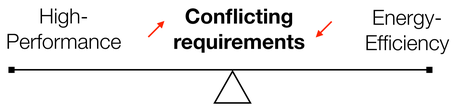
Eve: An Event-Driven Processor Architecture For Energy-Efficient Mobile Computing

Yuhao Zhu, Dustyn Blasig **Advisor: Vijay Janapa Reddi**
The University of Texas at Austin

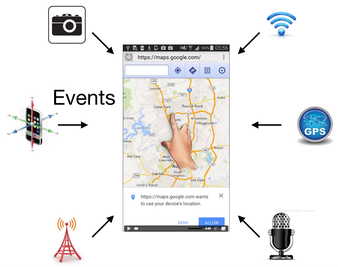
1. INTRODUCTION

Mobile devices must satisfy two requirements: **high-performance** and **energy-efficiency**. The limited battery capacity and the demise of Dennard scaling limit today's mobile processors from satisfying both goals.



We propose to design a new customized processor architecture that can improve both energy-efficiency and the performance of mobile computing by orders of magnitude.

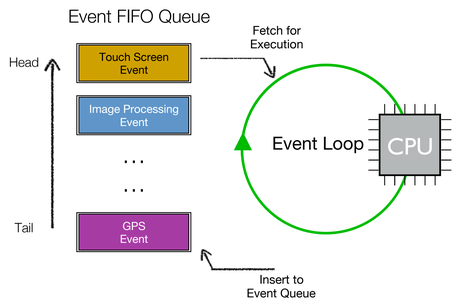
Key idea: To use the **event-driven** nature of mobile applications to drive the architecture design.



2. BACKGROUND

Event-driven applications are executed in the following manner:

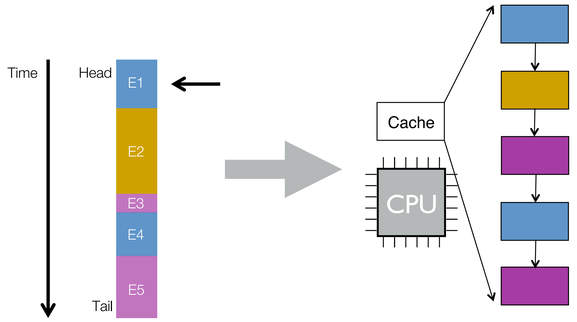
- Each event is registered with an event callback.
- Whenever an events is triggered, its callback handler is inserted to a FIFO-like queue.
- A single threaded event loop **sequentially** fetches and executes events from the queue.
- An event callback can then generate new events that are queued into the FIFO.



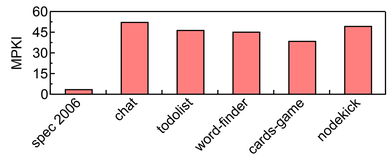
3. INEFFICIENCIES OF CPUS

The sequential, in-order event processing leads to three major sources of inefficiencies in current generation mobile CPUs.

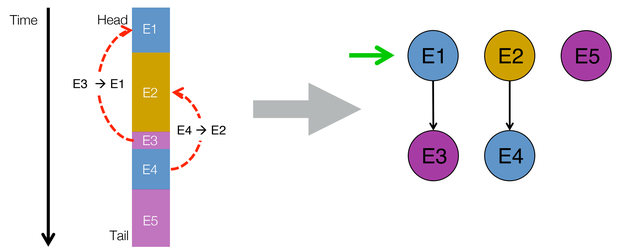
- **Poor locality**: The interleaved execution of different events types can destroy instruction locality by polluting micro-architectural structures, such as caches and branch predictors.

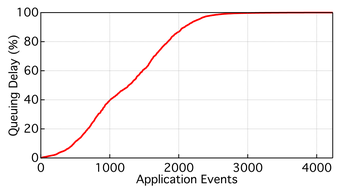


E.g.: L1 instruction cache MPKI of event-driven applications is 8~9X higher than SPEC CPU 2006 applications.



