Power and Energy Efficiency in Modern Processors

Aleksandar Milenković

November 30, 2021

Based on dissertation of Ranjan Hebbar,
VMware, Palo Alto, CA

Department of Electrical and Computer Engineering
University of Alabama in Huntsville

Overview

- Introduction
- Background
- Experimental Setup
- SPEC CPU2017 Characterization
- Proposed DVFS Techniques
- Results
- Conclusions

Introduction

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

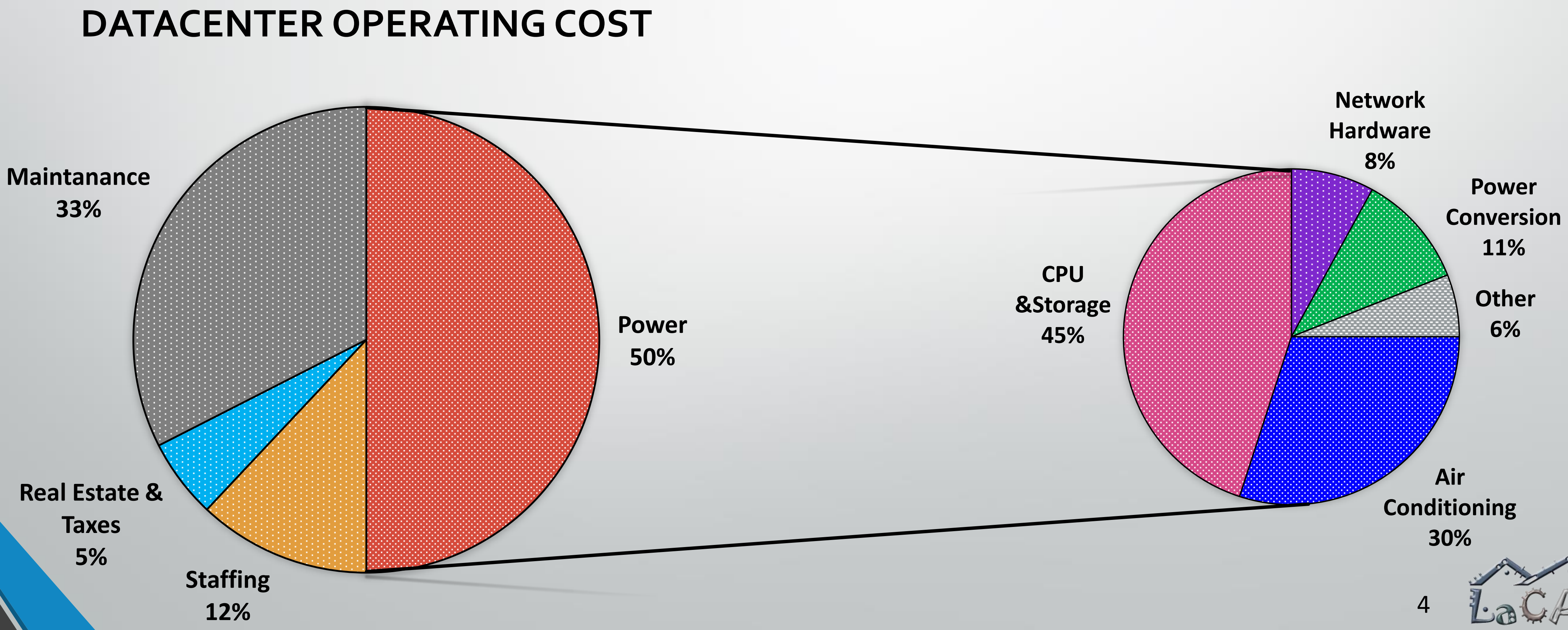
- Technology
  - Moore’s Law and Dennard’s Law
  - Processor advances: Caches, Out-of-Order, Vector Processing, Prefetching
- Applications
  - Increasing sophistication and complexity
  - Machine Learning, Block Chains, Information Security, Big Data
  - Software developers are not expected to be familiar with hardware internals
- Markets
  - Shortening time-to-market
  - Demand for high performance computing and data center capacity have grown exponentially

LaCASA

Typical Server Operating Cost Distribution

Energy consumption challenge

- Data centers in the US consume about 2% of the country’s total energy consumption (per US DoE)
- Forecast to reach ~8% by 2030 (Andrae and Edler, 2015)



State-of-the-art

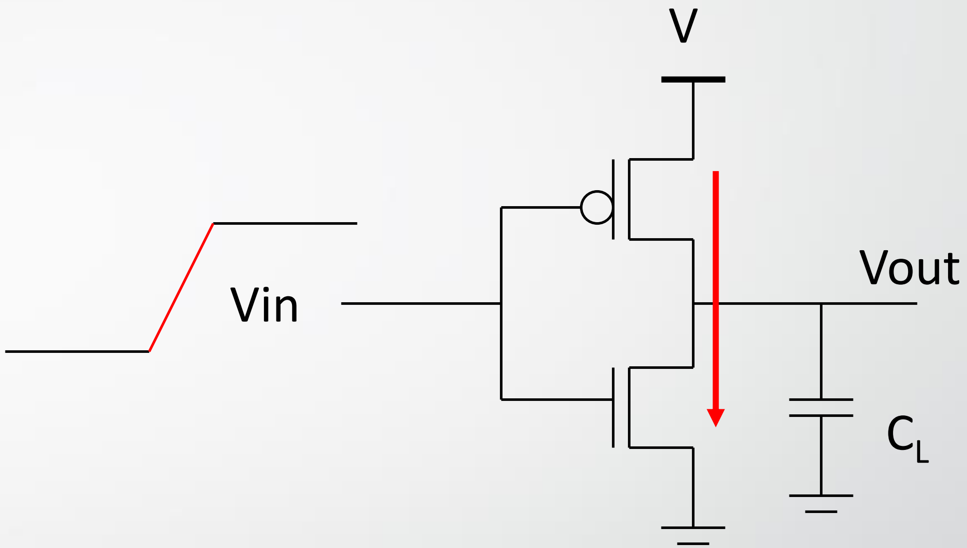
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- A majority of the servers in data centers currently utilize x86 processors
- x86 processors are complex structures integrating
  - Multiple physical cores
  - On-chip Interconnect
  - Uncore Cache
  - Memory Controllers
  - Hardware Accelerators
- Dedicated hardware resources for monitoring and management of its operating states
- DVFS: Most powerful technique in regulating CPU power consumption

LaCASA

Processor Power Dissipation

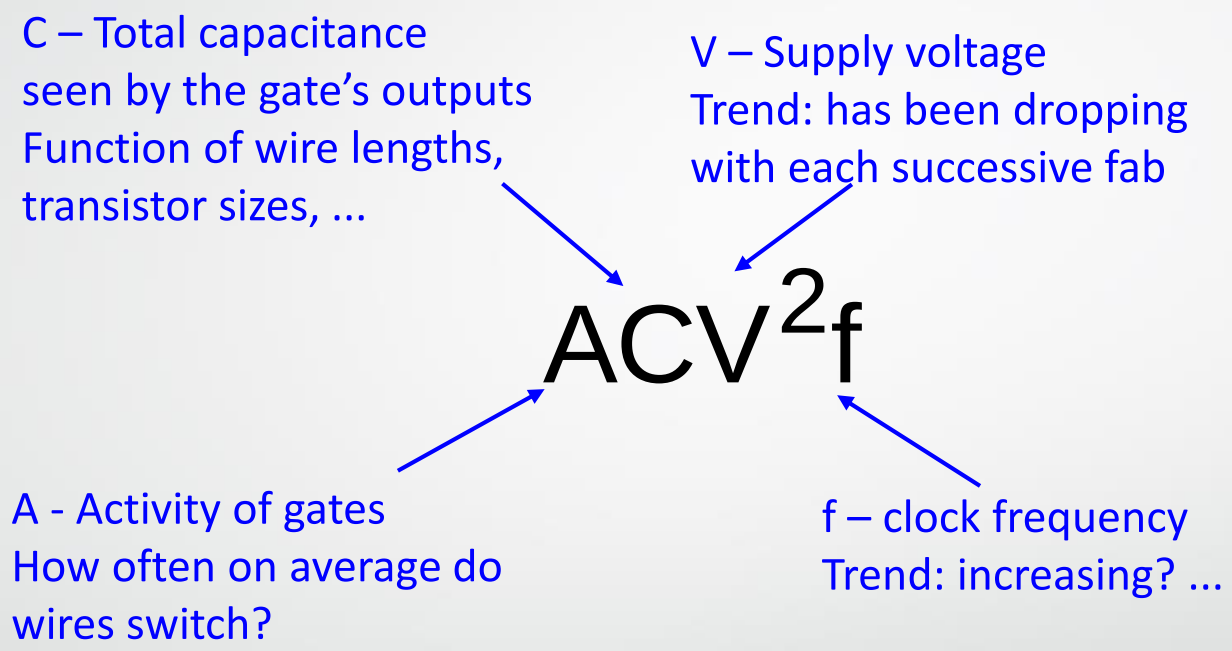
- Three factors
  - Dynamic power (switching)
  - Short-circuit power
  - Leakage power



- $P_{CPU} = P_{dyn} + P_{sc} + P_{leak}$

- $P_{CPU} = ACV^2f + A\tau I_{sc}Vf + VI_{leak}$

Dynamic Power



C – Total capacitance seen by the gate's outputs
Function of wire lengths, transistor sizes, ...

V – Supply voltage
Trend: has been dropping with each successive fab

$$ACV^2f$$

A - Activity of gates
How often on average do wires switch?

