Dynamic Runtime Optimizations for Systems of Heterogeneous Architectures

Geoffrey Phi Tran
Graduate Research Assistant
Ming Hsieh Dept. of Electrical Engineering
University of Southern California / Information Sciences Institute
Dong-In Kang, Ph.D.
University of Southern California / Information Sciences Institute
Stephen P. Crago, Ph.D.
Deputy Director, Computational Systems and Technology
Research Associate Professor, Ming Hsieh Dept. of Electrical Engineering

September 10, 2014

4676 Admiralty Way
Marina del Rey, CA

3811 N Fairfax Drive
Arlington, VA

Outline

- Introduction
- Problem Description
- Hierarchical Task Model
- Optimizations
- Simulation
- Experimental Results and Analysis
- Conclusion and Future Work

Introduction

- **Embedded processors are becoming more heterogeneous and parallel, providing a rich area for optimizations**
  - Leads to more efficient performance in similar power envelope
- **Objective: Minimize energy consumption while meeting deadlines for dynamic tasks**

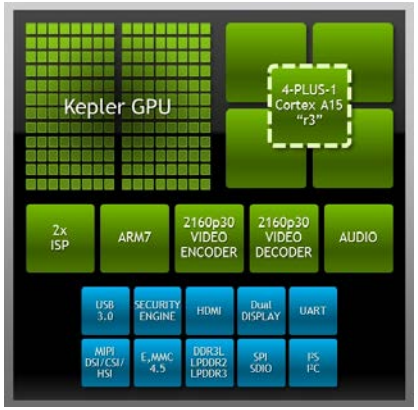


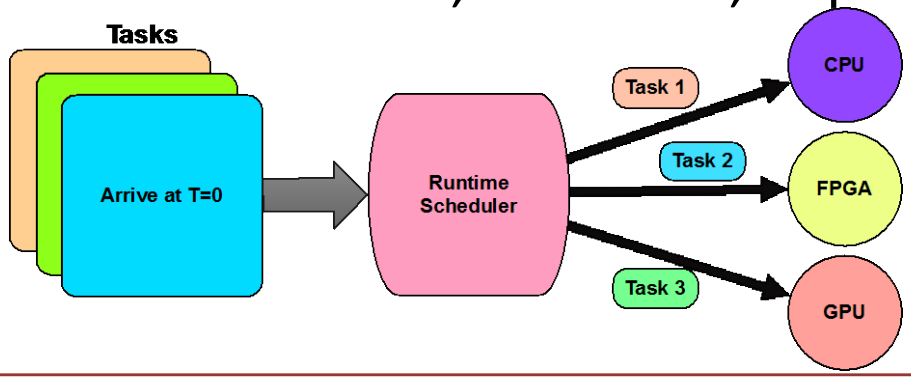
Image credit: NVIDIA Tegra K1 Whitepaper

Problem Description (1)

- **Given: heterogeneous computing architecture and dynamically arriving tasks**
  - Sensors
  - User Input
  - Modes
- **Problem: execute tasks in such a way that the energy consumed and deadlines missed are minimized**

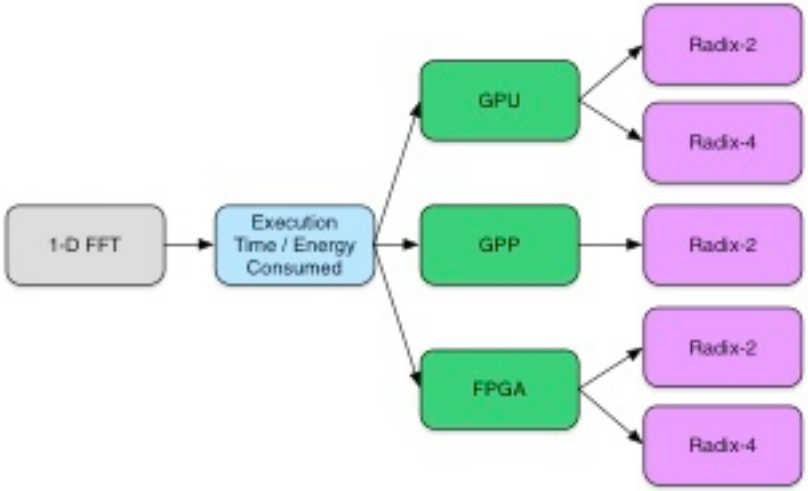
Problem Description (2)

