QoS Handling with DVFS (CPUfreq & Devfreq)

MyungJoo Ham
SW Center, Samsung Electronics

Performance Issues of DVFS

- Performance Sucks w/ DVFS!
- Battery-life Still Matters
  - More Devices (components) w/ DVFS
- ➔ More Performance Issues

Topics

- Introduction
  - DVFS (Dynamic Voltage & Frequency Scaling)
  - QoS (Quality of Service)
- The Issues & Solutions
  - QoS on DVFS devices
  - QoS on DVFS mechanisms
- Conclusion
  - Preliminary Experimental Results

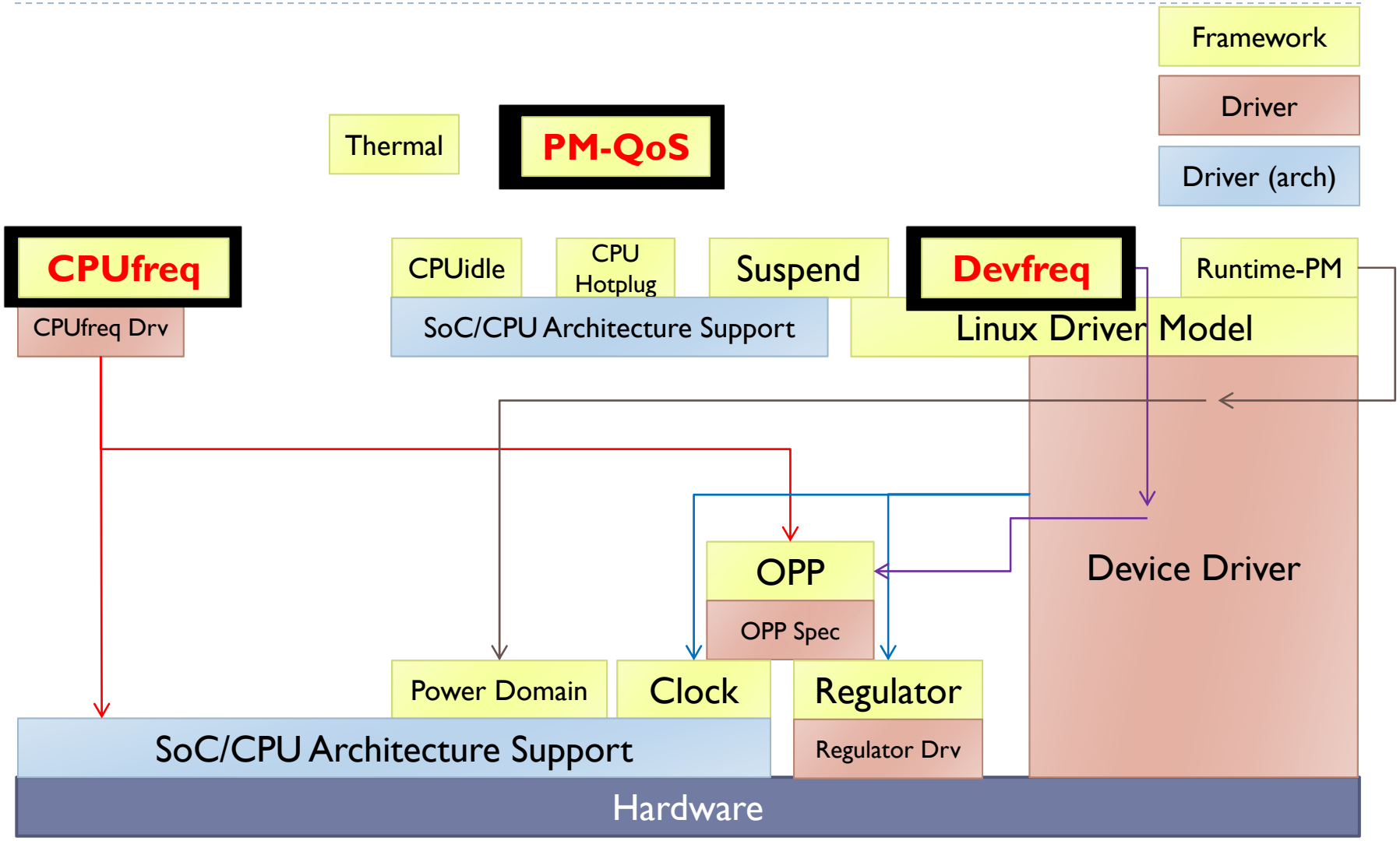
Introduction

- DVFS
- QoS
- Terms

Issues & Solutions

Conclusion

Linux Kernel Power Management
Linux 3.4



- The frameworks are not hierarchical. Connections show typical usage.