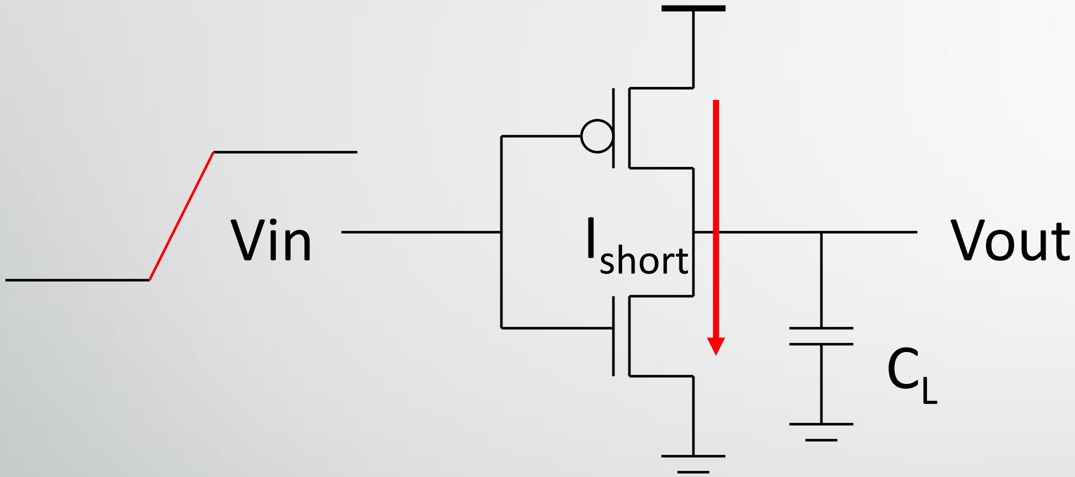
f – clock frequency
Trend: increasing? ...

Reducing Dynamic Power

1) Reducing V has quadratic effect; Limits?
2) Lower C - shrink structures, shorten wires
3) Reduce switching activity - Turn off unused parts or use design techniques to minimize number of transitions

Short-Circuit Power

$$\tau A V I_{short} f$$



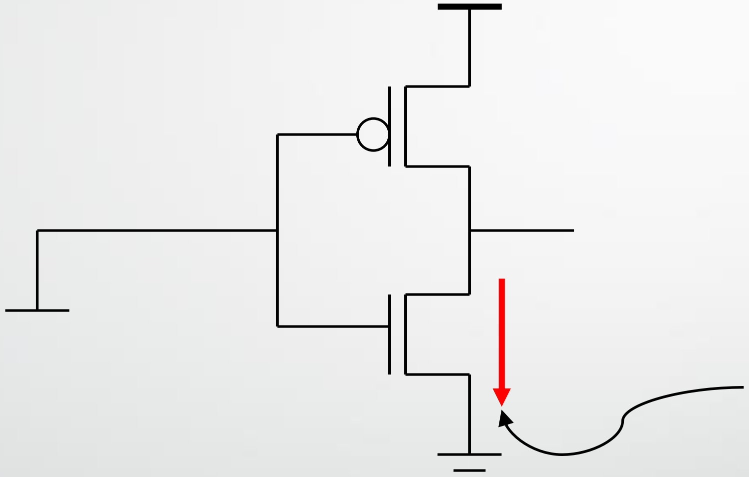
Finite slope of the input signal causes a direct current path between V_{DD} and GND for a short period of time during switching when both the NMOS and PMOS transistors are conducting

Reducing Short-circuit

1) Lower the supply voltage V
2) Slope engineering – match the rise/fall time of the input and output signals

Leakage Power

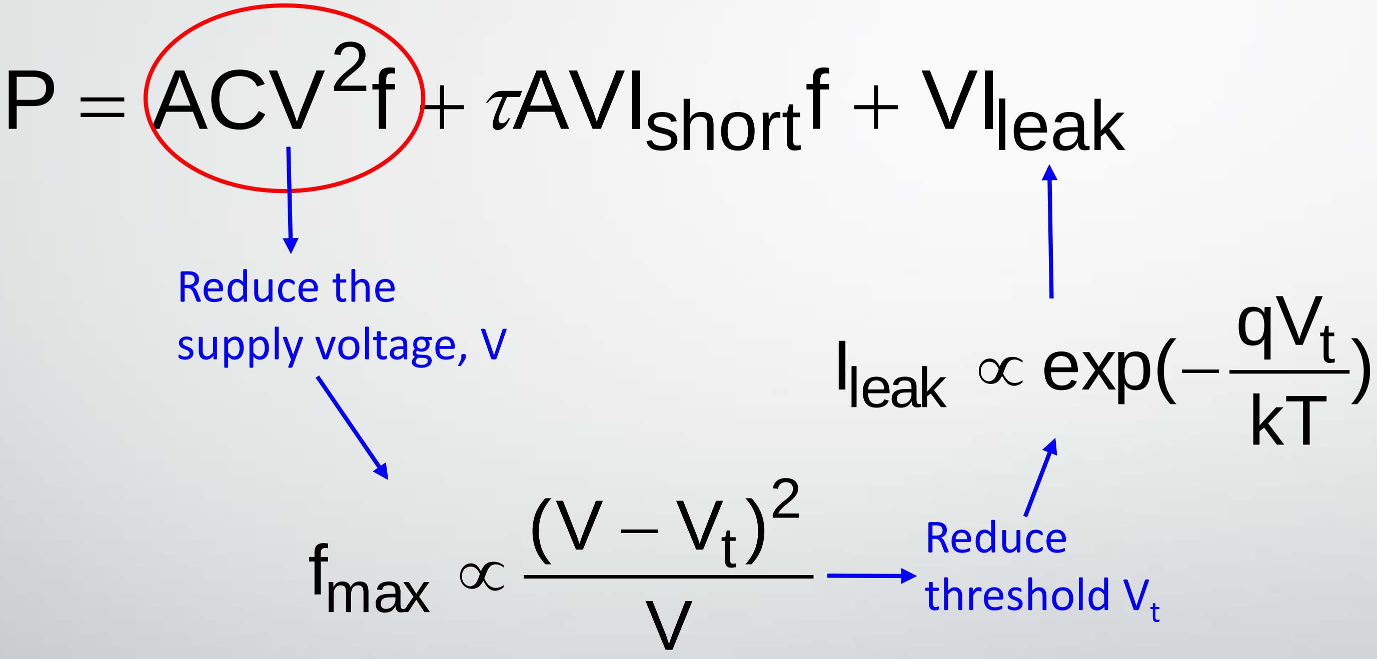
VI_{leak}



Sub-threshold current

Sub-threshold current grows exponentially with increases in temperature and decreases in Vt

CMOS Power Tradeoffs



$$P = ACV^2 f + \tau AVI_{short} f + VI_{leak}$$

Reduce the supply voltage, V

$$f_{max} \propto \frac{(V - V_t)^2}{V}$$

Reduce threshold V_t

$$I_{leak} \propto \exp\left(-\frac{qV_t}{kT}\right)$$

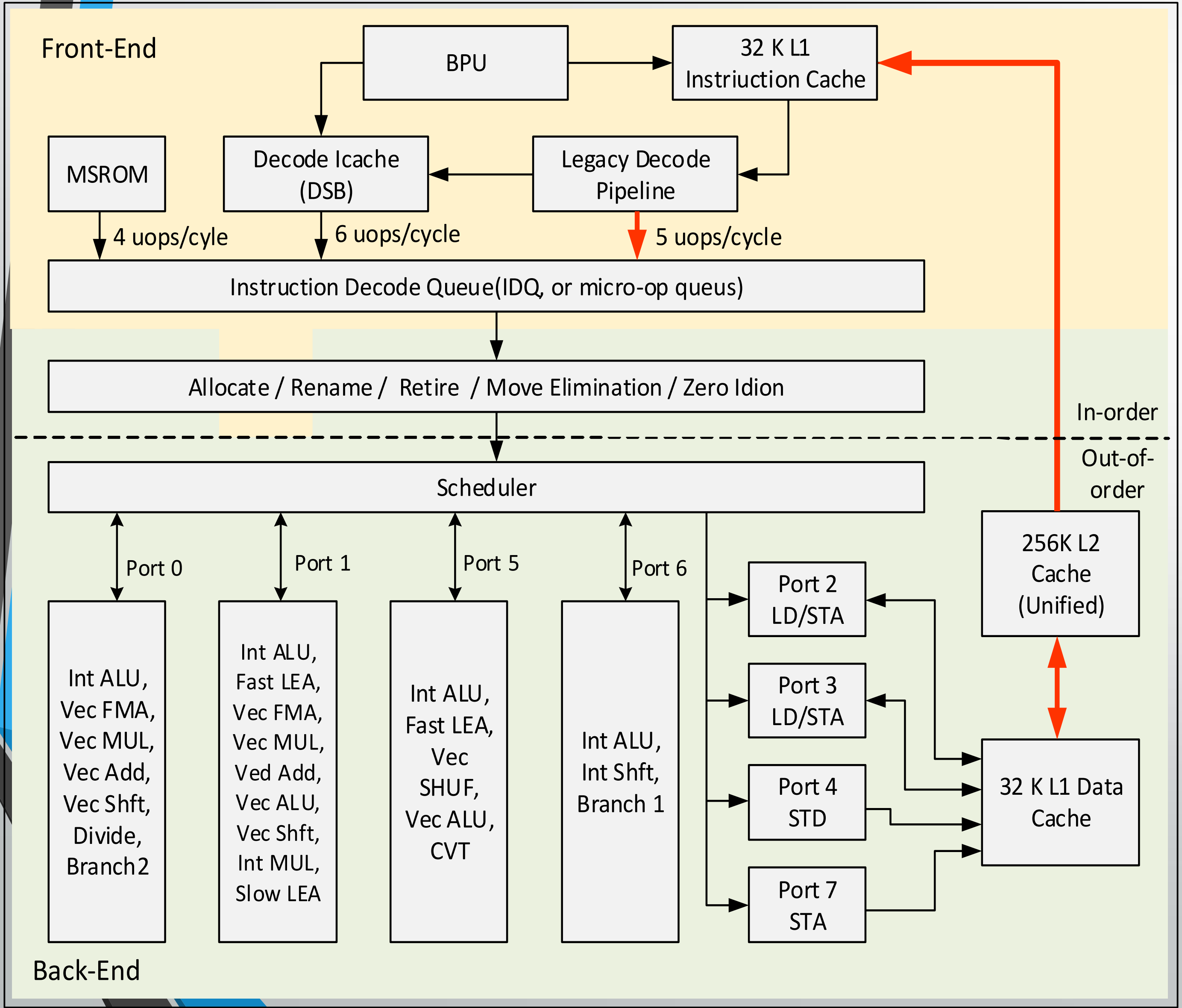
Overview

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Introduction
- Background
- Experimental Setup
- SPEC CPU2017 Characterization
- Proposed DVFS Techniques
- Results
- Conclusions

