A Workload-Aware DVFS Robust to Concurrent Tasks for Mobile Devices

Chengdong Lin$^{1,2}$, Kun Wang$^1$, Zhenjiang Li$^1$, Yu Pu$^2$

City University of Hong Kong$^1$, Alibaba DAMO Academy$^2$
Motivation - Power and Thermal Management

Mobile Edge Devices

Computing power drops significantly e.g., 40~50% reduction

Compromising device and application reliability
DVFS (Dynamic Voltage and Frequency Scaling) - OS-level Tool

\[ P \propto f^3 \]

**Power Consumption**

**Processor Frequency**

---

**Dynamic Workloads**

- Gaze
- Speech
- Network
- ... APP N

**DVFS Decision**

- **CPU**
- **GPU**

---

**Frequency Table**

<table>
<thead>
<tr>
<th>A15 Cluster of Exynos 5422</th>
<th>A7 Cluster of Exynos 5422</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency(KHz)</td>
<td>Voltage(uV)</td>
</tr>
<tr>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td>2000000</td>
<td>1250000</td>
</tr>
<tr>
<td>1900000</td>
<td>1250000</td>
</tr>
<tr>
<td>1800000</td>
<td>1250000</td>
</tr>
<tr>
<td>1700000</td>
<td>1250000</td>
</tr>
<tr>
<td>1600000</td>
<td>1250000</td>
</tr>
</tbody>
</table>

**DVFS Decision**

- **Match**
- **Require**

---

This diagram illustrates the relationship between power consumption, processor frequency, and dynamic workloads in the context of DVFS (Dynamic Voltage and Frequency Scaling) for OS-level tools. The frequency table provides specific instances of frequency and voltage combinations for different clusters of Exynos 5422 processors.
Questions

Q1. How to represent dynamic workloads?

Q2. How to make suitable frequency decision?
Prior Arts: Application Specific

Application based Solutions:

- **QoS as Indicator**
  
  QoS: frame rate, response latency, ....

- **Rationale:** keep QoS at a desired level
  
  e.g., fps = 30

Multitasks have multi QoS
Prior Arts: Application Agnostic

General OS Solutions:
• **Utilization** as Indicator
  \[ u = \frac{T_{busy}}{T_{win}} \]
• Rationale: keep utilization at a desired level
e.g. \( u = 80\% \)

![Diagram showing utilization over time with equation and graph](image)

\[ \text{freq}_{\text{next}} = 1.25 \times \text{freq}_{\text{curr}} \times \text{util} \]

Not Accurate, why?
Representation of Workload-Awareness context

Core Utilization Formula

\[ u = \frac{T_{busy}}{T_{win}} \]

![Diagram showing CPU cycles and their breakdown into busy, stalled, and idle categories.](image)
Representation of Workload-Awareness context

Core Utilization Formula

\[ u = \frac{T_{busy}}{T_{win}} \]
Representation of Workload-Awareness context

Core Utilization Formula

\[ u = \frac{T_{\text{busy}}}{T_{\text{win}}} = \frac{T_{\text{act}} + T_{\text{sta}}}{1} = \beta \times \frac{W_{\text{act}}}{f} + T_{\text{sta}} \]

\[ u = \frac{a}{f} + b \]

workload-awareness context

Different \( <a,b> \)

Different curves
Representation of Workload-Awareness context

Our work

Past works

---

\[ (f^0, u^0) \]

---

<table>
<thead>
<tr>
<th>Process View</th>
<th>Busy</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Cycles</td>
<td>Act. Cycles</td>
<td>Stalled Cycles</td>
</tr>
</tbody>
</table>

Total CPU Cycles
Representation of Workload-Awareness context

\[ u = \frac{a}{f} + b. \]

System Workloads \(<a,b>\)
Learn Workload-Awareness context - Hardware Meta-data

- **Hardware statistics of Processors**
  - CPU\GPU utilization (active & stalled cycles)
  - Cache hits/misses
  - Frequency
  - Temperature

- **No Labeling effort**
  - OS generate hardware statistics automatically

Meta-Data $\rightarrow$ $<a,b>$

How?
Learn Workload-Awareness context - Meta-state Learner

$X(t)$: Hardware raw observation data

$Z(t)$: Intermediate latent variable

$m(t)$: Using LSTM to capture the temporal feature
RL-based solution

$$ (P) : \min \frac{1}{T \times M} \sum_{t=1}^{T} \sum_{i=1}^{M} |u_i(t) - u_i^q|,$$

s.t. $$f_i^{\min} \leq f_i(t) \leq f_i^{\max}, \ \forall i, t,$$

$$c_i(t) \leq c_i^{\text{thermal}}, \ \forall i, t.$$

State: Meta-State

Action: Select the Frequency

Reward Function: Need to Design
Make DVFS decision - Large action space challenge

Frequency Action Search Space:

\[
\begin{array}{c|c}
\text{A15 Cluster of Exynos 5422} & \text{A7 Cluster of Exynos 5422} \\
\hline
\text{Frequency(KHz)} & \text{Frequency(KHz)} \\
2000000 & 1400000 \\
1900000 & 1300000 \\
1800000 & 1200000 \\
1700000 & 1100000 \\
1600000 & 1000000 \\
1500000 & 900000 \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{Voltage(μV)} & \text{Voltage(μV)} \\
1250000 & 1250000 \\
1250000 & 1250000 \\
1250000 & 1250000 \\
1250000 & 1250000 \\
1100000 & 1100000 \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{CPU} & \text{CPU} & \text{GPU} \\
12 & 12 & 13 = 1872 \\
\text{(big)} & \text{(little)} & \\
\end{array}
\]
Make DVFS decision

Action Branching based Deep Q-Network

State $S_{MS}$ → Feature → Feature → Feature

Branches
- Q-Values Big-Cores
- Q-Values Little-Cores
- Q-Values GPU

Power Domains

Search Space: $12 + 12 + 13 = 37$
- BIG
- Little
- GPU
Implementation

Proposed DVFS Framework

1. Self-driving (4 tasks): lane detection, object detection, segmentation, depth estimation
2. Robot (3 tasks): object detection, video uploading, speech recognition
3. UAV ground station (2 tasks): object detection, multi-stream video receiving
4. Smartphone APPs (3 APPs): Tik Tok, PUBG, Zoom

Scenarios
"zTT: Learning-based DVFS with Zero Thermal Throttling for Mobile Devices." ACM MobiSys 2021
Evaluation Results - Compare with the SOTA

"zTT: Learning-based DVFS with Zero Thermal Throttling for Mobile Devices." ACM MobiSys 2021
Evaluation Results - System Overhead

Overhead: ~100 mw
Reduction: ~2W

System Overhead on Jetson NX
Conclusion

- We explored recent DVFS solution is not effective, introduced new hardware based DVFS metrics.

- We proposed a DVFS framework that could learn dynamic workload contexts in multitasks scenarios.

- We developed a prototype DVFS system to demonstrated its efficiency on different embedded platforms.
Thanks!

Scan to visit the project website, or type GearDVFS.github.io