Intro: DVFS 1/2

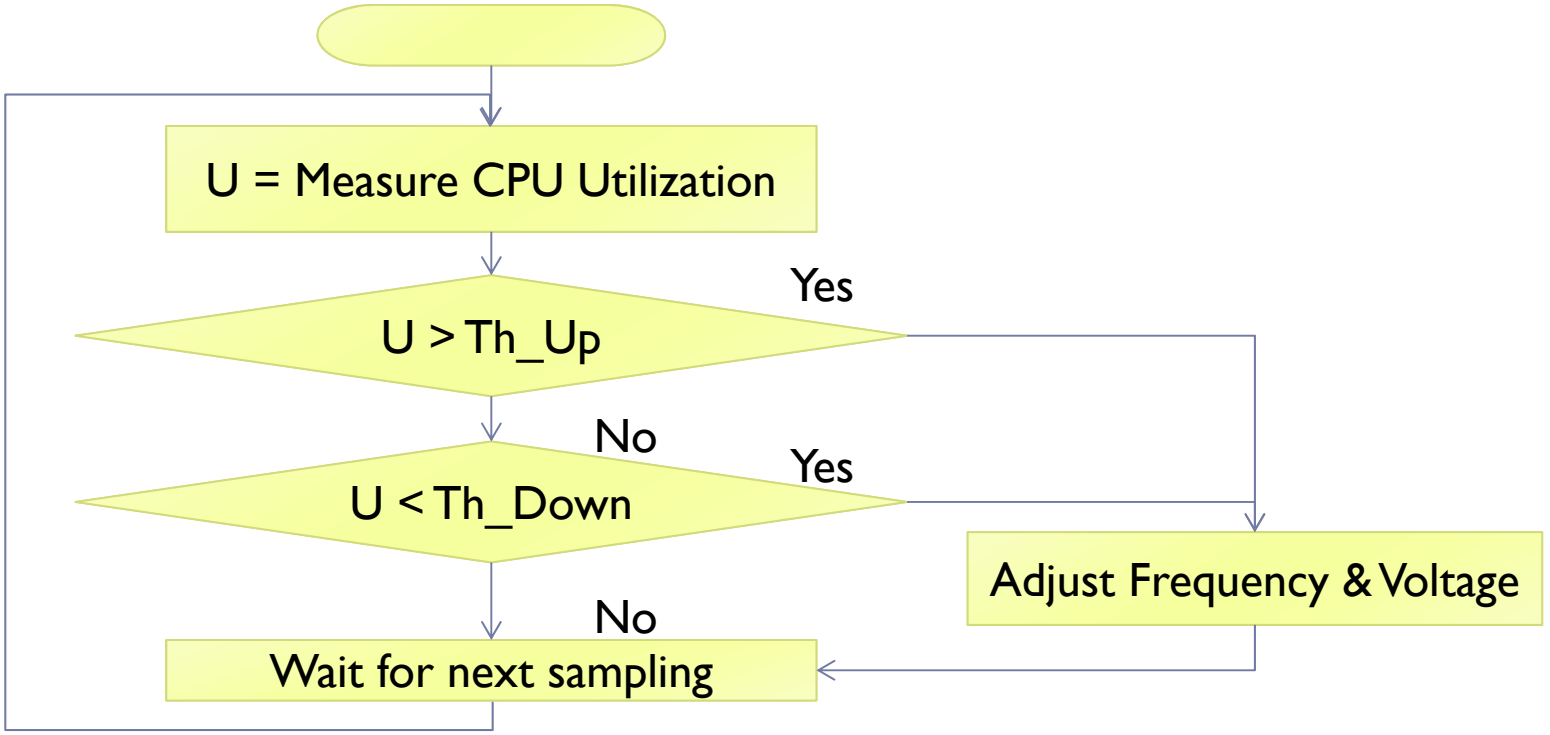
- Dynamic Voltage and Frequency Scaling

$$P = C \bullet f \bullet V^2$$

- CPUfreq: DVFS for CPU Core
- Devfreq: DVFS for other devices
  - Memory-Interface, Bus, GPU, …

Intro: DVFS 2/2

- Sampling the utilization, periodically
- Adjust frequency based on the utilization
- Adjust voltage based on frequency



Intro: QoS

- Linux PM-QoS Framework
  - Global QoS Request
    - "CPU-DMA-Latency", "Network-Throughput", "Network-Latency"
  - Per-dev QoS Request
    - "Device A", "S5Pxxx.0"
  - Manage QoS Requests for QoS Handlers
    1. Thread A: DMA latency < 100us
    2. Thread B: DMA latency < 15us
       - PM-QoS tells DMA driver: "Do < 15us"
    3. Thread B: Cancel the request
       - PM-QoS tells DMA driver: "Do < 100us"
    4. …

Intro: Terms

- DVFS Target
  - A device w/ DVFS capability
  - DVFS Target = CPU w/ CPUfreq
- DVFS Driver
  - Device driver controlling DVFS mechanism of a DVFS target.
  - (CPUfreq) DVFS Driver = "/drivers/cpufreq/exynos4x12-cpufreq.c"