- **Proposed Optimization**
  - Tasks and applications to be executed are submitted to runtime scheduler
  - Scheduler makes decisions in real-time and assigns tasks to compute nodes
- **Required Data**
  - Architecture models
  - Task models: execution time, deadlines, dependencies



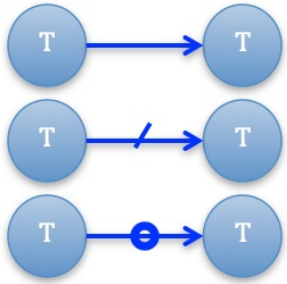
Hierarchical Task Model (1)

- **Models contain three characteristics of tasks**
  - Execution time (per computational unit)
  - Energy consumed (per computational unit)
  - Dependency relationships between tasks
- **Runtime characteristics, such as execution time, are deterministic**



Hierarchical Task Model (2)

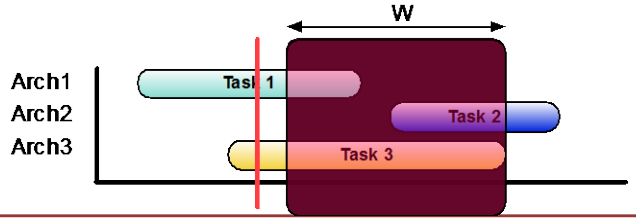
- **Tasks may be dependent on other tasks**
  - One-to-one
  - One-to-many
  - Many-to-one



- **Number of dependent tasks may be deterministic or stochastic**
- **May represent data or control dependencies, we modeled both**

Scheduling Algorithms (1)

- **Greedy: Assign to most efficient node that becomes available**
- **Greedy with DVFS: Greedy schedule, then reduces frequency (F) and voltage (V) to lowest speed that meets deadline**
- **Time-Window (TW): Waits for time window *W*, for most efficient resource to become available**
- **Time-Window with DVFS: Schedules as TW, but reduces *F* to lowest speed that meets deadline**

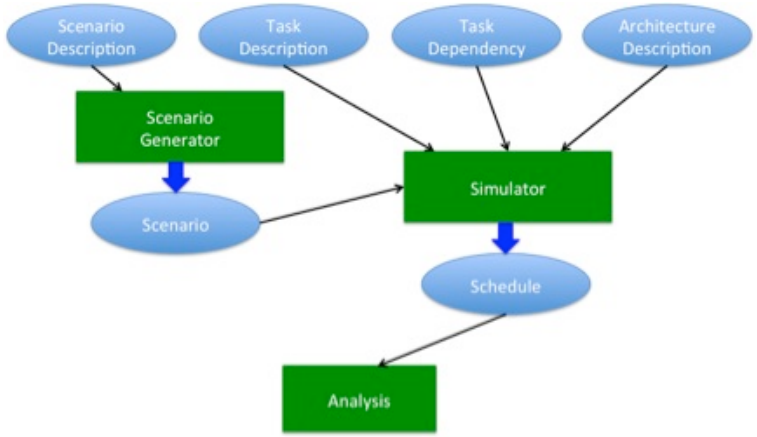


Scheduling Algorithms (2)

- **Time-Window with Local Queues (LQ): New data structure to keep track of execution times for each task in local resource queus. Tasks submitted to local queues in each compute node**
- **Time-Window with Local Queues and DVFS: Schedules with lowest *F* and *V* that still meets deadline**
- **Runtime DVFS Adjustment: Works in conjunction with algorithms that have LQ enabled**
  - Allows a local scheduler on each resource to modify DVFS parameters for each task in its local queue

