by process* above.

1.6.14 load average

This metric measures the average length of the running process queue in the kernel. These are processes that are waiting for some CPU time. This is the number of of processes that can be executed concurrently provided there are enough CPUs available. This metric is related to *% CPU idle time.*

1.6.15 nice value by process

This metric measures the nice value (priority) of each process on the machine. This nice value can be manually changed with the `nice` command. The nice value may be any integer from -20 (highest priority) to 19 (lowest priority), inclusive.

1.7 Resource `LINUX_FILESYSTEM`

1.7.1 % free by file system

Percentage of disk space free, enumerated by filesystem.

1.7.2 % used by file system

Metric measuring disk usage enumerated by file system.

1.7.3 % of space free

This metric measures the percentage of free space available across the entire filesystem (the sum of all mounted file systems).

1.7.4 % of space used

The complement to the above *% of space free.*

1.7.5 available (in 1 MB blocks) by file system

Similar to *% free by filesystem* – only the measurements are presented in MB instead of percentages.

1.7.6 size (in 1 MB blocks) by file system

This metric simply lists the size of each filesystem. This metric's values won't change (often).

1.7.7 total size of all file systems (in 1 MB blocks)

This metric is simply the summation of *size by file system* above over all file systems mounted. This metrics values should rather change.

1.7.8 total space available (in 1 MB blocks)

Total space available (free space) on all file systems mounted.