LaCASA

Intel Skylake Microarchitecture


Front-End
BPU
32 K L1 Instriuction Cache
MSROM
Decode Icache (DSB)
Legacy Decode Pipeline
4 uops/cyle
6 uops/cycle
5 uops/cycle
Instruction Decode Queue(IDQ, or micro-op queus)
Allocate / Rename / Retire / Move Elimination / Zero Idion
In-order
Out-of-order
Scheduler
Port 0
Port 1
Port 5
Port 6
Int ALU, Vec FMA, Vec MUL, Vec Add, Vec Shft, Divide, Branch2
Int ALU, Fast LEA, Vec FMA, Vec MUL, Ved Add, Vec ALU, Vec Shft, Int MUL, Slow LEA
Int ALU, Fast LEA, Vec SHUF, Vec ALU, CVT
Int ALU, Int Shft, Branch 1
Port 2 LD/STA
Port 3 LD/STA
Port 4 STD
Port 7 STA
256K L2 Cache (Unified)
32 K L1 Data Cache
Back-End

- 19-stage pipeline
- Superscalar & Multi-threaded

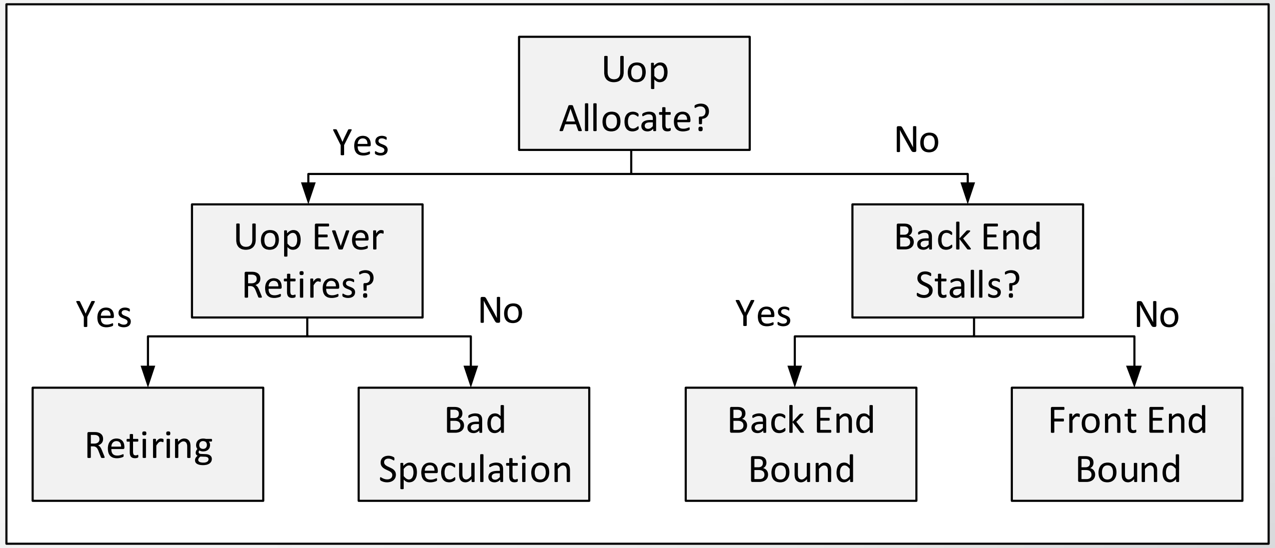
- Front-end (in-order)
  - Fetch from memory (L1 Instruction Cache)
  - Decode: Convert to micro-operation (uops)

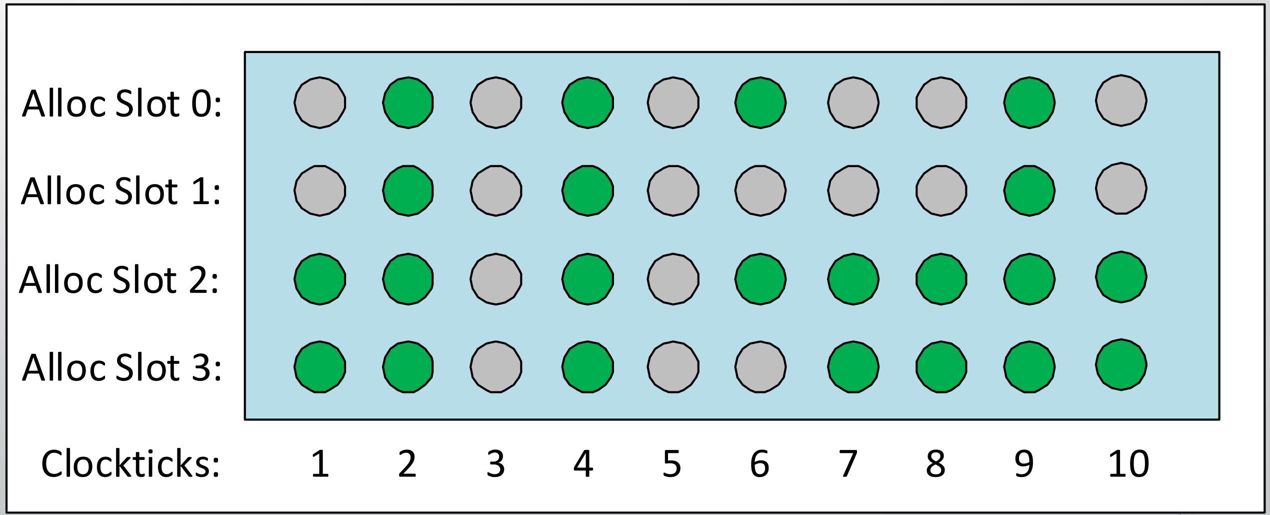
- Back-end (out-of-order)
  - Schedule micro-operations
  - Execute
  - Retire

Top-down Microarchitectural Analysis (TMAM)

Issue Stage of the Pipeline

- Each stalled pipeline slot can be attributed to one of three causes
  - Front-end Bound
  - Back-end Bound
  - Bad Speculation
- Ideally high retiring is preferred
- Example: 10 Clock cycles
  - 40 available pipeline slots (4-wide pipeline)
  - Micro-operations were issues for 22 slots
  - Pipeline Slot Utilization: (22/40) : **0.55**
  - Clock Cycle Utilization: (8/10) : **0.80**