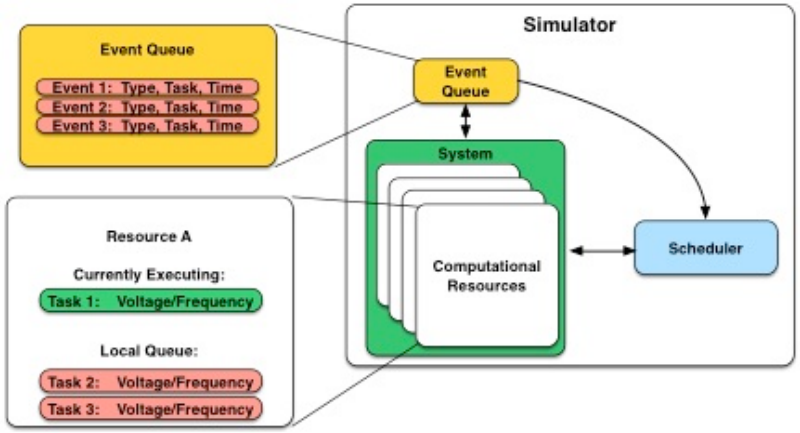
Simulation Tool Introduction

- Created simulator to collect data on performance of scheduling algorithms
- Simulated scheduling decisions and resource availability, not task execution
- Scenario generator used to convert description of tasks, periods, and deadlines to an instantiated scenario



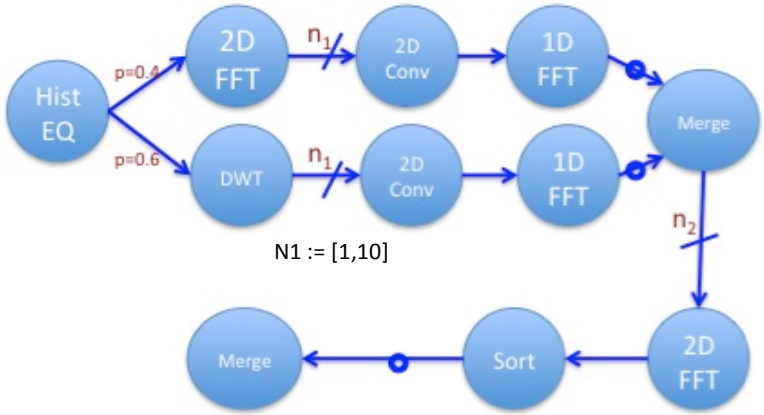
Simulation Tool Details

- **Modularity of scheduler allows different schedules to be implemented**
- **Event queue tracks task arrivals, execution completion, window expiring, etc.**
- **Resources**
  - Current task and voltage/frequency pair
  - Future tasks in queue



Experimental Setup (1)

- **Representative task set scenario**
  - Probabilistic number of dependent tasks, noted by labeling edges
  - Various modes
- **Metrics: energy consumed, number of missed deadlines**
- **Baseline for comparison: Greedy**
- **For each algorithm, 10 runs of 10,000 periods were simulated**
- **Task modeling populated using experimental results and projections**

