LUSH: Lightweight Framework for User-level Scheduling in Heterogeneous Multiprocessors

Vasco Xu, Liam White McShane, Daniel Mossé (University of Pittsburgh)

The IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-2021)
**Norm:** Heterogeneous Multicore Architecture

- **Homogeneous Processors:**
- **Heterogenous Processors:**
Norm: Heterogeneous Multicore Architecture
Embedded processors have become increasingly configurable with a number of resources that can be managed through software, such as core type and frequency.
Embedded processors have become increasingly configurable with a number of resources that can be managed through software, such as core type and frequency.

To meet performance guarantees while minimizing energy usage requires intelligent management and scheduling of such resources.
State-of-the-Art Paradigm

Sensing ➔ Performance Metrics ➔ Controller ➔ Resource Allocation ➔ Application System
Challenges: Large configuration space, dynamic environment, low overhead
Challenges: Large configuration space, dynamic environment, low overhead
State-of-the-Art

Challenges: Large configuration space, dynamic environment, low overhead
**Challenges:** Large configuration space, dynamic environment, low overhead
State-of-the-Art

**Challenges:** Large configuration space, dynamic environment, low overhead
**State-of-the-Art**

**Challenges:** Large configuration space, dynamic environment, low overhead

- **Core Frequency**
- **Core Type**
- **Number of processes per chip**
- **How much memory to allocate?**
- **What type of memory to use?**

**Performance**

**Metrics**

**Controller**

**Resource Allocation**

**Application**

**System**
Challenges: Large configuration space, dynamic environment, low overhead
### Previous Work

#### Controller Design
- Fair Energy-Efficient Scheduling [1]
- Estimate Per-Thread Performance and Power [2]
- Approximate Computing [3]

#### Framework Design
- POET [5]
- BEEPS [6]

Previous Work

Previous Work

Controller Design

1. Fair Energy-Efficient Scheduling [1]
2. Estimate Per-Thread Performance and Power [2]
3. Approximate Computing [3]

Framework Design

1. POET [5]
2. BEEPS [6]

Previous Work

SPDA Paradigm

Common Paradigm

Sensing → Performance Metrics → Controller → Resource Allocation → Application

Sensing → Performance Metrics → Prediction → Application Behavior → Decision

Prediction → Decision → Actuation → Resource Allocation → System
LUSH Framework: sensing

A parameterized and flexible interface (targeted at researchers) for developing, deploying, and evaluating novel algorithms at user-level applied to OS scheduling policies.
Fine-grained data collection \textbf{(at every context-switch)} with low overhead.

Suitable for offline profiling of applications.

Serves to build good machine learning models.
A parameterized and flexible interface (targeted at researchers) for developing, deploying, and evaluating novel algorithms at user-level applied to OS scheduling policies.
A parameterized and flexible interface (targeted at researchers) for developing, deploying, and evaluating novel algorithms at user-level applied to OS scheduling policies.
LUSH Framework: *decision and actuation*

A parameterized and flexible interface *(targeted at researchers)* for developing, deploying, and evaluating novel algorithms at **user-level** applied to OS scheduling policies.
Experimental Setup

**ODROID-XU3**
- Powered by Samsung Exynos5422 SoC with four Cortex-A15 big cores and four Cortex-A7 LITTLE cores.
- 2GB RAM.

**Google Pixel 4a**
- Powered by two Cortex-A76 big cores and six Cortex-A55 LITTLE cores.
- 6GB RAM.
- Capture power measurements by polling respective power counters.

### Experimental Evaluation

#### Sensing Comparison ($\mu$s)

<table>
<thead>
<tr>
<th></th>
<th>Modified Kernel</th>
<th>Unmodified Kernel (perf)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>23</td>
<td>19</td>
</tr>
</tbody>
</table>

#### Framework Overhead

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Runtime Overhead (%)</th>
<th>Energy Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Pixel 4A</td>
<td>0.043</td>
<td>0.233</td>
</tr>
<tr>
<td>ODROID-XU3</td>
<td>0.625</td>
<td>0.069</td>
</tr>
</tbody>
</table>
Potential Application: Application Phases

1: Application Phases

- Applications experience different phases during execution (memory-intensive or CPU-intensive).
- Migrating the application to the appropriate core based on the phase, can potentially lead to power savings.
- Preemptively predict, using machine learning the upcoming phase, and migrate thread accordingly.

*astar* benchmark from the SPEC2006 suite
Potential Applications

2: Heuristic-Based Scheduling

- Memory-intensive programs exhibit significant number of L2 cache misses.
- Change scheduler to base its affinity decision on a ratio of L2 cache misses to instructions retired.
- Migrate cores on a ratio of 0.1% cache misses to instructions.

custom memory-intensive benchmark\(^1\)

\(^1\)https://github.com/Pitt-RASG/benchmarks/blob/master/randomwalk.c
Conclusion

1. Present LUSH, a lightweight framework for user-level scheduling in heterogeneous multiprocessors.

2. Present a fine-grained OS-level monitoring tool that tracks PMCs at every context-switch with predictable low overhead.

3. Describe the implementation of a framework that allows researchers to develop OS scheduling policies without requiring kernel expertise or time-consuming kernel development.

Reference Implementation: https://github.com/Pitt-RASG/LUSH