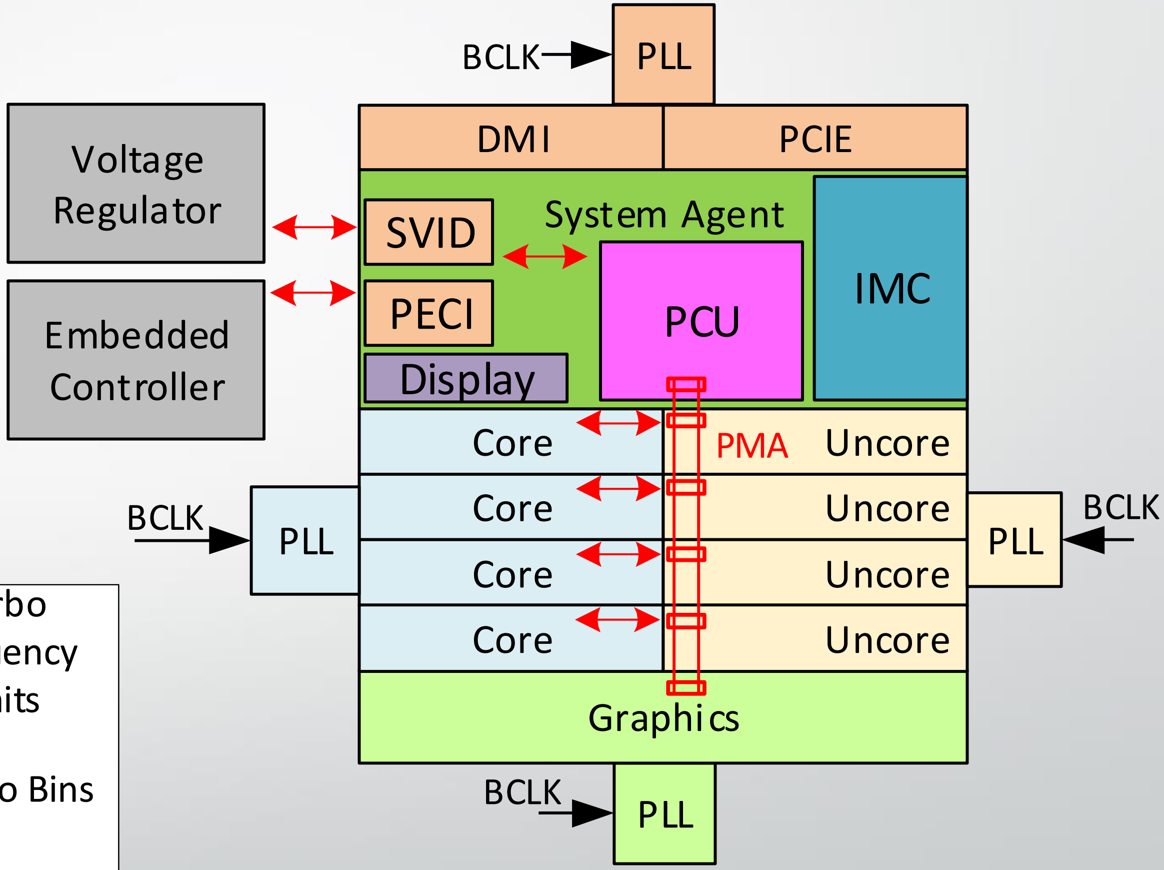


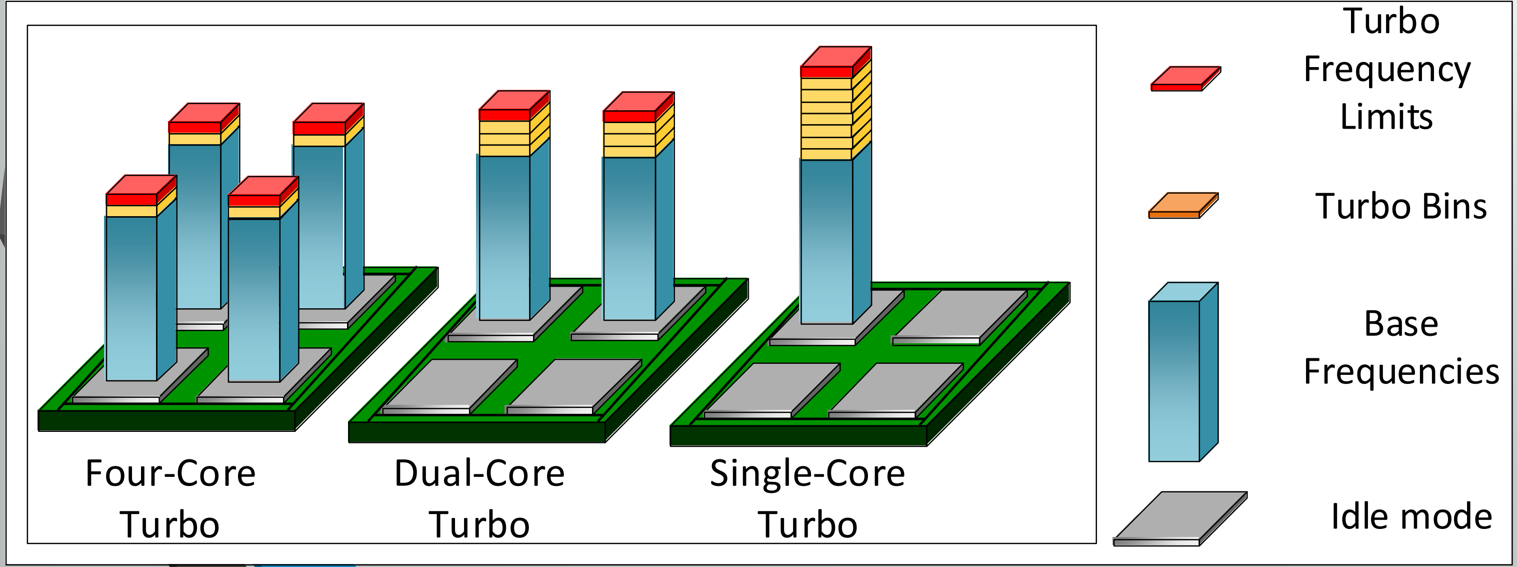


Alloc Slot 0:	⚪	🟢	⚪	🟢	⚪	🟢	⚪	⚪	🟢	⚪
Alloc Slot 1:	⚪	🟢	⚪	🟢	⚪	⚪	⚪	⚪	🟢	⚪
Alloc Slot 2:	🟢	🟢	⚪	🟢	⚪	🟢	🟢	🟢	🟢	🟢
Alloc Slot 3:	🟢	🟢	⚪	🟢	⚪	⚪	🟢	🟢	🟢	🟢
Clockticks:	1	2	3	4	5	6	7	8	9	10

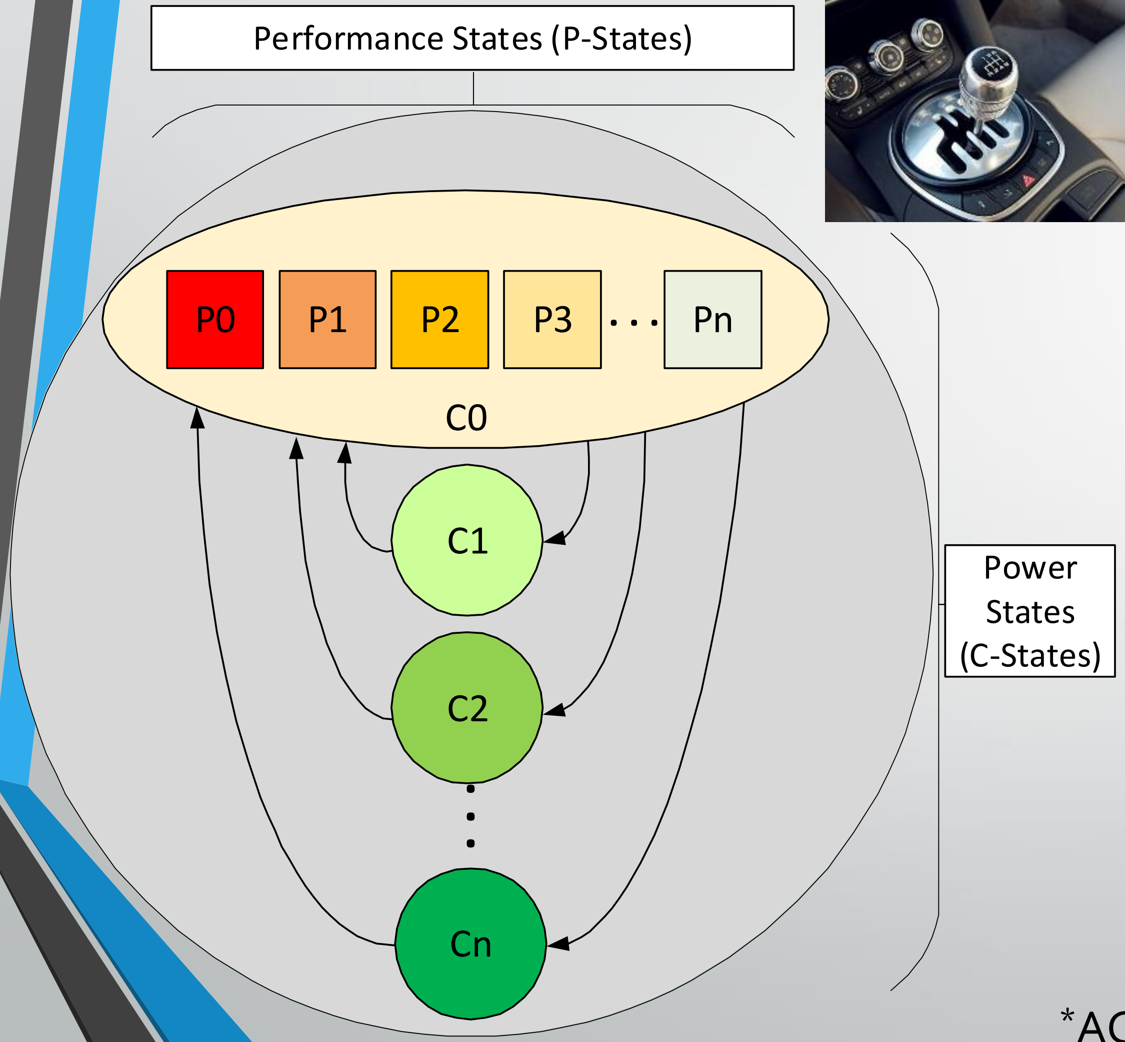
Hardware Power Management Features

- External Clock (BCLK): 133 MHz
- 4 Voltage/Power Domains
  - Core Domain
  - Uncore Domain
  - Graphics Domain
  - Fixed System Agent
- Thermal Design Power (TDP)





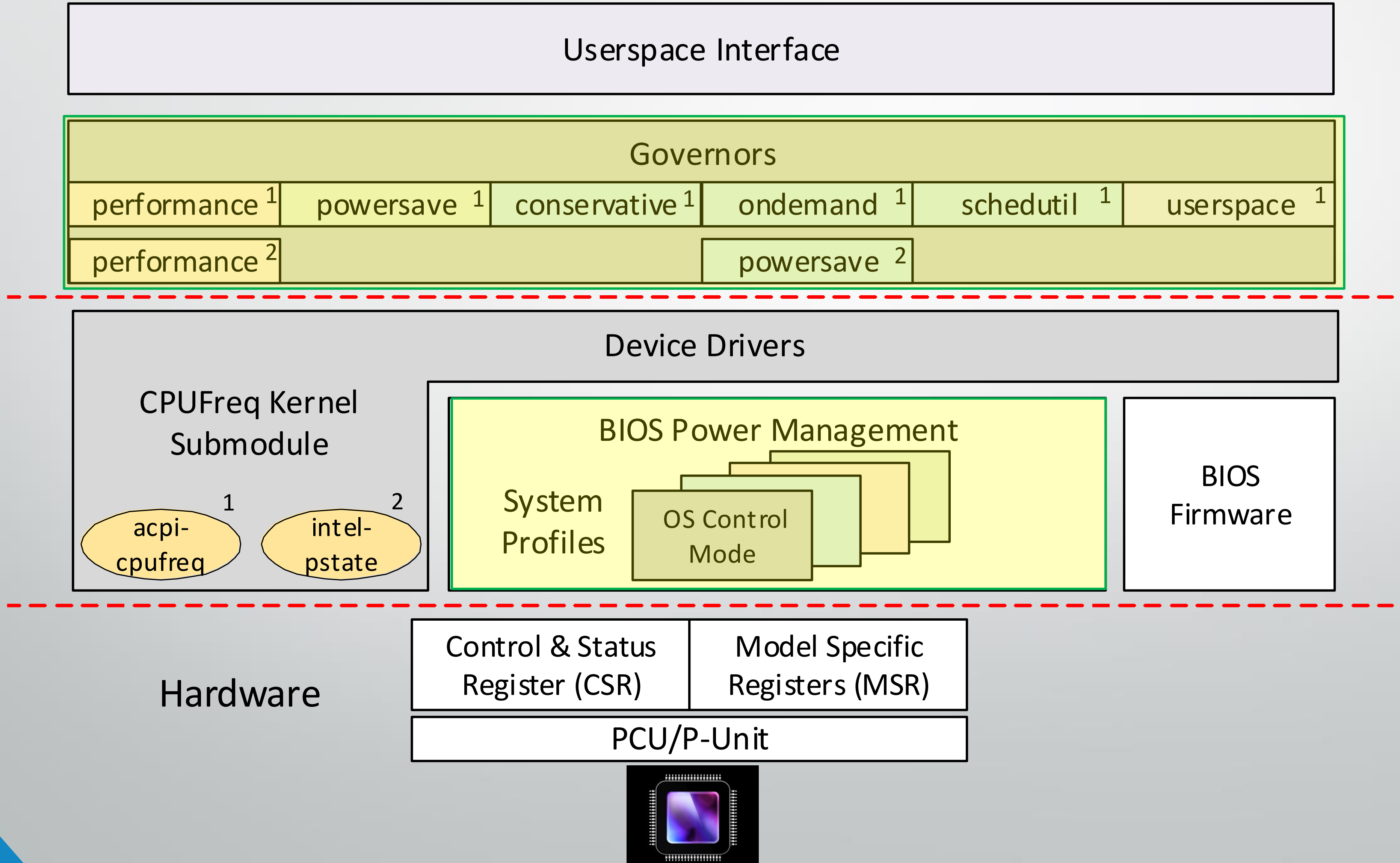
ACPI* Power & Performance States



- C-States: Power States or Idle States
  - C1 to Cn: No instructions are executed
  - Pros: turns off sections of the CPU ⇒ reduce idle power
  - Cons: introduces latency to get back to operational state
- P-States: Performance/Operational States (C0 Only)
  - Each P-State directs the processor to operate at a particular clock frequency and CPU voltage
  - P-States are the basis for Dynamic Voltage & Frequency Scaling (DVFS)
  - P-States is generally assigned based on CPU load