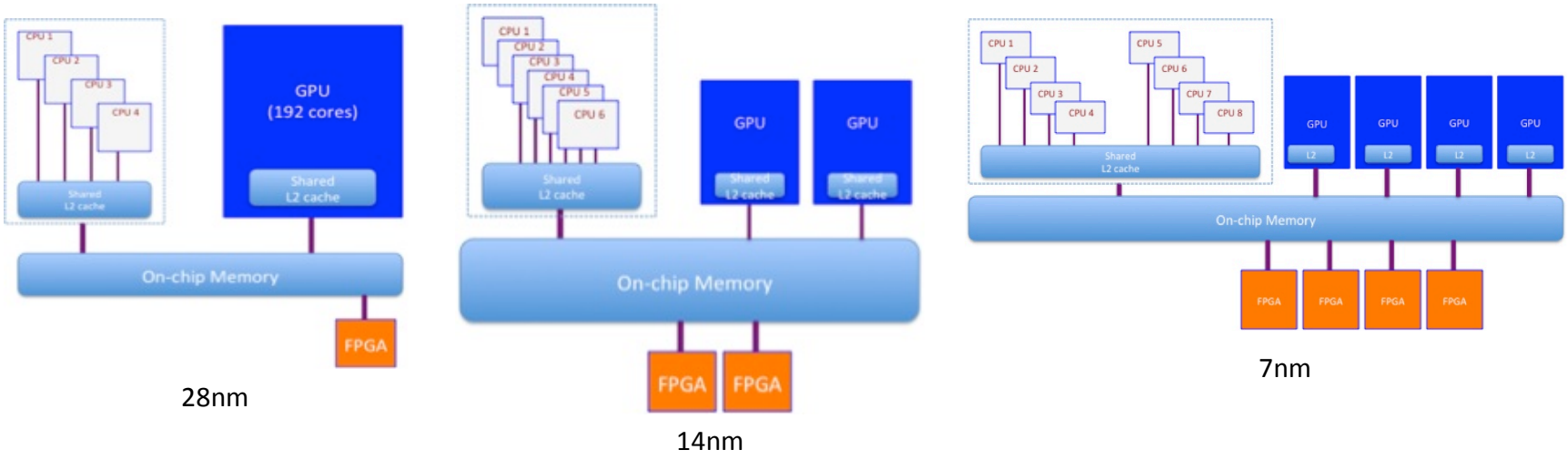


Mode	P^+	P^0	P^-	n_{max}
HI	0.5	0.3	0.2	20
MD	0.3	0.5	0.2	8
LO	0.3	0.5	0.2	2

Parameters for Modes and n2

Experimental Setup (2)

- Assumption: computational units (CU) may have different Dynamic Voltage and Frequency Scaling (DVFS) levels and are turned off when not in use
- Communication cost is paid by the producer
- Three systems represent nodes based on scaling

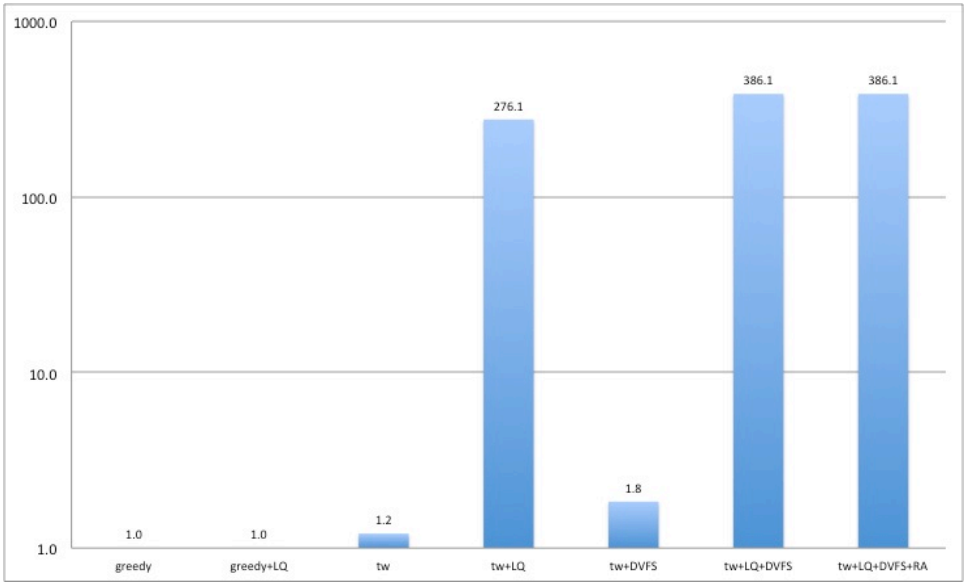


Experimental Results and Analysis (1)

- Use of local processor queues brings large improvement by leveraging known runtimes
- DVFS brings performance slightly higher
- Runtime adjustment brings no further improvement in this case

	28nm	14nm	7nm
Greedy	0.00%	0.00%	0.00%
Greedy+LQ	0.00%	0.00%	0.00%
TW	0.21%	0.48%	0.32%
TW+LQ	0.40%	0.71%	1.01%
TW+DVFS	0.20%	0.45%	0.30%
TW+LQ+DVFS	0.40%	0.71%	1.00%
TW+LQ+DVFS+RA	0.40%	0.71%	1.00%