Introduction

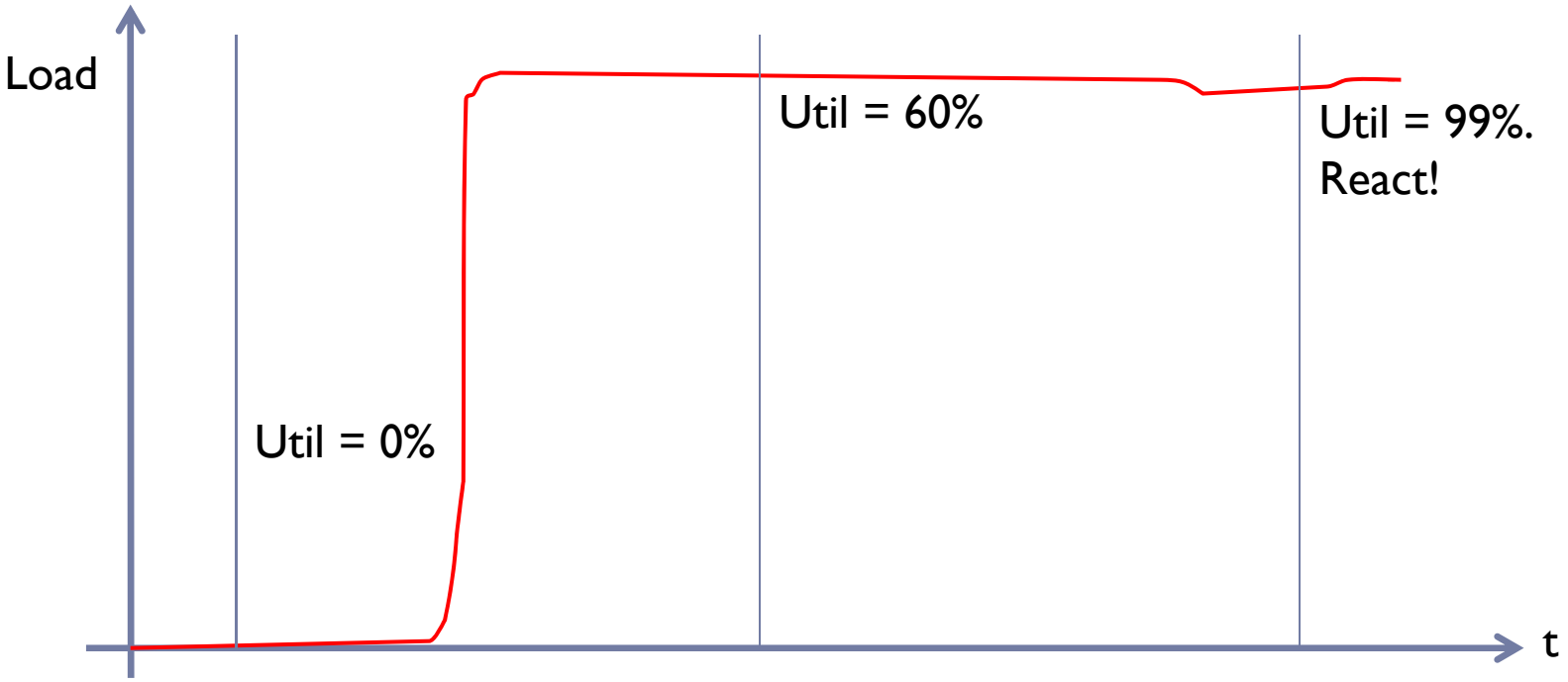
Issues & Solutions

- Performance Issues of DVFS
- QoS on DVFS Devices
- QoS on DVFS Mechanisms

Conclusion

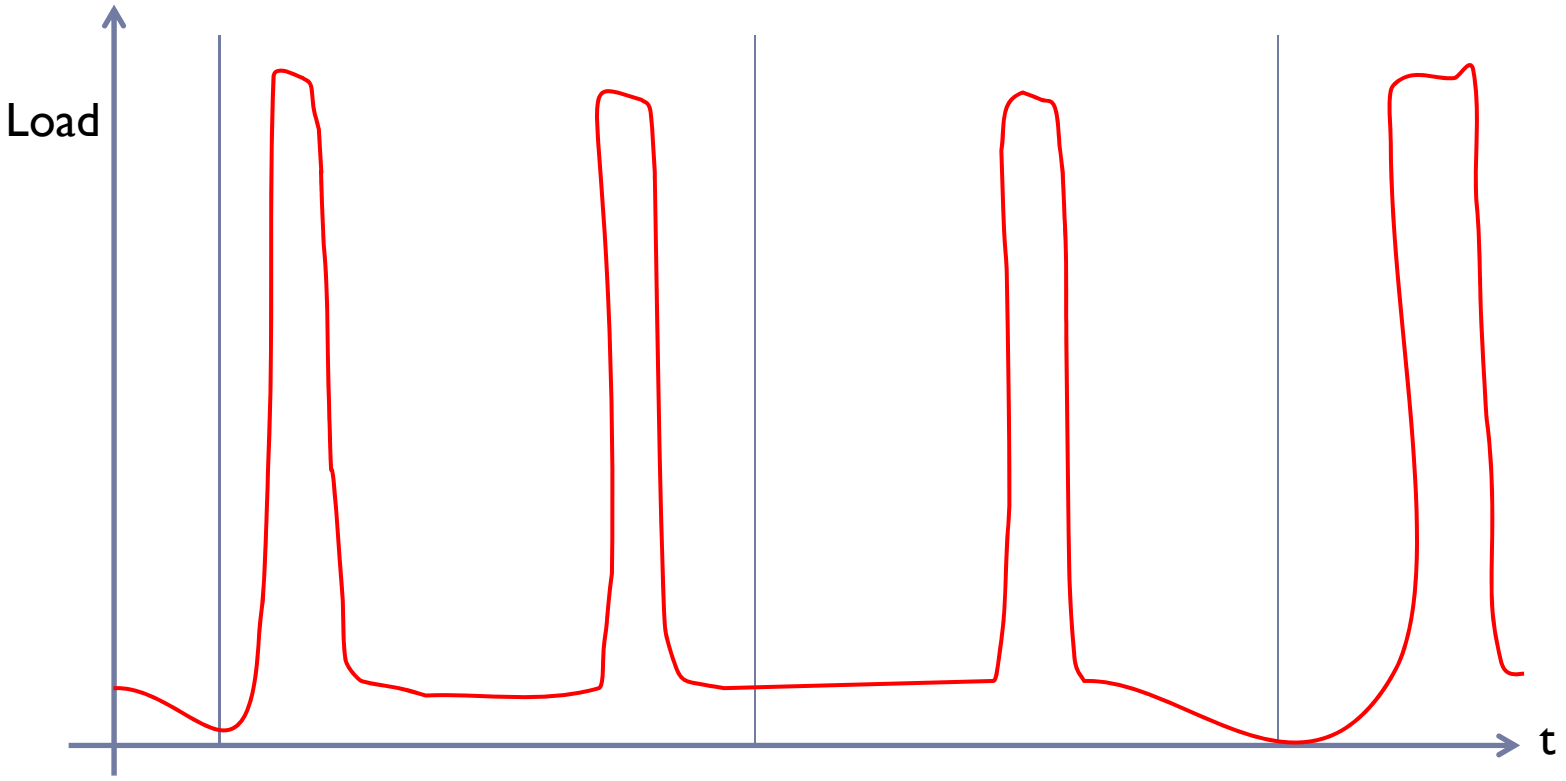
Performance Issues of DVFS: 1/3

- Issue 1: Too Late to React



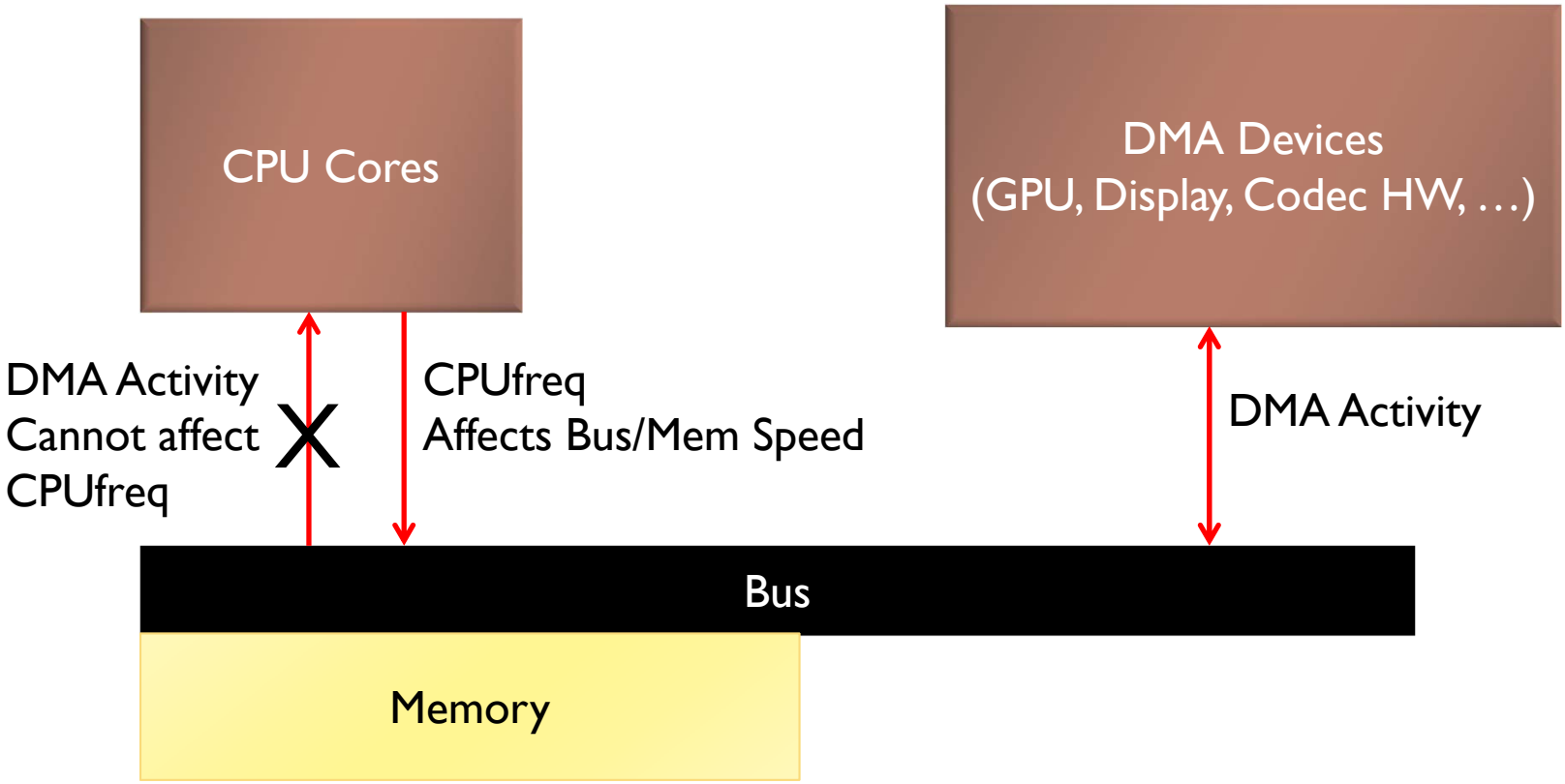
Performance Issues of DVFS: 2/3

- Issue 2: Cannot Detect Short Bursts



Performance Issues of DVFS: 3/3

- Issue 3: Asymmetric Inter-device Dependency
  - DMA op latency/throughput depends on CPU frequency.
  - Activity on DMA doesn't affect CPU load.



Introduction

Issues & Solutions

- Performance Issues of DVFS
- QoS on DVFS Devices
- QoS on DVFS Mechanisms

Conclusion

The Possible Solution

- QoS handling at DVFS devices

- Let DVFS frameworks (CPUfreq/Devfreq) handle not, each DVFS drivers.