*ACPI – *Advanced Configuration Power Interface*

BIOS & OS Power Profiles



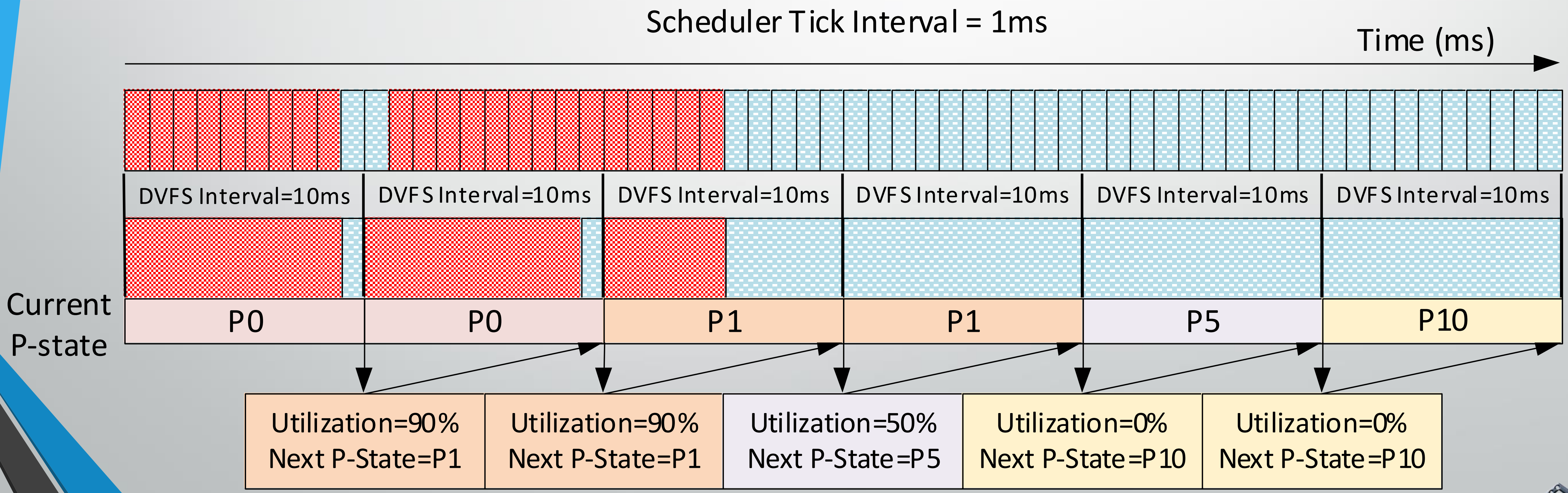
Utilization Based P-State Selection

***Ondemand* governor functioning**

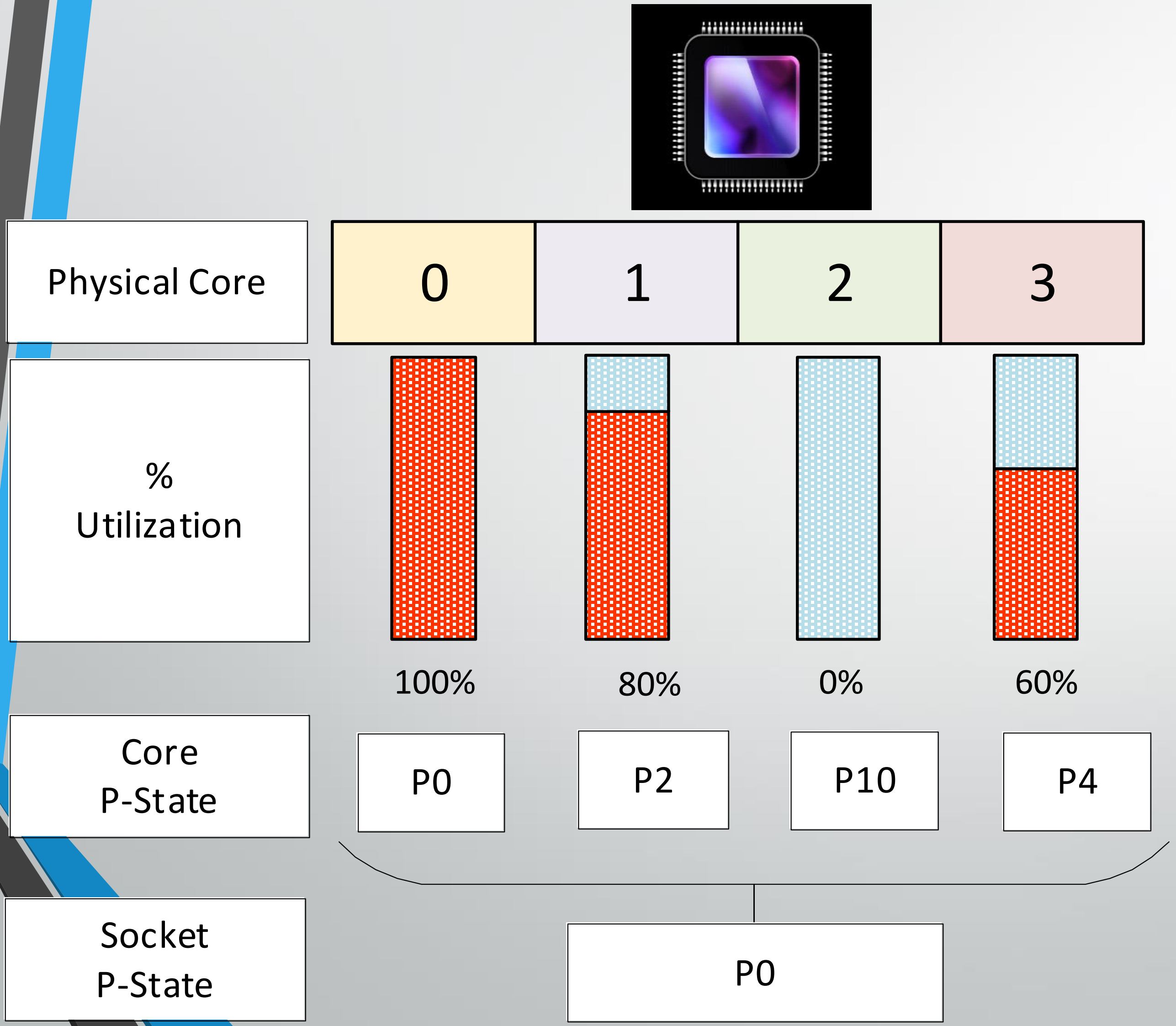
- Schedular provides CPU utilization metric

$$\%\ \text{CPU Utilization} = \frac{time\ in\ non\ idle\ thread}{time\ duration} * 100$$

- Scheduler update interval – 1 ms to 10 ms
  - Linux default scheduler tick is 1 ms
- DVFS Interval: 10ms to 100ms
  - Linux Default is 10ms


Scheduler Tick Interval = 1ms
Time (ms)
DVFS Interval=10ms
DVFS Interval=10ms
DVFS Interval=10ms
DVFS Interval=10ms
DVFS Interval=10ms
DVFS Interval=10ms
Current P-state
P0
P0
P1
P1
P5
P10
Utilization=90% Next P-State=P1
Utilization=90% Next P-State=P1
Utilization=50% Next P-State=P5
Utilization=0% Next P-State=P10
Utilization=0% Next P-State=P10

P-State Voting Mechanism



Physical Core	0	1	2	3
% Utilization	100%	80%	0%	60%
Core P-State	P0	P2	P10	P4
Socket P-State	P0			

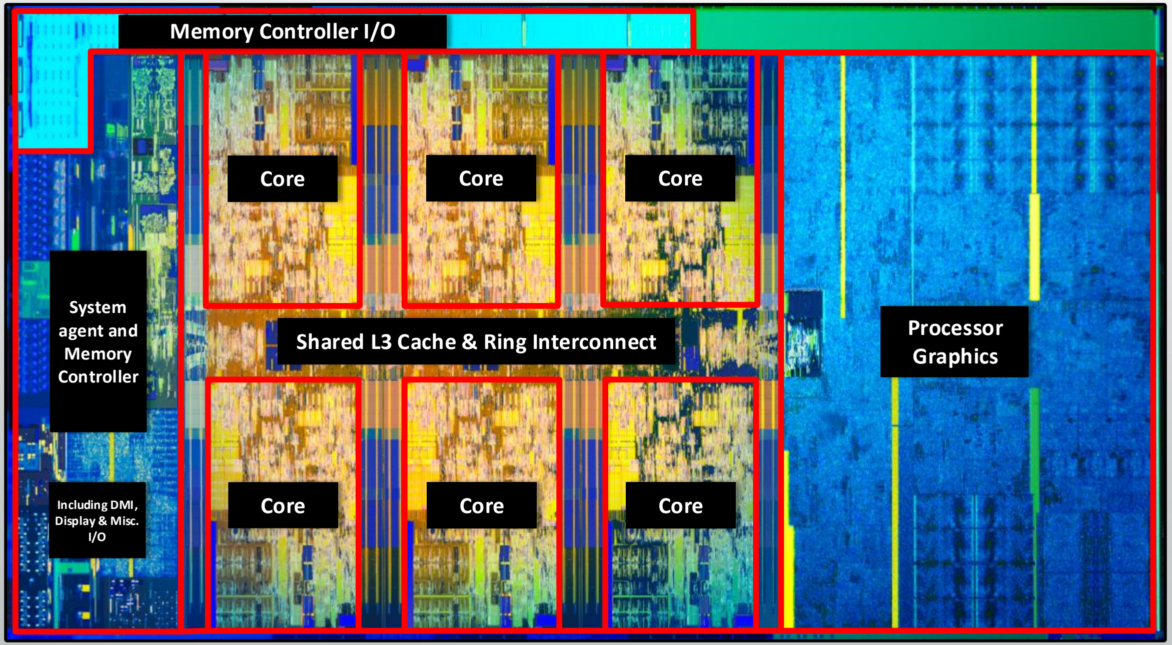
- P-State Management
  - Evaluate utilization for each core
  - Select P-state using linear mapping of CPU utilization to clock frequency
  - Lower P-state of logical cores is used to determine P-state of a physical core
- Processors with multiple voltage domains can have physical cores in different P-states
- Processor without multiple voltage domains select lowest P-state