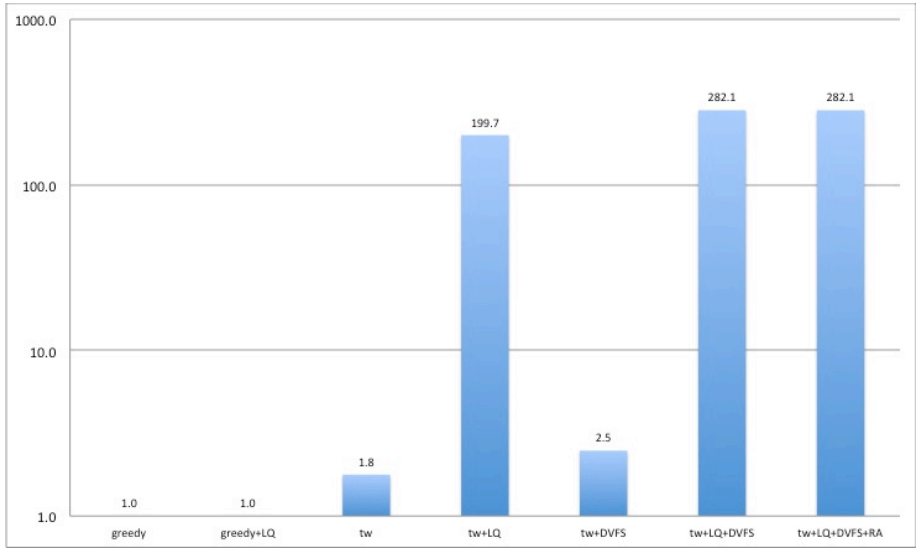
Standard Deviation for
10,000 Periods



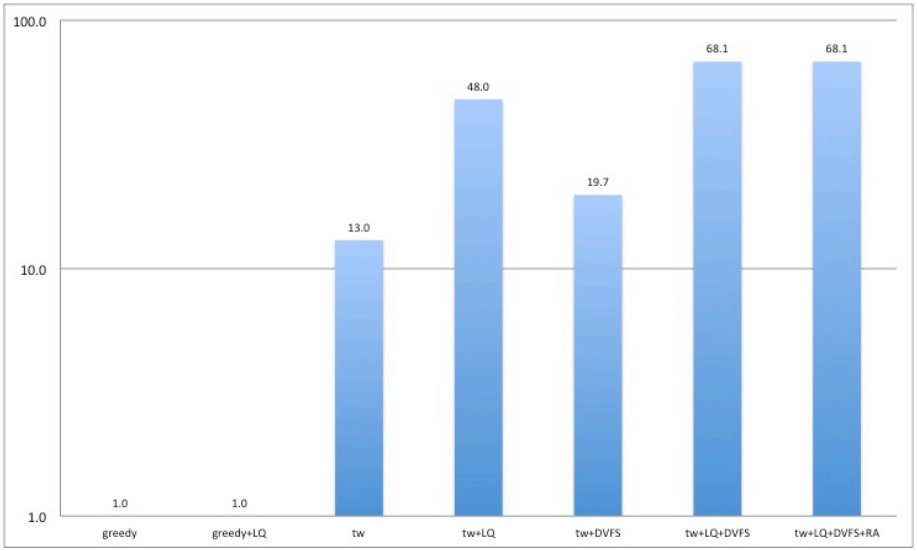
Normalized Improvement Factors (28nm Scenario)

Experimental Results and Analysis (2)

- **Improvement factors decrease due to more available resources, but same application scenario**



Normalized Improvement Factors (14nm Scenario)



Normalized Improvement Factors (7nm Scenario)

Conclusion and Future Work

- **Evaluated a number of dynamic runtime optimizations using simulation of three different heterogeneous task models**
- **Showed an improvement of 390x over a baseline greedy algorithm in the best case**
- **Greatest improvement demonstrated by Time-Window with local queues and DVFS adjustment**
- **Future work**
  - Testing on real hardware
  - Explore other application scenarios
  - More scheduling heuristics
  - Refine communication model
  - Computational node locality

Acknowledgements

- **TAPAS Group for valuable comments and FPGA data**
  - Professor Viktor Prasanna
  - Andrea Sanny
  - Yusong Hu
  - Ren Chen
  - Sanmukh Kuppannagari
  - Shreyas Singapura
  - Shijie Zhou
- **NVIDIA Team for GPU data**
  - Steve Keckler
  - Jason Clemons
- **DARPA for sponsoring this research**

Thank you!