/*
 * pcc-cpufreq.txt - PCC interface documentation
 *
 * Copyright (C) 2009 Red Hat, Matthew Garrett <mjg@redhat.com>
 * Copyright (C) 2009 Hewlett-Packard Development Company, L.P.
 *   Nagananda Chumbalkar <nagananda.chumbalkar@hp.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; version 2 of the License.
 *
 * This program is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE, GOOD TITLE or NON
 * INFRINGEMENT. See the GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program; if not, write to the Free Software Foundation, Inc.,
 * 675 Mass Ave, Cambridge, MA 02139, USA.
 *
 */

Processor Clocking Control Driver

Contents:

1. Introduction:

Processor Clocking Control (PCC) is an interface between the platform firmware and OSPM. It is a mechanism for coordinating processor performance (ie: frequency) between the platform firmware and the OS.

The PCC driver (pcc-cpufreq) allows OSPM to take advantage of the PCC interface.

OS utilizes the PCC interface to inform platform firmware what frequency the OS wants for a logical processor. The platform firmware attempts to achieve the requested frequency. If the request for the target frequency could not be satisfied by platform firmware, then it usually means that power budget

conditions are in place, and "power capping" is taking place.

1.1 PCC interface:

The complete PCC specification is available here:
http://www.acpica.org/download/Processor-Clocking-Control-v1p0.pdf

PCC relies on a shared memory region that provides a channel for communication between the OS and platform firmware. PCC also implements a "doorbell" that is used by the OS to inform the platform firmware that a command has been sent.

The ACPI PCCH() method is used to discover the location of the PCC shared memory region. The shared memory region header contains the "command" and "status" interface. PCCH() also contains details on how to access the platform doorbell.

The following commands are supported by the PCC interface:
* Get Average Frequency
* Set Desired Frequency

The ACPI PCCP() method is implemented for each logical processor and is used to discover the offsets for the input and output buffers in the shared memory region.

When PCC mode is enabled, the platform will not expose processor performance or throttle states (_PSS, _TSS and related ACPI objects) to OSPM. Therefore, the native P-state driver (such as acpi-cpufreq for Intel, powernow-k8 for AMD) will not load.

However, OSPM remains in control of policy. The governor (eg: "ondemand") computes the required performance for each processor based on server workload. The PCC driver fills in the command interface, and the input buffer and communicates the request to the platform firmware. The platform firmware is responsible for delivering the requested performance.

Each PCC command is "global" in scope and can affect all the logical CPUs in the system. Therefore, PCC is capable of performing "group" updates. With PCC the OS is capable of getting/setting the frequency of all the logical CPUs in the system with a single call to the BIOS.

1.1.1 Get Average Frequency:

This command is used by the OSPM to query the running frequency of the processor since the last time this command was completed. The output buffer indicates the average unhalted frequency of the logical processor expressed as a percentage of the nominal (ie: maximum) CPU frequency. The output buffer also signifies if the CPU frequency is limited by a power budget condition.

1.1.2 Set Desired Frequency:

This command is used by the OSPM to communicate to the platform firmware the desired frequency for a logical processor. The output buffer is currently ignored by OSPM. The next invocation of "Get Average Frequency" will inform OSPM if the desired frequency was achieved or not.

1.2 Platforms affected:

The PCC driver will load on any system where the platform firmware:
* supports the PCC interface, and the associated PCCH() and PCCP() methods
* assumes responsibility for managing the hardware clocking controls in order to deliver the requested processor performance

Currently, certain HP ProLiant platforms implement the PCC interface. On those platforms PCC is the "default" choice.

However, it is possible to disable this interface via a BIOS setting. In such an instance, as is also the case on platforms where the PCC interface is not implemented, the PCC driver will fail to load silently.

2. Driver and /sys details:

When the driver loads, it merely prints the lowest and the highest CPU frequencies supported by the platform firmware.

The PCC driver loads with a message such as:
pcc-cpufreq: (v1.00.00) driver loaded with frequency limits: 1600 MHz, 2933 MHz

This means that the OPSM can request the CPU to run at any frequency in between the limits (1600 MHz, and 2933 MHz) specified in the message.

Internally, there is no need for the driver to convert the "target" frequency to a corresponding P-state.

The VERSION number for the driver will be of the format v.xy.ab.
eg: 1.00.02

```
   ----- --
    |    |
    |    -- this will increase with bug fixes/enhancements to the driver
    |-- this is the version of the PCC specification the driver adheres to
```

The following is a brief discussion on some of the fields exported via the /sys filesystem and how their values are affected by the PCC driver:

2.1 scaling_available_frequencies:
---
scaling_available_frequencies is not created in /sys. No intermediate frequencies need to be listed because the BIOS will try to achieve any frequency, within limits, requested by the governor. A frequency does not have to be strictly associated with a P-state.

2.2 cpuinfo_transition_latency:

The cpuinfo_transition_latency field is 0. The PCC specification does not include a field to expose this value currently.

2.3 cpuinfo_cur_freq:

A) Often cpuinfo_cur_freq will show a value different than what is declared in the scaling_available_frequencies or scaling_cur_freq, or scaling_max_freq.

This is due to "turbo boost" available on recent Intel processors. If certain conditions are met the BIOS can achieve a slightly higher speed than requested by OSPM. An example:

```
scaling_cur_freq        : 2933000
cpuinfo_cur_freq        : 3196000
```

B) There is a round-off error associated with the cpuinfo_cur_freq value. Since the driver obtains the current frequency as a "percentage" (%) of the nominal frequency from the BIOS, sometimes, the values displayed by scaling_cur_freq and cpuinfo_cur_freq may not match. An example:

```
scaling_cur_freq        : 1600000
cpuinfo_cur_freq        : 1583000
```

In this example, the nominal frequency is 2933 MHz. The driver obtains the current frequency, cpuinfo_cur_freq, as 54% of the nominal frequency:

```
        54% of 2933 MHz = 1583 MHz
```

Nominal frequency is the maximum frequency of the processor, and it usually corresponds to the frequency of the P0 P-state.

2.4 related_cpus:

The related_cpus field is identical to affected_cpus.

```
affected_cpus   : 4
related_cpus    : 4
```

Currently, the PCC driver does not evaluate _PSD. The platforms that support PCC do not implement SW_ALL. So OSPM doesn't need to perform any coordination to ensure that the same frequency is requested of all dependent CPUs.

3. Caveats:

The "cpufreq_stats" module in its present form cannot be loaded and expected to work with the PCC driver. Since the "cpufreq_stats" module provides information wrt each P-state, it is not applicable to the PCC driver.