Overview

- Introduction
- Background
- Experimental Setup
- SPEC CPU2017 Characterization
- Proposed DVFS Techniques
- Results
- Conclusions

Test Machine

- 8th Generation Intel Core i7-8700K (6 Physical Cores)
- 32 GiB of DRAM Memory: 2 Channels with a total bandwidth of 41.6 GB/s
- Coffee Lake, manufactured using Intel’s 14nm++ technology node
- Nominal clock frequency of 3.70 GHz and all core turbo of 4.30 GHz
- State-of-the-art P-state management (40 P-states)


Memory Controller I/O
Core
Core
Core
System agent and Memory Controller
Shared L3 Cache & Ring Interconnect
Processor Graphics
Including DMI, Display & Misc. I/O
Core
Core
Core

[source-https://en.wikichip.org/wiki/intel/core_i7/i7-8700k]

Metrics

- Performance Speedup

$$P.S\ (B_i, PG_{GOV}) = \frac{T\left(B_i, OD_{GOV}\right)}{T\left(B_i,\ PG_{GOV}\right)}$$

- Energy Efficiency Improvement

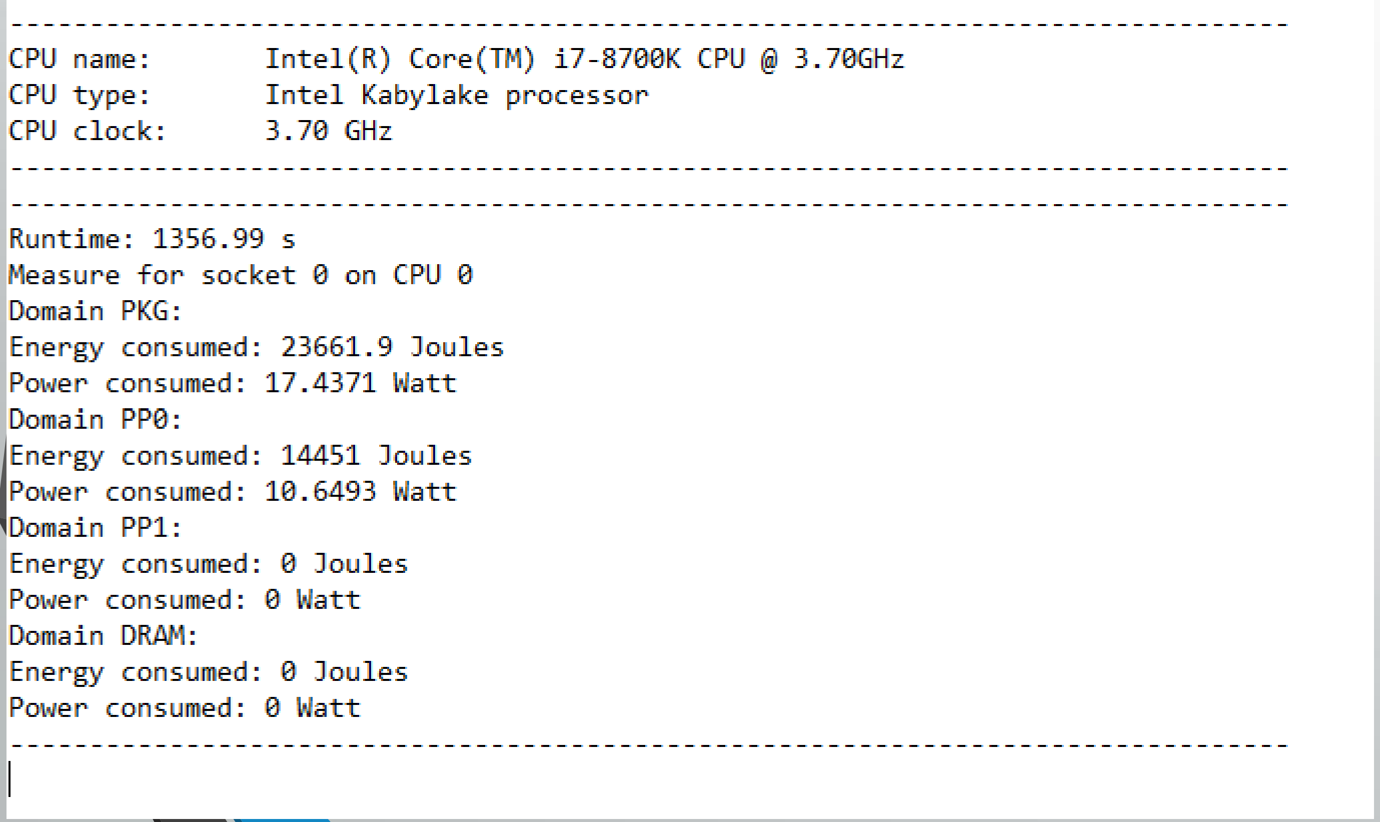
$$EE.I\ (Bi, PG_{GOV}) = \frac{E(B_i, OD_{GOV}\ )}{E(B_i, PG_{GOV})}$$

- PxEE Improvement

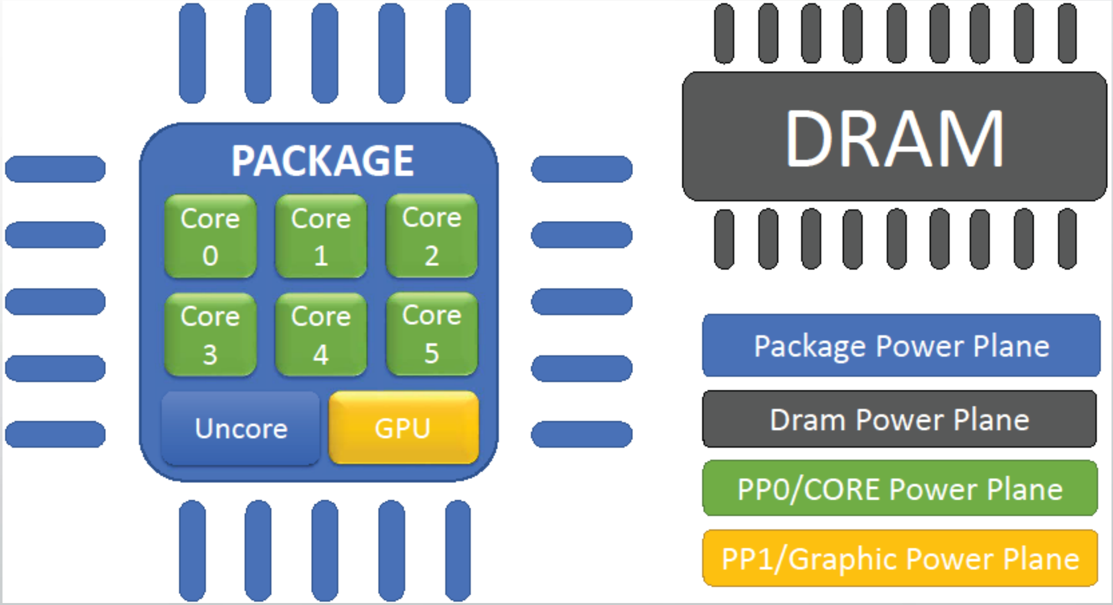
$$PxEE.I(B_i, PG_{GOV}) = \frac{T(B_i, OD_{GOV}) * E(B_i, OD_{GOV}\ )}{T(B_i, PG_{GOV}\ ) * E(B_i, PG_{GOV})}$$

Tools: likwid

- LIKWID (Like I Knew What I'm Doing): Has a set of tools with specific purposes to measure performance/energy groups
- Utilizes the RAPL counters for measuring power and energy



```
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
CPU type:       Intel Kabylake processor
CPU clock:      3.70 GHz
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Runtime: 1356.99 s
Measure for socket 0 on CPU 0
Domain PKG:
Energy consumed: 23661.9 Joules
Power consumed: 17.4371 Watt
Domain PP0:
Energy consumed: 14451 Joules
Power consumed: 10.6493 Watt
Domain PP1:
Energy consumed: 0 Joules
Power consumed: 0 Watt
Domain DRAM:
Energy consumed: 0 Joules
Power consumed: 0 Watt
--------------------------------------------------------------------------------
```



Workloads

- SPEC CPU2017
  - The SPEC CPU2017 benchmark suites contains standardized, CPU intensive suites for measuring and comparing performance
  - Stresses a system's processor, memory subsystem, and compiler
- SPECpower_2008ssj
  - SPECpower_ssj2008 is an industry-standard benchmark designed for experimental power and performance evaluation of server computers
  - Varying transactional load from 100% to active idle in steps of 10%

