Reprinted from the

Proceedings of the Linux Symposium

July 23rd–26th, 2008
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*

C. Craig Ross, *Linux Symposium*

Review Committee

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*

Dirk Hohndel, *Intel*
Gerrit Huizenga, *IBM*
Dave Jones, *Red Hat, Inc.*
Matthew Wilson, *rPath*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*
Gurhan Ozen, *Red Hat, Inc.*
Eugene Teo, *Red Hat, Inc.*
Kyle McMartin, *Red Hat, Inc.*
Jake Edge, *LWN.net*
Robyn Bergeron
Dave Boutcher, *IBM*
Mats Wichmann, *Intel*

Authors retain copyright to all submitted papers, but have granted unlimited redistribution rights to all as a condition of submission.

Thermal Management in User Space

Sujith Thomas
Intel Ultra-Mobile Group
sujith.thomas@intel.com

Zhang Rui
Intel Open Source Technology Center
zhang.rui@intel.com

Abstract

With the introduction of small factor devices like Ultra Mobile PC, thermal management has gained a higher level of importance. Existing thermal management solutions in the Linux kernel lack a standard interface to user space. This mandates that all the thermal management policy control needs to reside in the kernel. As more and more complex algorithms are being developed for thermal management, it makes sense to allow moving the *policy control decisions* part into user space and allow the kernel to just facilitate these decisions.

In this paper, we will introduce a generic solution for Linux thermal management, which usually contains a user application for policy control; a generic thermal sysfs driver, which provides a set of platform-independent interfaces; native sensor drivers; and device drivers for thermal monitoring and device throttling. We will also take a look at the software stack of Intel's Menlow platform, where this solution is already enabled.

1 Thermal Modeling

Even though there are many thermal modeling concepts out there, the crux still remains the same.

- There are sensors associated with devices. The devices have various throttle levels and by putting the device into a lower performance state, the temperature of the device as well as the overall platform will decrease.
- There may be provisions for programming sensor trips to send notifications to the monitoring application.
- There may be a fan in the platform and it may have multi-speed control.

Provided these hardware features are available, how can the software manage thermals for a platform? This can be implemented either in kernel space or in user space. The kernel-space implementation by using the framework is enough as long as there are only a few thermal contributors, and mainly the CPU.

But with the Ultra Mobile PCs (UMPC) and Mobile Internet Devices (MID), the CPU is no longer the major thermal contributor. There may be cases where, over a period of time, multiple devices' contributions cause the platform temperature to rise significantly. So, the real challenge here is to choose the right device(s) and to pick up the right performance levels.

2 The Concept of 'Thermal management in User Space'

Thermal management in user space would imply that all the policy decisions will be taken from user space and the kernel's job would be only to facilitate those decisions. This model gives us the these advantages:

- The algorithms can scale well from simple scripts to complex algorithms involving neural networks.
- The kernel is freed from consuming CPU cycles for non-critical tasks. Response to a non-critical thermal scenario, which is the most common case, is not immediately required—that is, we don't have to account for decisions which are in milliseconds or microseconds. This is because raising the platform temperature about one degree Celsius takes around 20s–30s on most platforms. So moving thermal management from kernel space, where we have critical things to do, is the right thing to do.
- This also guarantees that the same application will work on different platforms even though the thermal modeling is different at the hardware level.

But for the thermal management to shift to user space, applications still need to get support from the kernel. That's the role the generic thermal management framework plays.

3 ACPI vs. Generic Thermal Management

The ACPI 2.0 thermal model was a good start for thermal management. But for new platforms with small form factors like the Mobile Internet Devices (MID), this model is no longer sufficient. Some of the reasons are:

- ACPI proposes active trip points, but there may not be any fans on the handhelds.
- ACPI assumes that the CPU is the major thermal contributor and doesn't discuss other thermal contributors.
- ACPI doesn't support sensors with programmable AUX trip points.

Even with these limitations, many platforms still use ACPI because of its other benefits. As a matter of fact, the generic thermal management is not a thermal model itself; instead it complements existing models like ACPI.

