Intelligent Power Allocation for Consumer & Embedded Thermal Control

Ian Rickards
ARM Ltd, Cambridge UK

ELC San Diego
5-April-2016
Existing Linux Thermal Framework

- Thermal trip mechanism using cooling devices
  - Originally designed to turn on fans.
  - ‘step_wise’ governor adjusts MHz based on temp change
- **Reactive** on temperature overshoot
- **Fixed power partitioning** between different parts of SoC: GPU, CPU, DSP, video
  - does not adapt to current workload

**Solutions currently used in mobile**

- Linux thermal framework
- Custom firmware
- Hotplug
IPA advancements

- **Proactive vs. Reactive Thermal Management**
  - Continuously adapting response based on power consumption and thermal headroom
  - Closed-loop control uses **PID algorithm for accurate temperature control**

- **Dynamic Partitioning vs. Fixed**
  - Optimally allocates power to CPU or GPU depending on current workload

- **Merged in Linux-4.2**
  - Will benefit all operating systems based on Linux, no patches required
ARM Intelligent Power Allocation

Power to Heat

Temperature: $T_{\text{skin}}$

SoC

Device

IPA (PID control algorithm)

Performance Requests

- big
- LITTLE
- GPU

Allocated Performance

- big
- LITTLE
- GPU

Dynamic Allocation by:
- Performance required
- Thermal headroom

Real-time CPU & GPU Performance requests

Elements:
- Temperature control
- Power estimation using model
- Performance allocation
CMOS SoC Power Fundamentals

**Static power**
- Area of silicon (mm²)
- Threshold voltage ($V_t$)
  - “Low $V_t$” implementation faster (but more leaky)
  - “High $V_t$” implementation slower
- Temperature

**Dynamic power**
- Pipeline depth/complexity
- Toggling nodes x capacitance x voltage$^2$
Requirements to use IPA

- At least one on-chip temperature sensor

- Ability to passively control power of key System-on-Chip (SoC) IP blocks
  - Blocks are ‘big’, ‘LITTLE’, ‘GPU’ – their power is controlled so they are ‘cooling devices’
  - Set CPU power cap using max frequency/voltage via cpufreq
  - [optional: set other device frequency/voltage via devfreq, e.g. GPU]

- Power models of SoC IP blocks: Frequency/voltage & ‘Utilization’
  - Dynamic power model
  - Static power model [optional]

- Translate power <-> performance cap
IPA Algorithm - Overview

**Governor performs two tasks:**

1. **Keeps system within thermal envelope**
   - Controls total power budget
   - Exploits thermal headroom

2. **Dynamic power allocation per device**
   - Performance demand & power models
   - Power divided based on what each device requested. Anything left over is distributed among the devices, up to their maximum.
IPA benefits illustration

**PID accurate temp control**

- Optimal ramp up for max performance
- Precise temperature control

**Dynamic power allocation**

- GPU intensive
- CPU intensive

**Power re-allocation**

GPU to CPU as workload changes

- Little
- Big
- GPU

**Temp**

Time

IPA

Existing Linux

Step-wise governor

Power allocated
### Setup

#### Porting parameters

<table>
<thead>
<tr>
<th>struct element</th>
<th>DT name</th>
<th>sysfs (writeable)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>switch_on temperature</td>
<td>trip-point@0</td>
<td>trip_point_0_temp</td>
<td>Temperature above which IPA starts operating (first passive trip point – trip 0)</td>
</tr>
<tr>
<td>desired_temperature</td>
<td>trip-point@1</td>
<td>trip_point_1_temp</td>
<td>Target temperature (last passive trip point – trip 1)</td>
</tr>
<tr>
<td>weight</td>
<td>contribution</td>
<td>cdevX_weight</td>
<td>Weight for the cooling device</td>
</tr>
<tr>
<td>sustainable_power</td>
<td>sustainable-power</td>
<td>sustainable_power</td>
<td>Max sustainable power</td>
</tr>
<tr>
<td>k_po [not h/w property]</td>
<td>k_po</td>
<td></td>
<td>Proportional term constant during temperature overshoot periods</td>
</tr>
<tr>
<td>k_pu [not h/w property]</td>
<td>k_pu</td>
<td></td>
<td>Proportional term constant during temperature undershoot periods</td>
</tr>
<tr>
<td>k_i [not h/w property]</td>
<td>k_i</td>
<td></td>
<td>PID loop's integral term constant (compensates for long-term drift)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When the temperature error is below 'integral_cutoff', errors are accumulated in the integral term</td>
</tr>
<tr>
<td>k_d [not h/w property]</td>
<td>k_d</td>
<td></td>
<td>PID loop's derivative term constant (typically 0)</td>
</tr>
<tr>
<td>integral_cutoff [not h/w property]</td>
<td>integral_cutoff</td>
<td></td>
<td>Typically 0 so cutoff not used</td>
</tr>
</tbody>
</table>

**sysfs location:** 

```
/sys/class/thermal/thermal_zoneX/
```
DT registering thermal_zone and trip points

Currently only trip-points, sustainable-power and weights can be specified in DT as these are direct h/w properties

When using DT, boot happens with defaults and userspace can change it by writing to sysfs files.
Thermal zone hierarchy

- Thermal zones structured as tree (supports multiple temp sensors)

- Power restricted by cooling devices

- Allocated power is \textit{minimum} of that requested by all thermal zones

$\Rightarrow$ Rapid limiting from CPU cluster temp sensor

$\Rightarrow$ Additional limiting from motherboard sensor
IPA in action on 4x4 big.LITTLE

- GeekBench alternates between single and multithreaded phases
  - On big cores frequency is adjusted automatically based on parallel load
    - When 4 cores are active frequency is lower
  - On LITTLE cores frequency follows load
    - When 4 cores are active frequency is lower
Changing Capacities

capacity

LITTLE  big

Task

LITTLE  big

Thermal Capping
Changing Capacities – Energy Aware Scheduling

In EAS r5.2 git
http://linux-arm.org/git?p=linux-power.git
LKML posting planning for EAS RFCv6
Static Power - Hotplug

- Turn core OFF
- Start with ‘big’
- Progressively hotplug more cores
- Issues
  - Latency of thermal response
  - Ping-pong of core due to large discrete steps in power reduction
Summary

- ‘Intelligent Power Allocation’ now available for the Linux kernel

- IPA is designed to **maximise performance in the thermal envelope**:  
  - **Proactively** adjusts available power budget, based on device temperature  
  - Allocates power **dynamically** between CPU/GPU, based on workload

- Results show:  
  - **More accurate thermal control**  
  - **Better performance** than existing governors in Linux thermal framework

- Future: Core idling / Unified energy model
Demo - Technical info

- Odroid-XU3 Exynos 5422 (4xCortex-A15 + 4xCortex-A7)

- Automation: ARM ‘Workload Automation’
- ipython analysis and tuning flow using TRAPpy
  - See Patrick Bellasi presentation “LISA & friends” talk @ 4.20pm
  - ARM “Ice Cave” demo => GPU intensive
  - Antutu – varying CPU/GPU load

- IPA released in Linux 4.2, Linaro LSK 3.10 & 3.18
  - Tooling on https://github.com/arm-software
Thank you