Overview

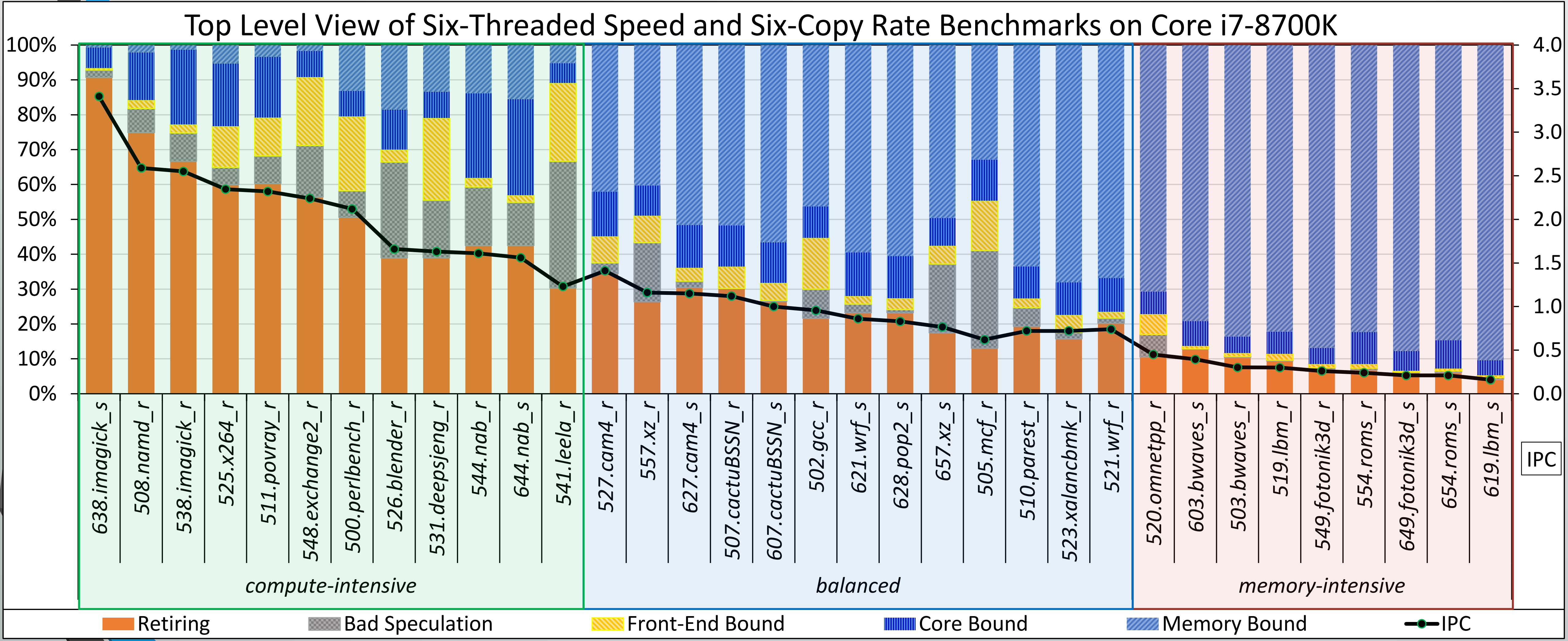
- Introduction
- Background
- Experimental Setup
- SPEC CPU2017 Characterization
- Proposed DVFS Techniques
- Results
- Conclusions

SPEC CPU2017 Benchmarks

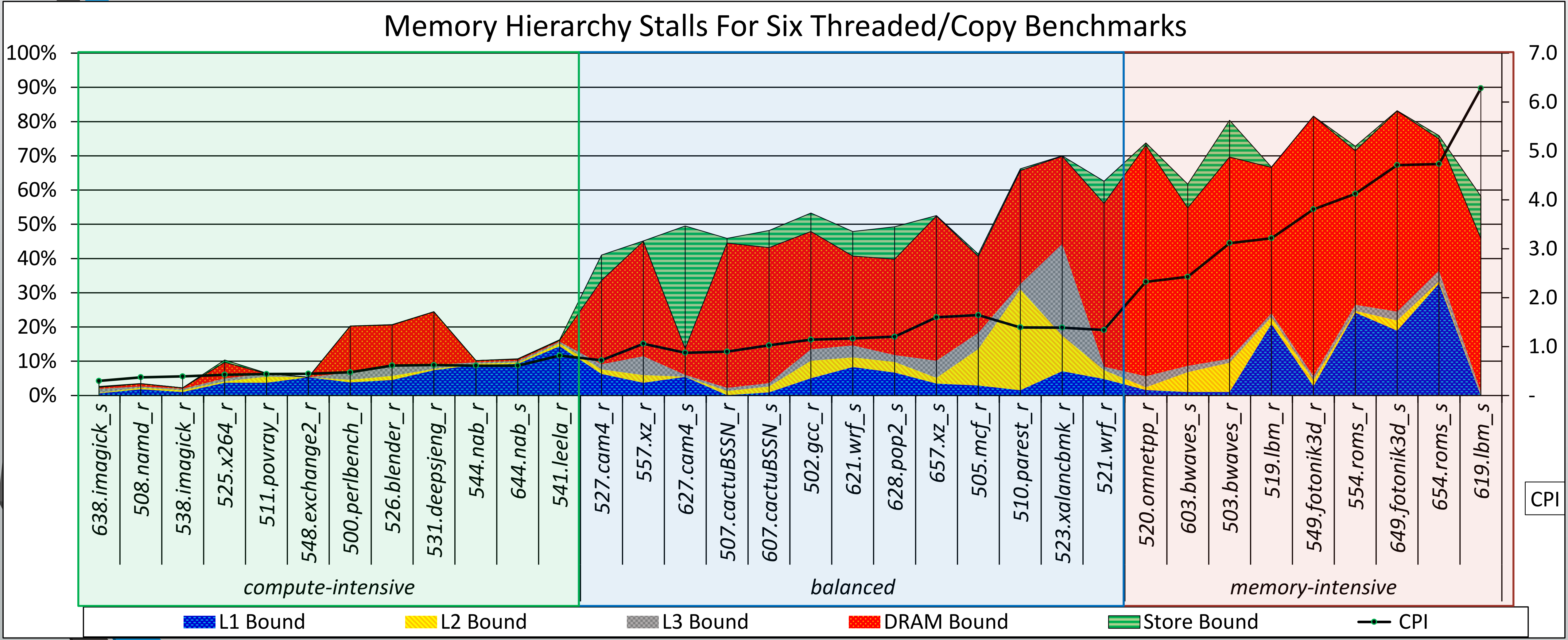
SPECrate 2017 Floating Point	SPECspeed 2017 Floating Point	Language	Application Area
503.bwaves_r	603.bwaves_s	Fortran	Explosion modeling
507.cactuBSSN_r	607.cactuBSSN_s	C++, C, Fortran	Physics: relativity
508.namd_r		C++	Molecular dynamics
510.parest_r		C++	Biomedical imaging: optical tomography
511.povray_r		C++, C	Ray tracing
519.lbm_r	619.lbm_s	C	Fluid dynamics
521.wrf_r	621.wrf_s	Fortran, C	Weather forecasting
526.blender_r		C++, C	3D rendering and animation
527.cam4_r	627.cam4_s	Fortran, C	Atmosphere modeling
	628.pop2_s	Fortran, C	Wide-scale ocean modeling (climate level)
538.imagick_r	638.imagick_s	C	Image manipulation
544.nab_r	644.nab_s	C	Molecular dynamics
549.fotonik3d_r	649.fotonik3d_s	Fortran	Computational Electromagnetics
554.roms_r	654.roms_s	Fortran	Regional ocean modeling

SPECrate 2017 Integer	SPECspeed 2017 Integer	Language	Application Area
500.perlbench_r	600.perlbench_s	C	Perl interpreter
502.gcc_r	602.gcc_s	C	GNU C compiler
505.mcf_r	605.mcf_s	C	Route planning
520.omnetpp_r	620.omnetpp_s	C++	Discrete Event simulation - computer network
523.xalancbmk_r	623.xalancbmk_s	C++	XML to HTML conversion via XSLT
525.x264_r	625.x264_s	C	Video compression
531.deepsjeng_r	631.deepsjeng_s	C++	Artificial Intelligence: alpha-beta tree search (Chess)
541.leela_r	641.leela_s	C++	Artificial Intelligence: Monte Carlo tree search (Go)
548.exchange2_r	648.exchange2_s	Fortran	Artificial Intelligence: recursive solution generator (Sudoku)
557.xz_r	657.xz_s	C	General data compression