The generic thermal management solution was designed to support thermal models (like ACPI 2.0) and to go beyond, to complement such models with proprietary platform-based sensors and devices. Intel's Menlow platform is a good example of using ACPI as the backbone for thermal management. Along with that, it uses sensors with programmable AUX trip points and it can even throttle the memory controller. A case study in the latter part of this paper illustrates this solution.

4 Generic Thermal Management Architecture

The generic thermal management has these key components:

- Thermal zone drivers for thermal monitoring and control.
- Cooling device drivers for device throttling.
- An event framework to propagate the platform events to user-space applications.
- A generic thermal sysfs driver which provides a set of platform-independent interfaces.

Figure 1 shows the software stack of the generic thermal solution.

4.1 Thermal Zone Drivers

A thermal zone, by definition, not only gives the temperature reading of a thermal sensor, but also gives the list of cooling devices associated with a sensor. The driver or application may in turn control these devices to bring down the temperature of this thermal zone. The thermal zone driver abstracts all the platform-specific sensor information and exposes the platform thermal data to the thermal sysfs driver. This may include data like temperature and trip points. In addition, it also binds cooling devices to the associated thermal zones. This driver is also responsible for notifying user space about thermal events happening in the platform.

4.2 Cooling Device Drivers

The cooling device drivers are associated with thermal contributors in the platform. The cooling device drivers can register with the generic thermal sysfs driver, thus becoming the part of platform thermal management. By registering, these drivers provide a set of thermal ops that they can support, like the number of cooling states they support and the current cooling state they are in. The generic thermal sysfs driver will redirect all the control requests to the appropriate cooling device driver when the user application sets a new cooling state. It is up to the cooling device driver to implement the actual thermal control.

4.3 Eventing Framework

Events will be passed from kernel to user space using the netlink facility. The applications may use `libnetlink` to receive these events and to do further processing.

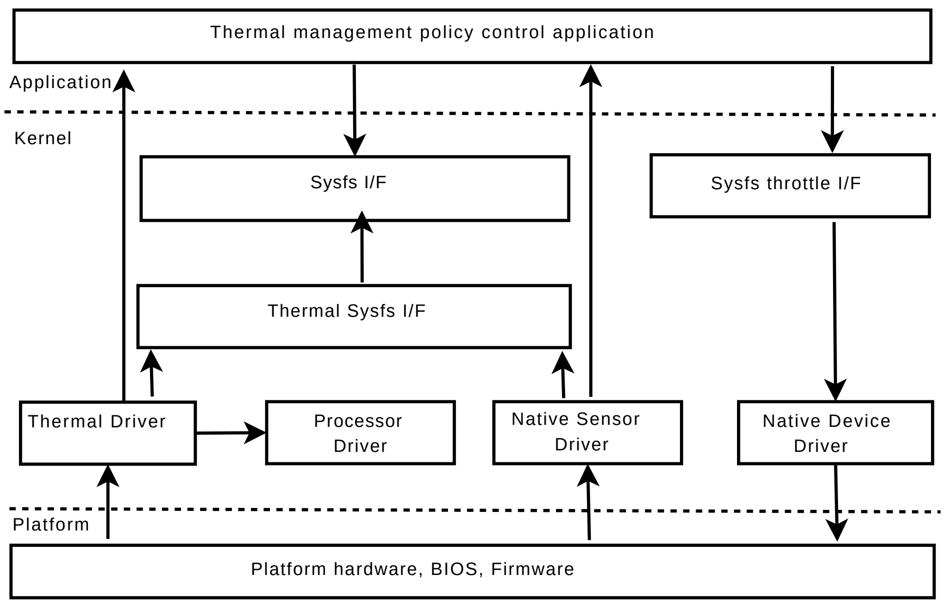

Figure 1: Generic thermal management architecture

4.4 Generic Thermal sysfs Driver

The generic thermal sysfs driver is a platform-independent driver which interacts with the platform-specific thermal zone drivers and cooling device drivers. This driver, in turn, builds a platform-independent sysfs interface (or I/F) for user space application. It mainly works on device management and sysfs I/F management for registered sensors and cooling devices.