  - Based on the table given by DVFS drivers
    - E.g.,

Bus frequency	DMA Throughput
100MHz	640MB/s
133MHz	851MB/s
266MHz	1702MB/s
400Mhz	2560MB/s

QoS on DVFS Devices (DVFS f/w)

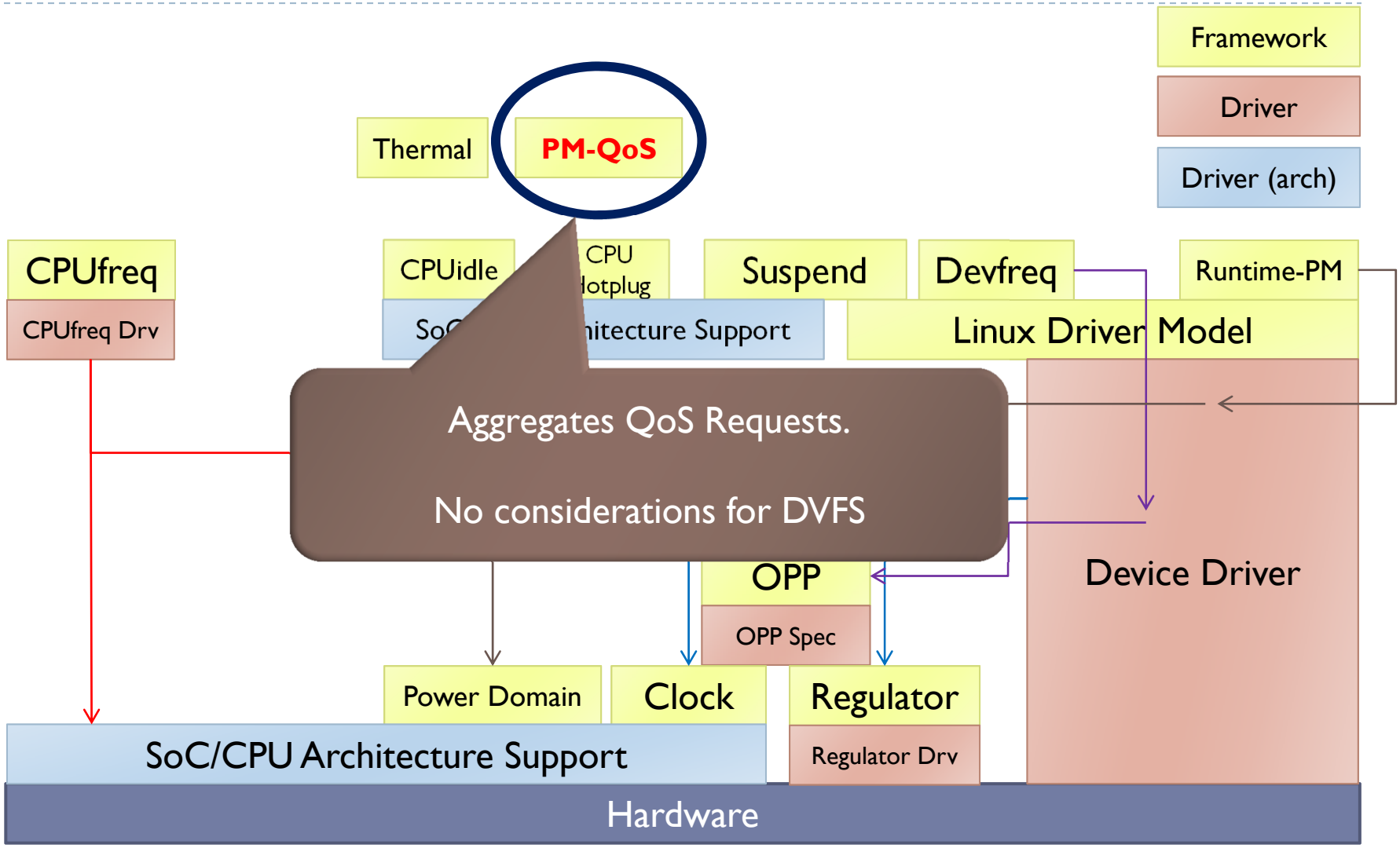
- Add the following information (devfreq driver) at probe
  - QoS-related info for Devfreq driver (/include/linux/devfreq.h)

```
struct devfreq_pm_qos_table {
        unsigned long freq; /* 0 if this is the last element */
        s32 qos_value;
};
struct devfreq_dev_profile {
                        ......
        /* Optional QoS Handling Specification */
        int qos_type;                       /* Global QoS Requests */
        bool qos_use_max;                   /* Throughput-like? Or Latency-like? */
        bool enable_dev_pm_qos; /* Per-dev QoS Requests */
        struct devfreq_pm_qos_table *qos_list;
};
```