- **Lack of parallelism**: Events in event-driven applications are executed by CPUs sequentially. In-order event processing does not identify inherent event-dependency, and therefore cannot exploit event-level parallelism when available.



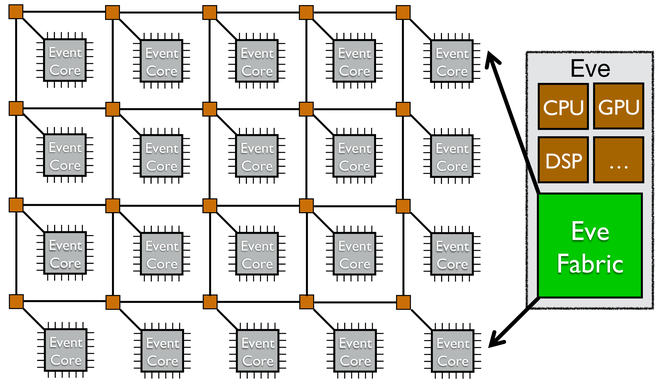


E.g.: Over 3,000 out of 4,000 Google Maps events spend over 40% of their execution time waiting in the event queue. These events could have executed in parallel.

- **Hard to prioritize**: Sequential event processing does not take into account the difference in priority between events, and as such it is unable to distinguish and prioritize important events from unimportant ones to execute them immediately.

4. EVE PROCESSOR ARCHITECTURE

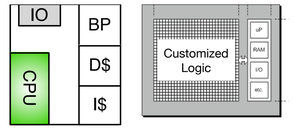
At the center of Eve is a tile of event cores that exploit potential event-level **parallelism**. Event cores are connected to an on-chip network that allows event cores to communicate with each other.



Each event core is designed to handle one (or a few) specific event type. Event types can thus experience the benefits of isolated execution contexts, and therefore instruction **locality** is improved.

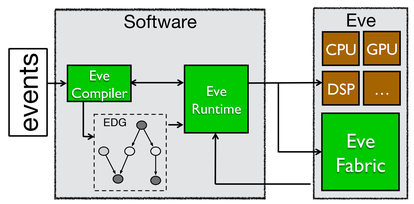
Two potential types of event cores include:

- A general-purpose CPU with private cache/branch predictors. They execute different events at different times.
- A fully customized event-processing logic tuned for critical and frequently executed events to maximize efficiency.



Type I: CPU Type II: Customized logic

5. EVE SYSTEM ARCHITECTURE



Eve requires system-level software support involving the Eve static compiler and the Eve runtime system.

Eve compiler identifies event dependencies, and constructs an Event Dependency Graph (EDG) to enable parallel execution.

Eve runtime decides where, how, and when events are executed based on runtime feedback information (i.e., feedback-driven optimization)

- **Where**: Eve runtime maps events to hardware to i) maximize parallelism, and ii) minimize communication. Frequently communicated events are mapped to adjacent cores or co-located in the same core.
- **How**: e.g., what frequency and/or voltage setting an event core runs to minimize energy while meeting end-users' QoS requirement.
- **When**: Eve runtime decides when various event callback handlers can start execution in parallel according to the EDG constraints.

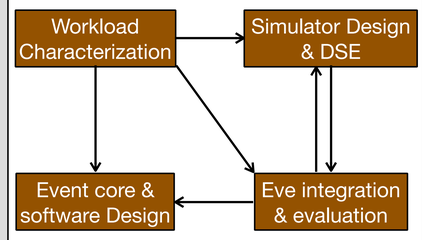
6. ROADMAP

Ultimate goal for Eve

- A hardware (FPGA) prototype and software toolchain for any event-driven (mobile) applications.

Four key steps in realizing Eve

- Workload characterization.
- Simulator design and design space exploration (DSE).
- Event core and Eve software design.
- Eve integration and assessment.



Deliverables for the first year

- A benchmark suite for event-driven applications.
- A parameterized, analytical-model based architectural simulator for Eve.
- An Eve compiler that can construct event dependency graph (EDG).

7. BEYOND MOBILE