Power Capping Linux
Agenda

• Context
• System Power Management Issues
• Power Capping Overview
• Power capping participants
• Recommendations
• Introduction of Linux Power Capping Framework
Power Hungry World

- Worldwide, the digital warehouses use about 30 billion watts of electricity
- Equivalent to the output of 30 nuclear power plants
- On average, using only 6 percent to 12 percent of the electricity powering their servers to perform computations.
- The rest was essentially used to keep servers idling and ready in case of a surge in activity that could slow or crash their operations.

Unpredictability of battery life

"Many report unpredictable spikes in battery use, and batteries becoming unnervingly hot, even when used outdoors in the shade."

Read more: http://www.dailymail.co.uk/sciencetech/article-2055562/Apple-iPhone-4S-battery-dies-12-hours--users-forced-ways-patch-mend.html#ixzz2dJ5L1Uqz
Constrained by Power and Thermal

- Thinner and more power efficient mobile devices
  - Limited power delivery capacity
  - Limited cooling capacity
  - Battery life predictability
  - Aggressive power budgeting
- Unexpected peaks in system utilization
- Handling catastrophic thermal/power exceptions are never graceful
Power Capping Linux Overview

- Monitoring power usage
- Limit power consumed by devices at runtime
- Redistribution of power to meet power budget at system level
- Maximize performance
- Maximize efficiency
- Avoid critical conditions
Power Capping Usage Model

- Management Interface, Preferences
  - Power Instrument Readings
  - Power Supply I/F
    - Power Capping Framework
      - Power Capping Drivers
  - Power Capping Sysfs
    - Thermal Zone/Temp Sensor
      - External Temp Readings
Power Capping Participants

- CPUs
- GPUs
- DRAM
- Others
  - Multimedia sub system
  - Wireless sub system
  - IO
Power Capping Ideas

• “Drink” less individually
  • Device performance states
  • Throttling states

• Let the lions handle it (reduce head counts)
  • CPU core offline
  • Reduce number of active lanes on an IO bus

• Beyond nature
  • Take turns (Idle injection)
  • Avoid fighting / catastrophic shutdown
Power Capping Techniques

- **CPU**
  - Dynamic performance states (aka P-state) adjustment
  - Dynamic throttling states (aka T-state) adjustment
  - Processor offline
  - Idle injection
  - ACPI power meter
  - ACPI processor aggregator

- **CPU/ DRAM / GPU**
  - Running Average Power Limit (RAPL)
Power Capping Measurements

• Test Setup
  • Intel Ivy Bridge Dual Core
  • Linux 3.11.rc3
  • Power meter: Yokogawa WT210
  • Test load: openssl speed sha256
P-States

- P-state: a voltage/frequency pair
- P1: Guaranteed performance
- Pn-P1 range in OS control/request
- P0: Max possible performance under HW control

\[(\text{Power} = C \cdot V^2 \cdot f)\]
Turbo States
Controlling P-states

• Intel P State Driver
  • sysfs: max_perf_pct, min_perf_pct, no_turbo

• CPUFREQ
  • Sysfs: scaling_max_freq, scaling_min_freq, scaling_setspeed

• Thermal Cooling device
  • sysfs: /sys/class/thermal/cooling_device#/type = “Processor”

• RAPL (Running Average Power Limit)
  • sysfs: via power capping framework
P States Performance (Using Intel P State Driver)

- Power Capping Percentage of Max P-states
- Performance vs. Power (watts)
RAPL

• Power monitoring capability
  • Very accurate, model based, per domain built-in power meter
  • Throttle duration

• Power Limiting

• Performance feedback mechanism
  • Notification when OS requested P-state is denied

• Implemented close to processor

• Interface via MSR, PCIe config space
RAPL Domains and Interfaces
RAPL Performance

![Graphs showing RAPL power limit percentage of TDP and Power (watts)]
RAPL DRAM Power Limit

- Throttles DRAM Bandwidth
- Significant DRAM throttling tend to be power inefficient
  - First throttle CPUs
T-States

- Allow Software Controlled Clock modulation
- Controls stop clock duty cycle
  - Time period for clock signal to drive processor
- Controlled via thermal cooling device interface for processor

Example: 25%
T-States Performance

- **Power Capping Percentage of Max T-states**
  - Graph showing power capping percentage against power capping percentage of Max T-states.