QoS on DVFS Devices (DVFS f/w): Works!

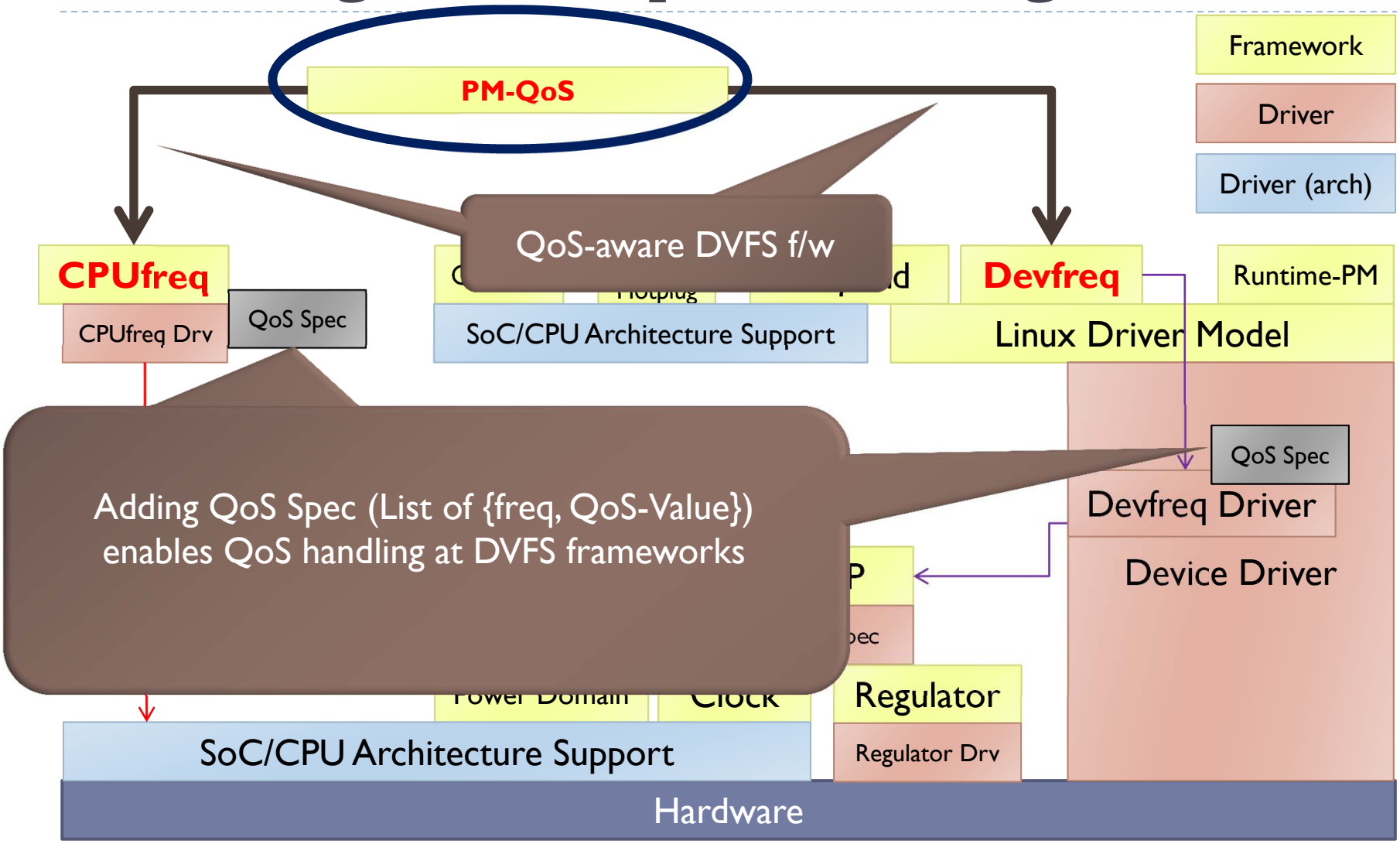
- Issue1, "Touchscreen Event"
  - User touch event → QoS request "at least 1000 BogoMIPs"
    → CPUfreq runs CPU at 1GHz
  - Reacting in ~100us (almost same w/ Issue 2)
- Issue 3, "Video Decoding"
  - Video decoder gets a 1080p60Hz job
    → QoS request "DMA throughput of 2.4GB/s"
    → CPUfreq runs CPU at 500MHz
  - No performance issues

