Kernel NFS Server Statistics

This document describes the format and semantics of the statistics which the kernel NFS server makes available to userspace. These statistics are available in several text form pseudo files, each of which is described separately below.

In most cases you don't need to know these formats, as the nfsstat(8) program from the nfs-utils distribution provides a helpful command-line interface for extracting and printing them.

All the files described here are formatted as a sequence of text lines, separated by newline '\n' characters. Lines beginning with a hash '#' character are comments intended for humans and should be ignored by parsing routines. All other lines contain a sequence of fields separated by whitespace.

/proc/fs/nfsd/pool_stats

This file is available in kernels from 2.6.30 onwards, if the /proc/fs/nfsd filesystem is mounted (it almost always should be).

The first line is a comment which describes the fields present in all the other lines. The other lines present the following data as a sequence of unsigned decimal numeric fields. One line is shown for each NFS thread pool.

All counters are 64 bits wide and wrap naturally. There is no way to zero these counters, instead applications should do their own rate conversion.

pool

> The id number of the NFS thread pool to which this line applies. This number does not change.
>
> Thread pool ids are a contiguous set of small integers starting at zero. The maximum value depends on the thread pool mode, but currently cannot be larger than the number of CPUs in the system. Note that in the default case there will be a single thread pool which contains all the nfsd threads and all the CPUs in the system, and thus this file will have a single line with a pool id of "0".

packets-arrived

> Counts how many NFS packets have arrived. More precisely, this is the number of times that the network stack has notified the sunrpc server layer that new data may be available on a transport (e.g. an NFS or UDP socket or an NFS/RDMA endpoint).
>
> Depending on the NFS workload patterns and various network stack effects (such as Large Receive Offload) which can combine packets on the wire, this may be either more or less than the number of NFS calls received (which statistic is available elsewhere). However this is a more accurate and less workload-dependent measure

of how much CPU load is being placed on the sunrpc server layer due to NFS network traffic.

sockets-enqueued

Counts how many times an NFS transport is enqueued to wait for an nfsd thread to service it, i.e. no nfsd thread was considered available.

The circumstance this statistic tracks indicates that there was NFS network-facing work to be done but it couldn't be done immediately, thus introducing a small delay in servicing NFS calls. The ideal rate of change for this counter is zero; significantly non-zero values may indicate a performance limitation.

This can happen either because there are too few nfsd threads in the thread pool for the NFS workload (the workload is thread-limited), or because the NFS workload needs more CPU time than is available in the thread pool (the workload is CPU-limited). In the former case, configuring more nfsd threads will probably improve the performance of the NFS workload. In the latter case, the sunrpc server layer is already choosing not to wake idle nfsd threads because there are too many nfsd threads which want to run but cannot, so configuring more nfsd threads will make no difference whatsoever. The overloads-avoided statistic (see below) can be used to distinguish these cases.

threads-woken

Counts how many times an idle nfsd thread is woken to try to receive some data from an NFS transport.

This statistic tracks the circumstance where incoming network-facing NFS work is being handled quickly, which is a good thing. The ideal rate of change for this counter will be close to but less than the rate of change of the packets-arrived counter.

overloads-avoided

Counts how many times the sunrpc server layer chose not to wake an nfsd thread, despite the presence of idle nfsd threads, because too many nfsd threads had been recently woken but could not get enough CPU time to actually run.

This statistic counts a circumstance where the sunrpc layer heuristically avoids overloading the CPU scheduler with too many runnable nfsd threads. The ideal rate of change for this counter is zero. Significant non-zero values indicate that the workload is CPU limited. Usually this is associated with heavy CPU usage on all the CPUs in the nfsd thread pool.

If a sustained large overloads-avoided rate is detected on a pool, the top(1) utility should be used to check for the following pattern of CPU usage on all the CPUs associated with the given nfsd thread pool.

- %us ~= 0 (as you're *NOT* running applications on your NFS server)

- %wa ~= 0

- %id ~= 0

- %sy + %hi + %si ~= 100

If this pattern is seen, configuring more nfsd threads will *not* improve the performance of the workload. If this patten is not seen, then something more subtle is wrong.

threads-timedout

Counts how many times an nfsd thread triggered an idle timeout, i.e. was not woken to handle any incoming network packets for some time.

This statistic counts a circumstance where there are more nfsd threads configured than can be used by the NFS workload. This is a clue that the number of nfsd threads can be reduced without affecting performance. Unfortunately, it's only a clue and not a strong indication, for a couple of reasons:

- Currently the rate at which the counter is incremented is quite slow; the idle timeout is 60 minutes. Unless the NFS workload remains constant for hours at a time, this counter is unlikely to be providing information that is still useful.

- It is usually a wise policy to provide some slack, i.e. configure a few more nfsds than are currently needed, to allow for future spikes in load.

Note that incoming packets on NFS transports will be dealt with in one of three ways. An nfsd thread can be woken (threads-woken counts this case), or the transport can be enqueued for later attention (sockets-enqueued counts this case), or the packet can be temporarily deferred because the transport is currently being used by an nfsd thread. This last case is not very interesting and is not explicitly counted, but can be inferred from the other counters thus:

```
packets-deferred = packets-arrived - ( sockets-enqueued + threads-woken )
```

More

Descriptions of the other statistics file should go here.

Greg Banks <gnb@sgi.com>
26 Mar 2009