4.4.1 Device Management

The thermal sysfs driver exports the following interfaces

- `thermal_zone_device_register()`
- `thermal_zone_device_unregister()`
- `thermal_cooling_device_register()`
- `thermal_cooling_device_unregister()`
- `thermal_zone_bind_cooling_device()`
- `thermal_zone_unbind_cooling_device()`

for the thermal zone drivers and cooling device drivers to register with the generic thermal solution and to be a part of it. The thermal sysfs driver creates a sysfs class during initialization and creates a device node for each registered thermal zone device and thermal cooling device. These nodes will be used later on to add the thermal sysfs attributes.

The `bind`/`unbind` interfaces are used by the thermal zones to keep a mapping of the cooling devices associated with a particular thermal zone. Thermal zone drivers usually call this function during registration, or when any new cooling device is registered.

4.4.2 Sysfs Property

The generic thermal sysfs driver interacts with all platform-specific thermal sensor drivers to populate the standard thermal sysfs entries. Symbolic links are created by the generic thermal driver to indicate the binding between a thermal zone and all cooling devices associated with that particular zone. Table 1 gives all the attributes supported by the generic thermal sysfs driver.

The generic thermal management uses a concept of *cooling states*. The intent of a cooling state is to define thermal modes for supporting devices. The higher the cooling state, the lower the device/platform temperature would be. This can be used for both passive and active cooling devices. It's up to the cooling device driver to

Sysfs	Location	Description	RW
type	/sys/class/thermal/thermal_zone[0-*]	The type of the thermal zone	RO
mode	/sys/class/thermal/thermal_zone[0-*]	One of the predefined values in [kernel, user]	RW
temp	/sys/class/thermal/thermal_zone[0-*]	Current temperature	RO
trip_point_[0-*]_temp	/sys/class/thermal/thermal_zone[0-*]	Trip point temperature value	RO
trip_point_[0-*]_type	/sys/class/thermal/thermal_zone[0-*]	Trip point type	RO
type	/sys/class/thermal/cooling_device[0-*]	The type of the cooling device	RO
max_state	/sys/class/thermal/cooling_device[0-*]	The maximum cooling state supported	RO
cur_state	/sys/class/thermal/cooling_device[0-*]	The current cooling state	RW
cdev[0-*]	/sys/class/thermal/thermal_zone[0-*]	Symbolic links to a cooling device node	NA
cdev[0-*]_trip_point	/sys/class/thermal/thermal_zone[0-*]	The trip point that this cooling device is associated with	RO

Table 1: Thermal sysfs file structure

implement the cooling states. In most of the cases, it may map directly to the power modes of the device. But in some other cases, it may not. The CPU is the example of when power modes are controlled by P states, but thermal is controlled by a combination of P and T states.

Besides the generic thermal sysfs files, the generic thermal sysfs driver also supports the `hwmon` thermal sysfs extensions. The thermal sysfs driver registers an `hwmon` device for each type of registered thermal zones. With the `hwmon` sysfs extensions, an attempt has been made to support applications which use the `hwmon` style of interfaces. Currently, using this interface, the temperature and critical trip point of the platform are exposed.

Table 2 shows the `hwmon` thermal sysfs extensions.

5 User Space Policy Control

The generic thermal management framework enables thermal management applications to collect all the relevant thermal data. The data will be pre-processed and then passed on to the intelligent algorithm where the throttling decisions are taken. The algorithm may consider user preferences before executing any decisions, again through the generic thermal management framework.

Here are the operations which applications can perform using the generic thermal management framework.

- Enumerate the list of thermal sensors in the platform.
- Enumerate the list of thermal contributors (CPU, Memory, etc.) in the platform.
- Enumerate the list of active cooling devices (fans) in the platform.
- Enumerate the thermal zones (to get device sensor associations) in the platform.
- Get sensor temperatures and trip points of various sensors.
- Set the threshold trip points if the underlying platform supports this feature.
- Get notifications on thermal events happening in the platform.
- Get exclusive control of any thermal zone in the platform.