Handling QoS Requests: Design-Before



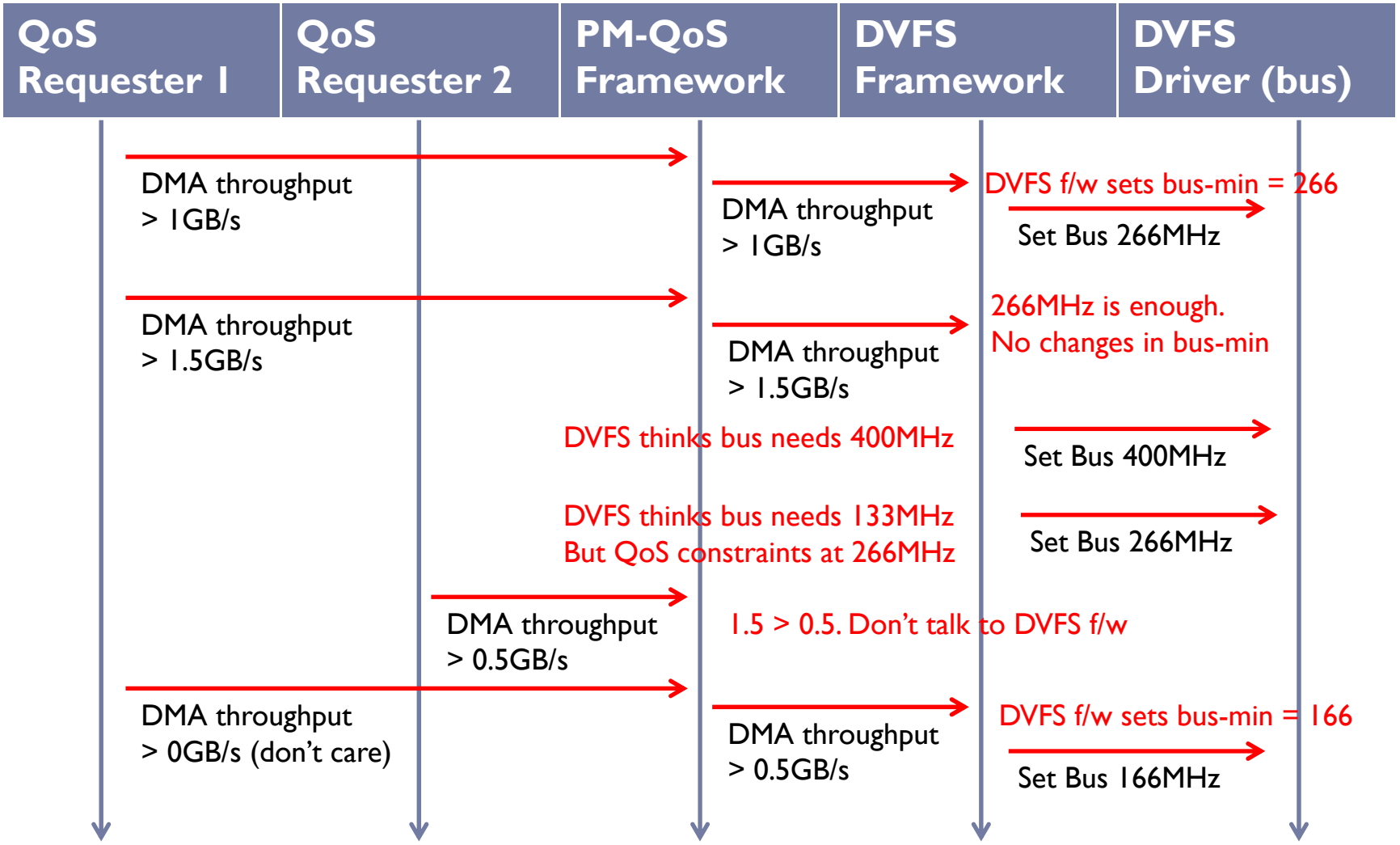
The frameworks are not hierarchical. Connections show typical usage.

Handling QoS Requests: Design-After



The frameworks are not hierarchical. Connections show typical usage.

Handling QoS Requests: How it Works



Handling QoS Requests: Status

- New Global QoS Metrics Required
  - DMA-Throughput: use kbytes/sec?
  - GPU Performance: ???
  - CPU Performance: ??? (BogoMIPS?? MIPS?? Clock?? …)
- QoS-Extension for CPUfreq
  - Work-to-do: Handling global QoS (after we get the metric)
- QoS-Extension for Devfreq
  - Done: Handling global QoS and per-dev QoS
  - In-progress: test & evaluation
- 3.5/3.6 Materials?

