NAME

rrd-system – description of the system monitoring graphs

DESCRIPTION

The *system.xml* file in the **rrdmon** package describes a couple of data channels retrieved from the operating system that help to assess the use of system resources.

CPU USAGE

The CPU usage graphs give information about how the CPUs available on the system are currently used.

The bars in three different shades of blue show the portion of the avaiable CPU time spent in the operating system kernel (*system* time, medium blue), in application processes (*user* time, dark blue) and the time spent by *nice* processes (*nice* time, light blue). Nice processes are lower priority processes. The will only consume signification portions of the CPU time if no normal priority process requests being scheduled. They can usually be ignored when assessing whether a system has enought CPU cores allocated, as they do not usually slow down other processes.

The CPU will always spend some time handling interrupts caused by periferal devices (*irq*, violett) or needed to complete them (*softirq*, pink). If this portion becomes excessive, then the periferal device seems to make more processing demands than the system can handle.

Very often, a process needs to wait for the completion of an IO request. This time is counted as IO wait time (*iowait*). If the process blocks without an outstanding IO request, the time spent in this state is counted as idle and does not show up in the graph. A large *iowait* time can indicate that there are too many processes competing for IO operations to complete that the disk system can handle. Sometimes it also means that the application does not use the IO system efficiently, i.e. by requesting the data in a large number of small requests instead of fewer but larger requests.

SYSTEM LOAD

The system load graphs summarize the traditional Unix system *load average* indicating the number of processes that are not blocked on some IO or network operation and are thus expecting to be scheduled for execution. The system provides moving averages of the number processes in the runnable state in the last minute (green curve), the last 5 minutes (blue curve) and the last 15 minutes (black curve).

It is normal that the number of runnable processes can be larger than the number of CPUs available on the system. This may happen when a long pipeline command is issued. If the components of the pipeline are approximately balanced in their CPU requirements, then a constant stream of data flows through the pipeline, keeping all the pipeline stage commands alive and competing for CPU cycles. This leads to many processes in the runable state waiting to get scheduled by the kernel.

It also happens of a process uses a large number of threads to perform some CPU intensive processing. If the processing done by the threads is CPU bound, most of them will be runnable and waiting for them to be schedule, inflating the load average.

If the number of processes is larger than than the number of CPUs for prolonged periods of time, then it is reasonable to conclude that the process load is to high, leading to frequent switches between processes or theads. As this will mostly invalidate CPU caches, the overall performance of the system will be reduced. Such workloads should be reconfigured so that the load average stays closer to the actual number of cores available on the system.

MEMORY USAGE

The memory graph shows how available memory is currently used. The yellow region indicates memory available on the system. The part of memory currently in use is shown in dark red. If a system always has

a large portion of yellow in this graph, then the workload on that system cannot make use of the available memory, the memory is wasted. Over time, the memory will fill up with data buffered during IO operations. It is therefore possible that the graph shows a large portion of the memory being in use although the size of the processes added together is only a small part of that.

The dark blue curve shows the memory in use but not for buffers. This is a better measure for the memory used by processes. A process does not only consist of executable pages, and data pages, it also has buffers containing file data associated with it. All the memory pages that are allocated to some process comprise what is called *active memory*. The portion of active memory is shown as a light blue line.

Some processes request memory they never use. The operating system takes note of the demands of that process, but only assigns memory pages to the process when the process actually first touches the address of that page. This is called *committed* memory and is shown in the green curve of the graph. Processes usually do this to ensure the get the memory they need if the system get's into trouble. Security monitoring software likes to do this to ensure an attacker cannot avoid being monitored just be using up all the memory. MDE (microsoft defender for endpoint) is famous for commiting a lot of memory it will never actually use. This is usually not a problem but it can be considered a ticking timebomb: if the going get's tough, the process may actually want to use the memory they had asked for, causing the system to run out of memory and to start killing random processes in order to prevent a crash.

On top of the yellow bar, a gray bar displays swap space available to the system. Swap space used to be useful when memory was much more expensive than disk space and disk space was not too slow. Nowadays, with large memory spaces available cheaply, large process sizes and disk space many orders of magnitude slower, it has become much less useful. It simply takes way to much time to page in pages previously sent to the swap space for the system to still be usable.

The used portion of swap space is indicated by the light red area. A large swap space usage does not automatically indicate an ongoing memory shortage. It is often caused by a temporary memory shortage in the past that caused the system to page out some pages that weren't in use by processes anyway. Since a considerable portion of a process's memory can be unused for most of the runtime of the process, this may not have any adverse effects on system performance.

A large portion of swap space can be an indication of a memory leak in some process. If a process leaks memory, it will cause the used portion of memory to increase until it is full. At this point, the system is forced to page out some of the memory, and because of the least-recently-used strategy, it will page out the oldest leaked memory first. The swap space essentially becomes a graveyard for leaked memory pages.

NETWORK

The network graphs show data flowing in and out of the system.

CONTROLLER AND DISK

This graphs show usage of disk controllers and disks. It contains separate graphs for read and write operations per second, data volume read and written and wait times for read and write operations.

In normal operation, most IO operations are expected to be caused by processes reading or writing application data. Executable code will usually already be in memory and thus not need IO operations to make it available. Read operations are then used to read data from disk, but if the working set of the data is small enough to fit into memory, no further read operations will be necessary and the system will mostly do write operations e.g. to store new data on disk or to write log files.

If a system does way more read operations then write operations the question must be asked, where does all the data read go? Such a system may be in need of larger memory to allow it to serve requests from memory withoug the need to reread it from disk.

Modern systems try to large IO operations, so large swaths of data will be transferred with only very few operations, which increases overall efficiency. The graphs show read and operations per second in yellow and orange, but MBs read/written per scond in light and dark violett. If the number of IO operations is way larger than the data size, this indicates that the system can only do small IO operations. This can be caused by inefficient programming in the application (there are applications calling the kernel for a write operation for every single byte the output).

The controller graphs show the aggregated IO load from and to all the disks controlled by this controller. On large systems it allows to see whether there is an imbalance in the IO load over different available IO channels.

SEE ALSO

rrdsetup(8), **rrdupdate**(8),

AUTHOR

Andreas Mueller, andreas.mueller@othello.ch