6 Thermal Management on Intel's Menlow Platform

Menlow is Intel's handheld platform for the 2008 time frame. It is a small form-factor device (screen size of about 5 inches), which makes its thermal management a challenge. The goal of the solution was that at any time, the skin temperature (top and bottom) should be below 45°C. The other challenge was that the CPU was not the major thermal contributor. There are other devices, like the memory controller and communication devices, which contributed equally to the platform's skin temperature. There was clearly a need for a complex algorithm to perform the thermal management by throttling the devices at the same time, while not compromising the performance.

6.1 Why ACPI Was Not Enough…

Menlow's thermal management solution leverages many of the ACPI standards available on the platform. But

Sysfs	Location	Description	RW
Name	/sys/class/hwmon/hwmon[0-*]	Same as the thermal zone 'type'	RO
Temp[1-*]_temp	/sys/class/hwmon/hwmon[0-*]	Current temperature value	RO
Temp[1-*]_crit	/sys/class/hwmon/hwmon[0-*]	Critical temperature value	RO

Table 2: `Hwmon` support file structure

relying only on the ACPI standards was not enough because sensors available in the platform were capable of doing more things than in ACPI 2.0. Hence the concept of *generic thermal management* was proposed.

The Menlow platform has many thermal sensors attached to the platform's embedded controller. The embedded controller firmware was in charge of reading the temperature from the sensors. These sensors had the additional capability of programming the AUX trips, wherein the application can program the upper and lower thresholds, based on the current temperature. Whenever the temperature exceeds any of these thresholds, the application will get an event and can make a decision based on the user policy. ACPI 2.0 didn't have support for AUX trip point programming. Generic thermal management was used to complement the ACPI standards.

6.2 How the Generic Thermal Management Works on Menlow Platform

Menlow's thermal management is the first use of the generic thermal solution. Thermal management on Menlow is made up of these components.

- An intelligent user-space application which can make throttling decisions based on thermal events it receives.
- ACPI thermal management, which has its thermal zone driver (ACPI thermal driver) and cooling device drivers (processor, fan, and video driver) registered with the thermal sysfs driver.
- `intel_menlow` platform driver, which provides required, extra thermal management, such as memory controller throttling and AUX trip point programming.
- ACPI BIOS which has objects for controlling the processor's P and T states.
- Embedded controller firmware which reads the sensor temperature and programs the temperature thresholds.

Figure 2 shows the thermal sysfs architecture on the Menlow platform.

6.3 Thermal Zone Driver on Menlow

The ACPI thermal driver plays a key role on Menlow, which is registered with the thermal sysfs driver to export the temperature and trip point information to user space. The ACPI thermal driver does the thermal management to some extent—that is, it controls the processor P and T states whenever the temperature crosses the configured `_PSV` temperature. The generic thermal management provides a way for a user-space application to override kernel algorithm using the sysfs-exported file named `mode`. If ACPI thermal zones' modes are set to "user," ACPI thermal zones will no longer follow thermal policy control. Instead, they will only export temperature change events to user space through netlink. Whenever the application exits, it can give back the thermal management control by writing "kernel" into the file `mode`. This was needed to guarantee the mutual exclusivity of the thermal management between the kernel and the user-space application.

6.4 Menlow's Native Driver

`Intel_Menlow` is a platform-specific driver which handles:

- AUX trip point programming for platform thermal sensors.
- Throttling of memory controller.

