Hardware Latency Detector

Introduction

The tracer hwlat_detector is a special purpose tracer that is used to detect large system latencies induced by the behavior of certain underlying hardware or firmware, independent of Linux itself. The code was developed originally to detect SMIs (System Management Interrupts) on x86 systems, however there is nothing x86 specific about this patchset. It was originally written for use by the "RT" patch since the Real Time kernel is highly latency sensitive.

SMIs are not serviced by the Linux kernel, which means that it does not even know that they are occuring. SMIs are instead set up by BIOS code and are serviced by BIOS code, usually for "critical" events such as management of thermal sensors and fans. Sometimes though, SMIs are used for other tasks and those tasks can spend an inordinate amount of time in the handler (sometimes measured in milliseconds). Obviously this is a problem if you are trying to keep event service latencies down in the microsecond range.

The hardware latency detector works by hogging one of the cpus for configurable amounts of time (with interrupts disabled), polling the CPU Time Stamp Counter for some period, then looking for gaps in the TSC data. Any gap indicates a time when the polling was interrupted and since the interrupts are disabled, the only thing that could do that would be an SMI or other hardware hiccup (or an NMI, but those can be tracked).

Note that the hwlat detector should *NEVER* be used in a production environment. It is intended to be run manually to determine if the hardware platform has a problem with long system firmware service routines.

Usage

Write the ASCII text "hwlat" into the current_tracer file of the tracing system (mounted at /sys/kernel/tracing or /sys/kernel/tracing). It is possible to redefine the threshold in microseconds (us) above which latency spikes will be taken into account.

Example:

```
# echo hwlat > /sys/kernel/tracing/current_tracer
# echo 100 > /sys/kernel/tracing/tracing_thresh
```

The /sys/kernel/tracing/hwlat_detector interface contains the following files:

- width - time period to sample with CPUs held (usecs)
  must be less than the total window size (enforced)
- window - total period of sampling, width being inside (usecs)

By default the width is set to 500,000 and window to 1,000,000, meaning that for every 1,000,000 usecs (1s) the hwlat detector will spin for 500,000 usecs (0.5s). If tracing_thresh contains zero when hwlat tracer is enabled, it will change to a default of 10 usecs. If any latencies that exceed the threshold is observed then the data will be written to the tracing ring buffer.

The minimum sleep time between periods is 1 millisecond. Even if width is less than 1 millisecond apart from window, to allow the system to not be totally starved.

If tracing_thresh was zero when hwlat detector was started, it will be set back to zero if another tracer is loaded. Note, the last value in tracing_thresh that hwlat detector had will be saved and this value will be restored in tracing_thresh if it is still zero when hwlat detector is started again.

The following tracing directory files are used by the hwlat_detector:

in /sys/kernel/tracing:

- tracing_threshold - minimum latency value to be considered (usecs)
- tracing_max_latency - maximum hardware latency actually observed (usecs)
- tracing_cpumask - the CPUs to move the hwlat thread across
- hwlat_detector/width - specified amount of time to spin within window (usecs)
- hwlat_detector/window - amount of time between (width) runs (usecs)
- hwlat_detector/mode - the thread mode

By default, one hwlat detector's kernel thread will migrate across each CPU specified in cpumask at the beginning of a new window, in a round-robin fashion. This behavior can be changed by changing the thread mode, the available options are:

- none: do not force migration
- round-robin: migrate across each CPU specified in cpumask [default]
- per-cpu: create one thread for each cpu in tracing_cpumask