- **Performance vs. Power (Watts)**
  - Graph showing performance against power (watts) with different states indicated:
    - Red circles: RAPL
    - Green circles: P-State
    - Blue circles: T-state
Idle Injection

- Idle states (C-states) get deeper in modern processors
- Synchronized idle time across all cores achieves the maximum power benefit
Idle Injection

- Implemented by Intel Power Clamp Driver
  - sysfs: /sys/class/thermal/cooling_device# /type = “intel_powerclamp”
- Monitors and enforces idle time for each online CPU
- User selectable idle ratio from 0 to 50%

```
CPU 0
Kidle_inject/0 inactive Force idle inactive

CPU 1
Kidle_inject/1 inactive Force idle inactive
```
Idle Injection Performance

Power Capping Percentage of Max Idle Injection Ratio (50%)

Power (Watts)
CPU Offline

• Migrate activity on current CPU to new CPU
  • Processes, interrupts, timers

• Logical online/offline CPUs using sysfs interface
  • sysfs: /sys/devices/system/cpu/cpu#/online

• Conditional physical offline/online
  • depends on BIOS and kernel build flags

• Limited CPU 0 offline
CPU Offline Performance
ACPI Power Meter

- Expose power meter support defined in ACPI 4.0
- Depends on BIOS support
- Interface to read power over a configurable interval
- Trip point configuration for notification
- Configuration for power capping parameters
  - power#_cap_min/power#_cap_max
ACPI Processor Aggregator (ACPI_PAD)

- Triggered by ACPI notification only
- Used to resolve short term thermal emergencies
- Not a CPU Offline/online but has similar affect
- Doesn’t affect cupset
- Puts affected CPUs in deep C states
ACPI_PAD Vs. CPU Offline

Test platform: Intel Romley 2 socket system
Recommendations

- In applicable range
- RAPL
- P State
- Idle Injection
- CPU Offline
- T States
Linux Power Capping Framework

- Interface
  - Set power limits
  - read current power for a system or a sub-system
- Client driver API
- Avoid code duplication
- RFC: [http://lwn.net/Articles/562015/](http://lwn.net/Articles/562015/)
Linux Power Capping Framework Class

Power cap class driver

Registration

Callbacks

Power cap client drivers

sysfs I/F
Power Capping Class Features

• Allow multiple power zones
  • Power zone: Independent unit which has capability to measure and enforce power
• Allow parent child relationship among zones
• Exports Sys-FS Interface (monitoring and control)
  • Get Current energy consumption per power zone
  • Set Power limit per zone
• Power capping driver interface (kernel API)
Power Capping Sys-FS hierarchy example

/sys/class/powercap

intel-rapl

  enabled

powerclamp

  control_type n

  enabled

  intel-rapl:0

    enabled

    name

    energy_uj

    max_energy_range_uj

  intel-rapl:1

    constraint_X_name

    constraint_X_power_limit_uw

  intel-rapl:n

    constraint_X_time_window_us

intel-rapl:0:1
Power Capping sysfs

- Arranged as a tree, with root as a control type
- Control type: Method to implement power capping. E.g. intel-rapl
- A control type contains multiple power zones
- Power zone node names are qualified with the control type, E.g. intel-rapl:0, intel-rapl:1
- Each power zone can have children as power zones
- Parent child relationship should be based on the relationship of power. E.g. When child power limit is applied, parent power is also affected
Takeaways*

• Prefer direct hardware interface
  • Prefer RAPL for power capping from P0 to Pn over indirect control via P-state governor
• Consider Idle injection instead of CPU Offline for extended power capping range, maybe platform dependent.
• Pay attention to hardware trend, e.g. deeper idle
• Reconsider the effectiveness of legacy features
  • e.g. T-states

Suggestions on Improving Power Capping Framework?

(*Disclaimer: only based on the test result shown in the slides, not an official recommendation by Intel)
Q&A

Contact: jacob.jun.pan@linux.intel.com
Thank You