Introduction

Issues & Solutions

- Performance Issues of DVFS
- QoS on DVFS Devices
- QoS on DVFS Mechanisms

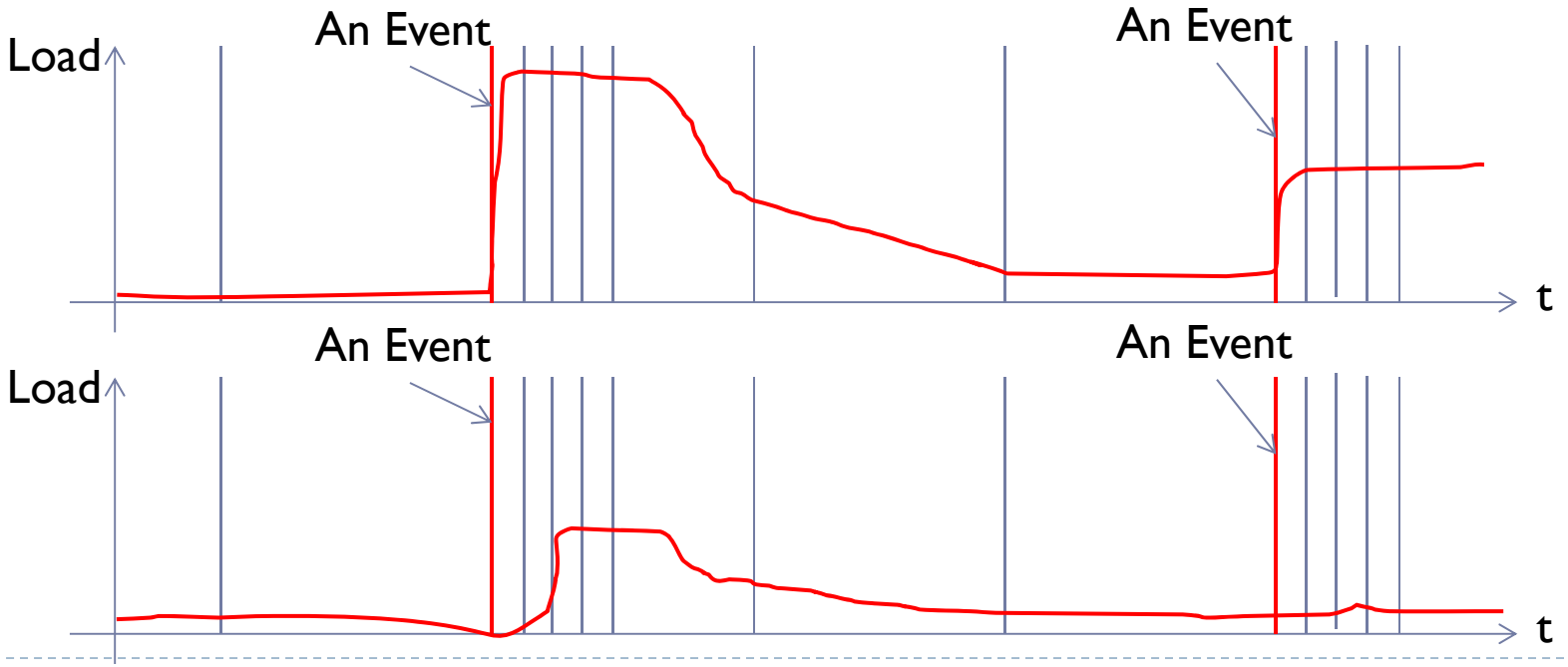
Conclusion

DVFS Response Latency: Motivation

- Reviving the "Issue 1"
  - Responding to user inputs (e.g., mobile phone touchscreen)
    - Case 1
      - Launching an app / Menuscreen flipping
        - Requires high performance. (Nearly full)
    - Case 2
      - Typing a TXT / Email
        - Requires low-mid performance. (Often lowest)
    - QoS requests upon inputs → Unconditional performance increase.
      - Case 2 wastes power!
- Do NOT increase performance unconditionally!
  - Decide faster, not acting blindly.
    - Control DVFS behavior!

DVFS Response Latency: Design

- Request faster reaction from DVFS mechanism.