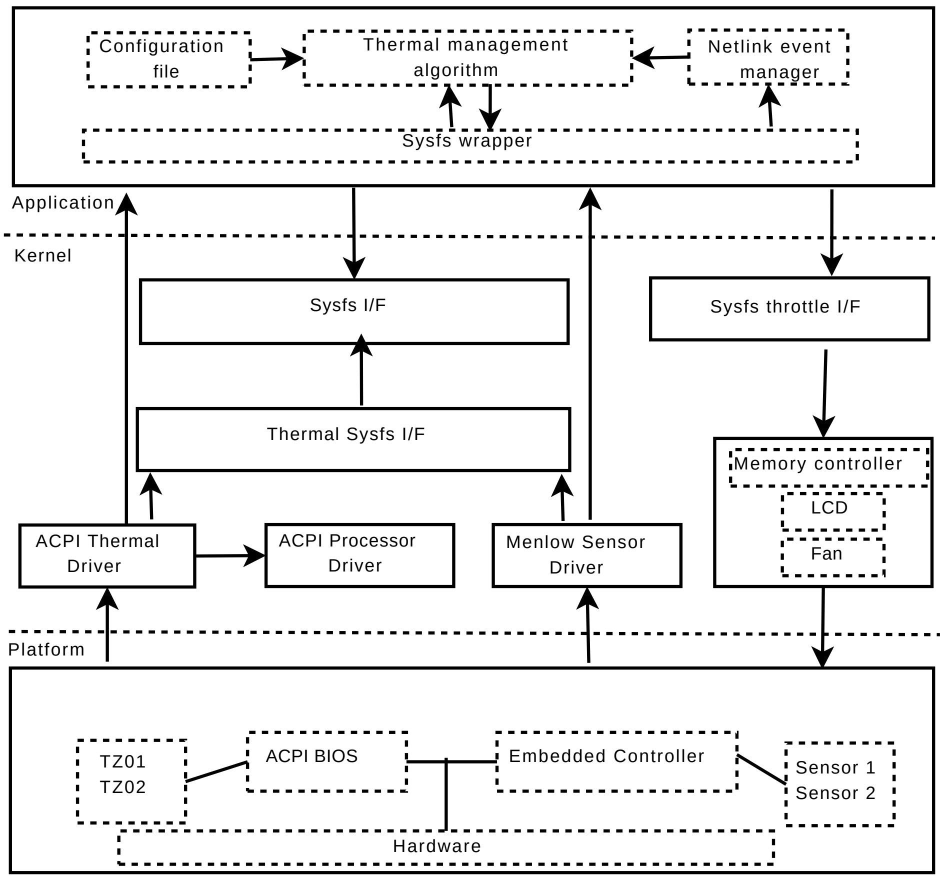

Figure 2: Menlow using the generic thermal management

6.5 Cooling device driver on Menlow

The following cooling devices are registered with the thermal sysfs driver on Menlow:

- Memory controller, which controls the temperature by throttling the memory bandwidth.
- ACPI processor cooling state is a combination of the processor P-state and T-state. The ACPI CPU frequency driver prefers to reduce the frequency first, and then to throttle.
- The ACPI fan driver supports only two cooling states: state 0 means the fan is off, state 1 means the fan is on.
- ACPI video throttles the LCD device by reducing the backlight brightness levels.

7 Conclusion

For handheld devices it is viable to move the thermal management to user space applications. By doing so the application are given the freedom to implement the algorithm that would be the best to handle thermals for a particular class of devices. The job of the kernel would be limited to delivering events and exposing device specific throttle controls. This approach can be used on platforms with ACPI, without ACPI, or to compliment thermal models like ACPI.

8 Acknowledgment

This paper is based on "Cool Hand Linux—Handheld Thermal Extensions" co-written by Len Brown and Harinarayan Seshadri. We would also like to acknowledge Nallaselan Singaravelan, Vinod Koul, and Sailaja Bandarupali of the Ultra Mobility Group, Intel Corporation, for their valuable contributions.

9 References

• "Cool Hand Linux—Handheld Thermal Extensions" by Len Brown and Harinarayan Seshadri, *Proceedings of the Linux Symposium*, Ottawa, Canada, 2007.

- ACPI Hewlett-Packard, Intel, Microsoft, Phoenix, Toshiba. *Advanced Configuration and Power Interface 3.0b*, October, 2006. `http://www.acpi.info`