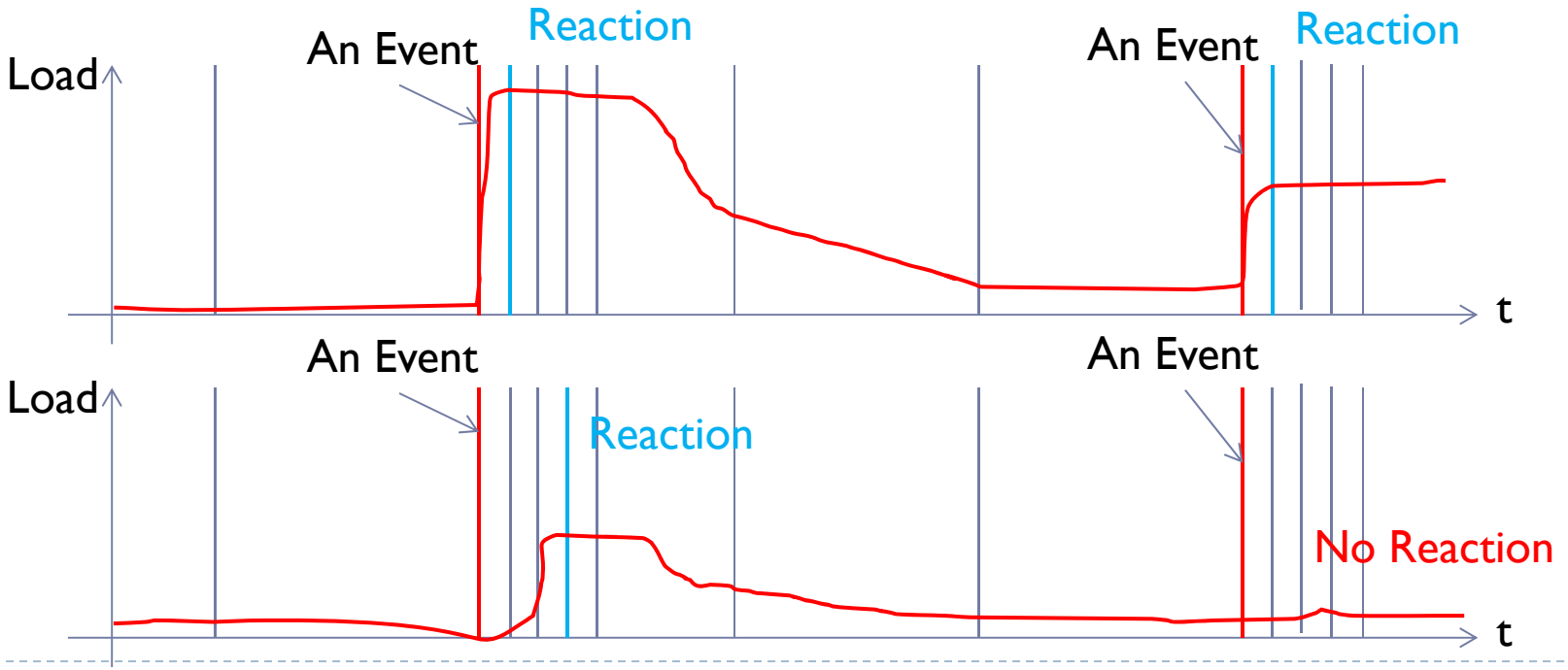


DVFS Response Latency: Design

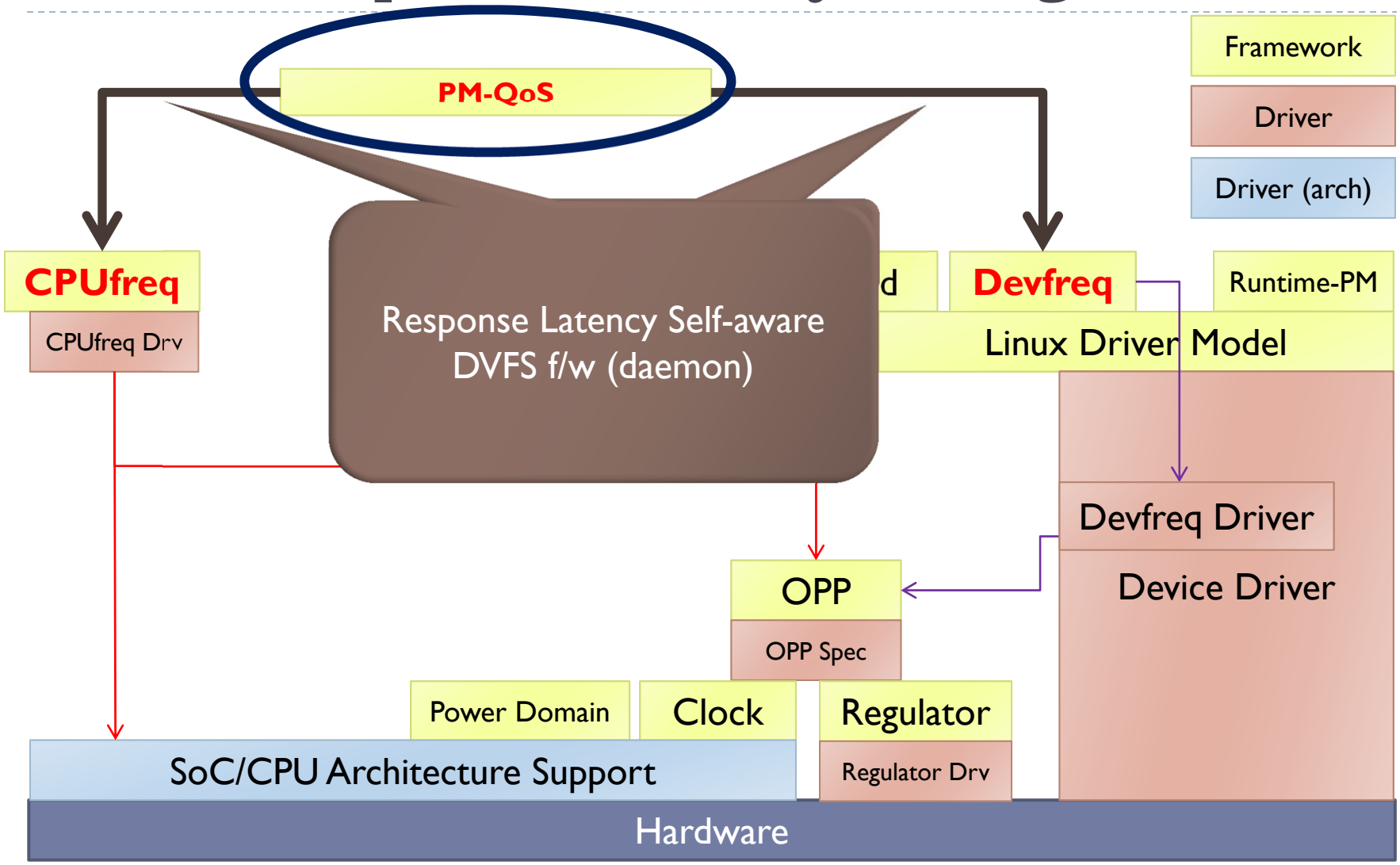
- Request faster reaction from DVFS mechanism.
  - "Quality-of-DVFS-Service"!
    - Add "DVFS_RESPONSE_LATENCY" QoS Class!
    - CPUfre/Devfreq controls sampling rate upon requests

DVFS Sampling

DVFS Sampling
Increasing Freq.



DVFS Response Latency: Design



 The frameworks are not hierarchical. Connections show typical usage.

DVFS Response Latency:(DVFS f/w)

- Add the following information (devfreq driver) at probe
  - QoS-related info for Devfreq driver (/include/linux/devfreq.h)

```
struct devfreq_dev_profile {
                        ......
        /* Optional DVFS-Response-Latency QoS Handling Specification */
        bool support_dvfs_latency;          /* Enable the feature */
};
```

  - The Devfreq f/w will update sampling rate accordingly

DVFS Response Latency: (DVFS f/w)

- The worst-case response latency
  - With sustained full-load.

$$L = (1 + Th) * R$$

  - $T_a \rightarrow 1$ (100%) ➔ $L \rightarrow 2R$
  - L Response Latency
  - Th Up-Threshold
  - R Sampling Rate
- Devfreq f/w sets sampling rate based on this.

DVFS Response Latency: Status

- Used in some product kernels.
- Trying to upstream
  - QoS (global)
    - Metric: "DVFS_RESPONSE_LATENCY" in us?
      - For DVFS drivers registered.
      - (?) Different "classes" of DVFS drivers???
  - CPUfreq
    - Done: Instant reaction patch
    - Work-to-do: redo response-latency after the metric is concluded.
  - Devfreq
    - Done: patchset
    - In-progress: test & evaluation

Introduction

Issues & Solutions

Conclusion

Conclusion – Future Work 1/3

- What's Next – 1
  - Test & Evaluation
    - QoS Handling in Devfreq
    - DVFS-Response-Latency Handling in Devfreq & CPUfreq
  - Development
    - QoS Handling in CPUfreq
  - QoS Metrics
    - DVFS-Response-Latency?
    - DMA Throughput
    - GPU???

Conclusion – Future Work 2/3

- What's Next – 2
  - Thermal-aware DVFS
    - Integrating w/ Thermal f/w
    - DVFS driver = Thermal cooler device
  - Scheduler-aware DVFS
    - Turbo Boost*-like Support
      - Adjust MAX freq according to # cores activated
        - # Threads ➔ # Cores activated

*Trademark of Intel

Conclusion – Future Work 3/3

- Future Work (farther…)
  - (Many) Multi/Hetero-core DVFS Support
    - Scheduler support might kick in.
    - ARM Cortex A15 Big-Little model is the starting point.
  - Compiler/Algorithm support for DVFS mechanisms
    - Had some approaches, but not efficient enough (yet)

Thank you!

Appendix

Appendix: Links to Related Code

- Devfreq
  - Linux 3.4 rc7 Tovalds'
    - Header / Core / Default-Governor (Daemon)
- Devfreq/CPUfreq/PM-QoS for Linux 3.5/3.6
  - PM / devfreq: handling QoS reqest on DVFS response latency (work-in-progress)
  - PM / devfreq: support per-dev PM-QoS in devfreq (not sent)
  - CPUfreq ondemand: handle QoS request on DVFS response... (pending)
  - CPUfreq ondemand: update sampling rate without waiting... (accepted)
  - PM / QoS: add pm_qos_update_request_timeout API (accepted)
  - PM / QoS: Introduce new classes: DMA-Throughput and... (pending)
  - PM / devfreq: add relation of recommended frequency. (accepted)
  - PM / devfreq: add PM QoS support (pending)