Top-Down Analysis of SPEC CPU2017 Benchmarks

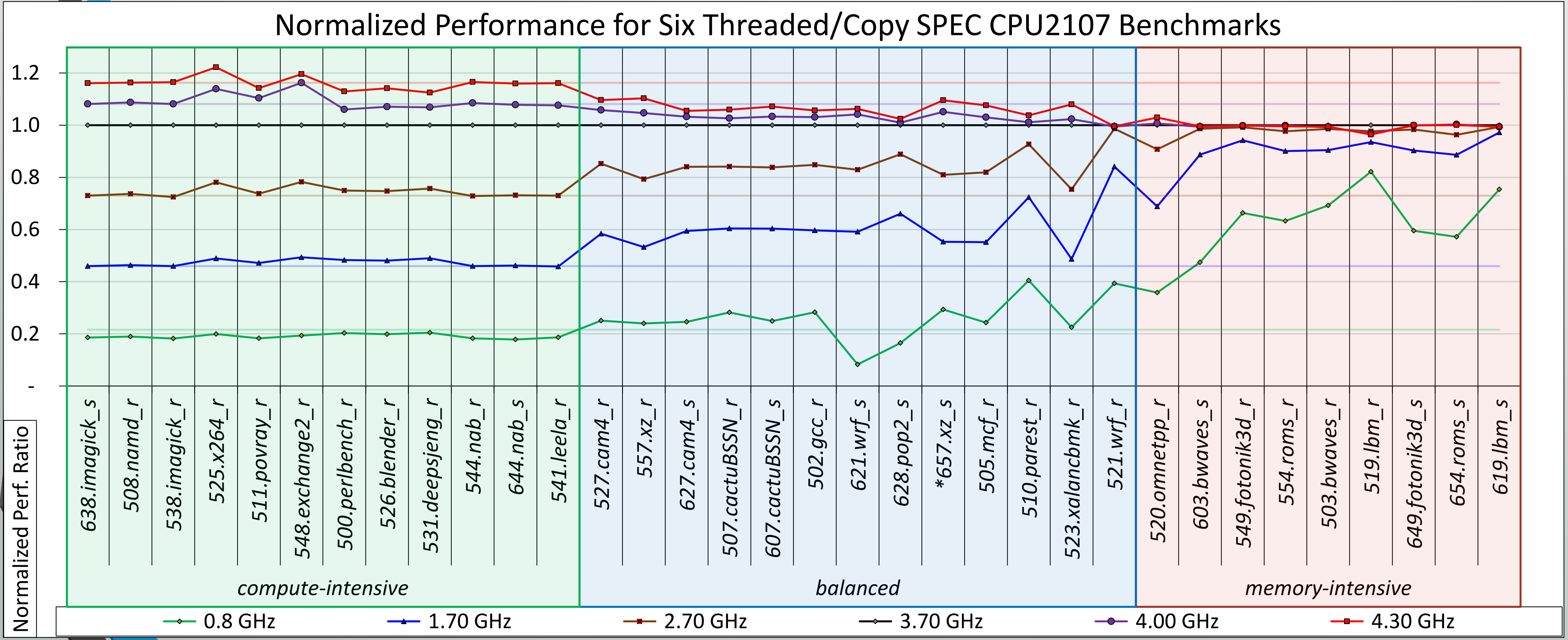


Memory Hierarchy Related Cycle Stall Breakdown



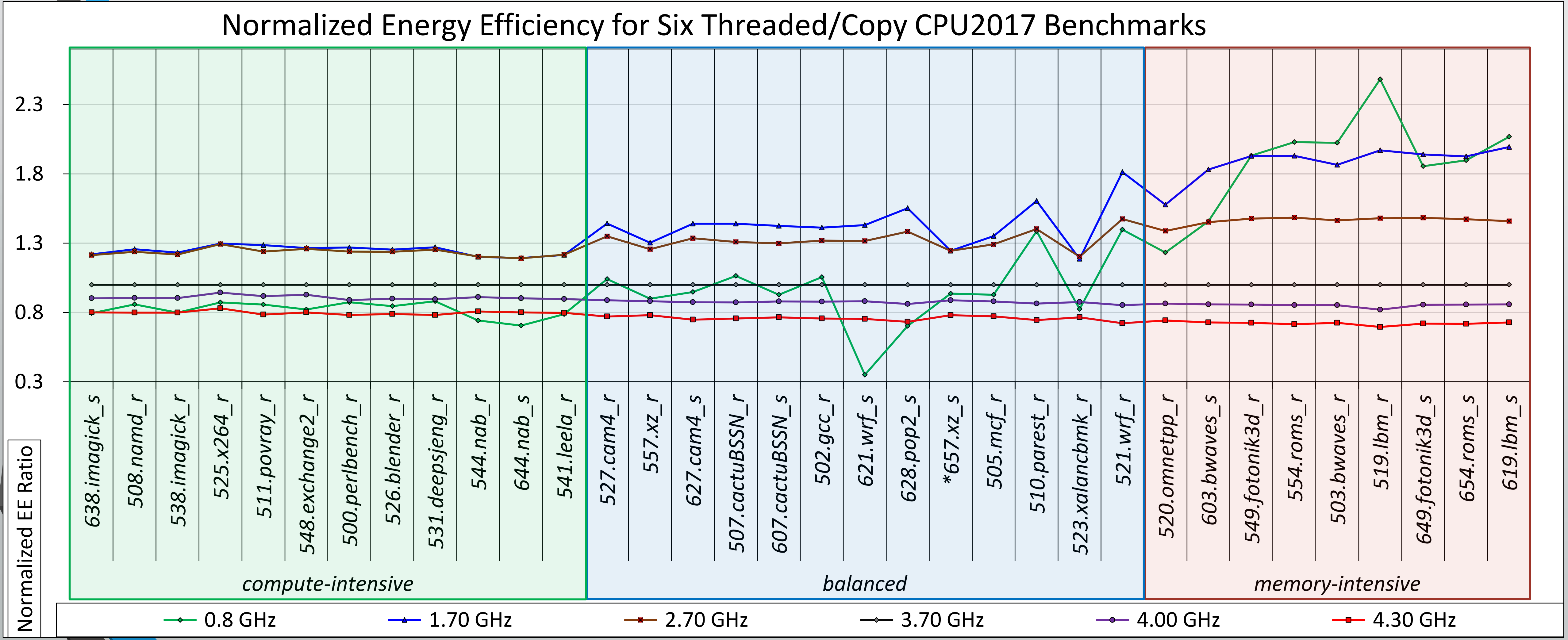
Normalized P for SPEC CPU2017 as a Function of Clock Frequency

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

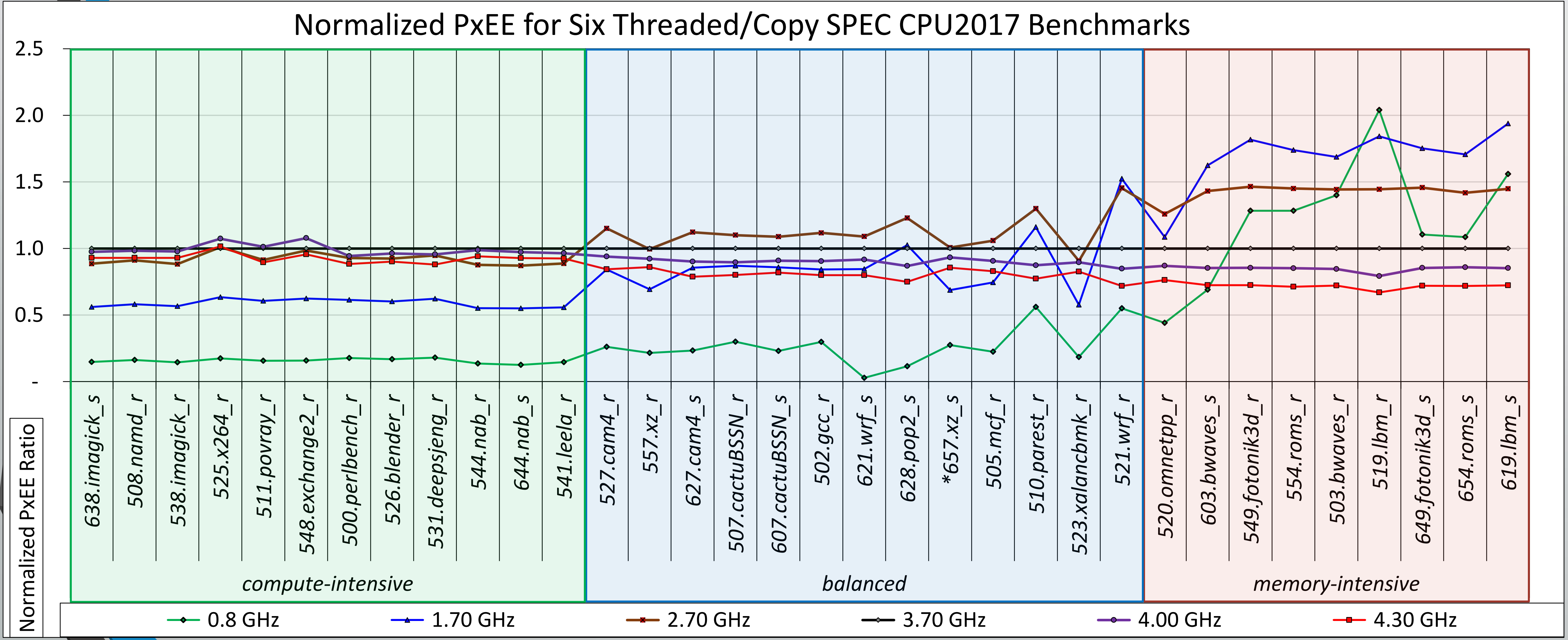


LaCASA

Normalized EE for SPEC CPU2017 as a Function of Clock Frequency

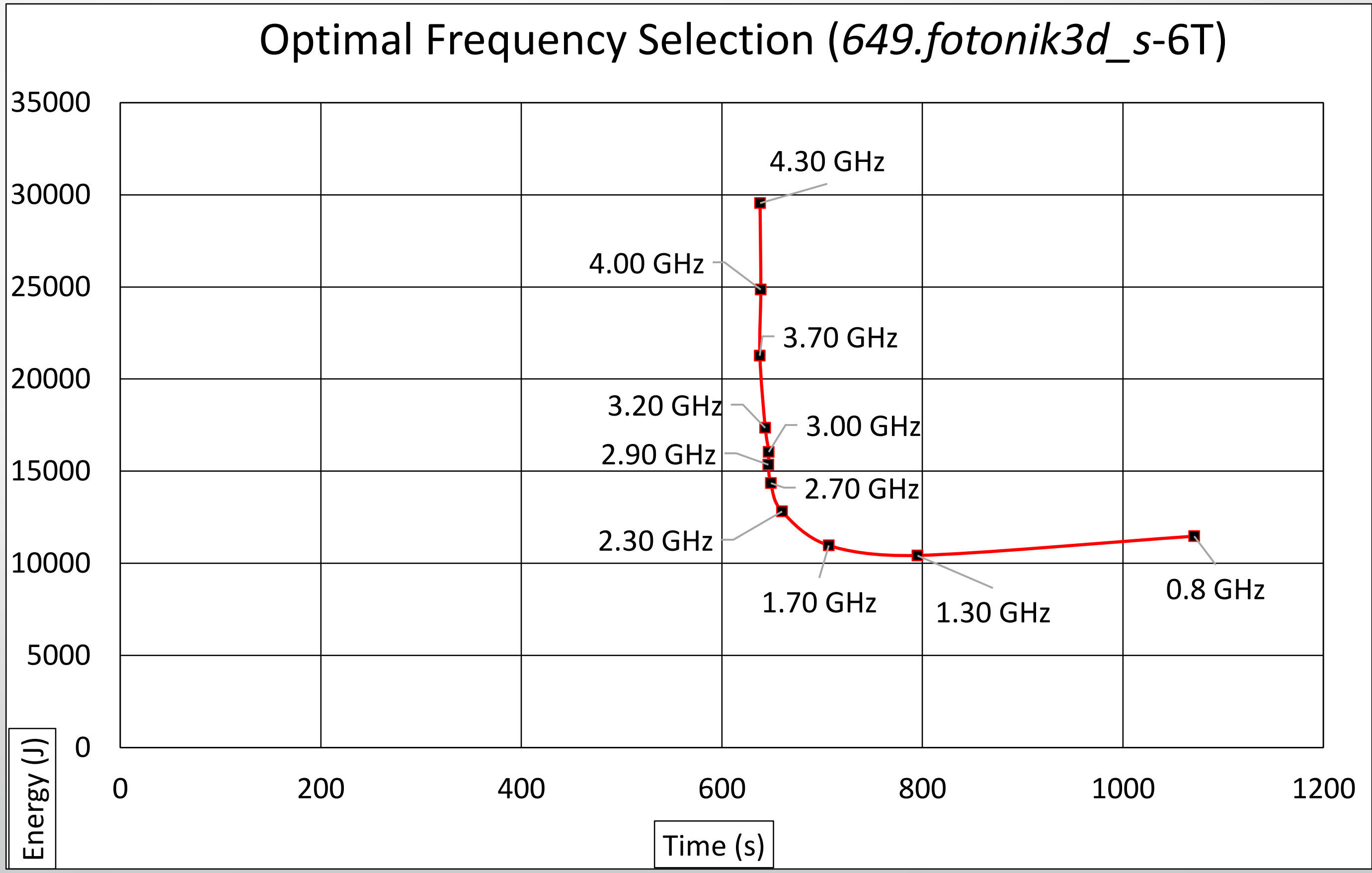


Normalized PxEE for SPEC CPU2017as a Function of Clock Frequency



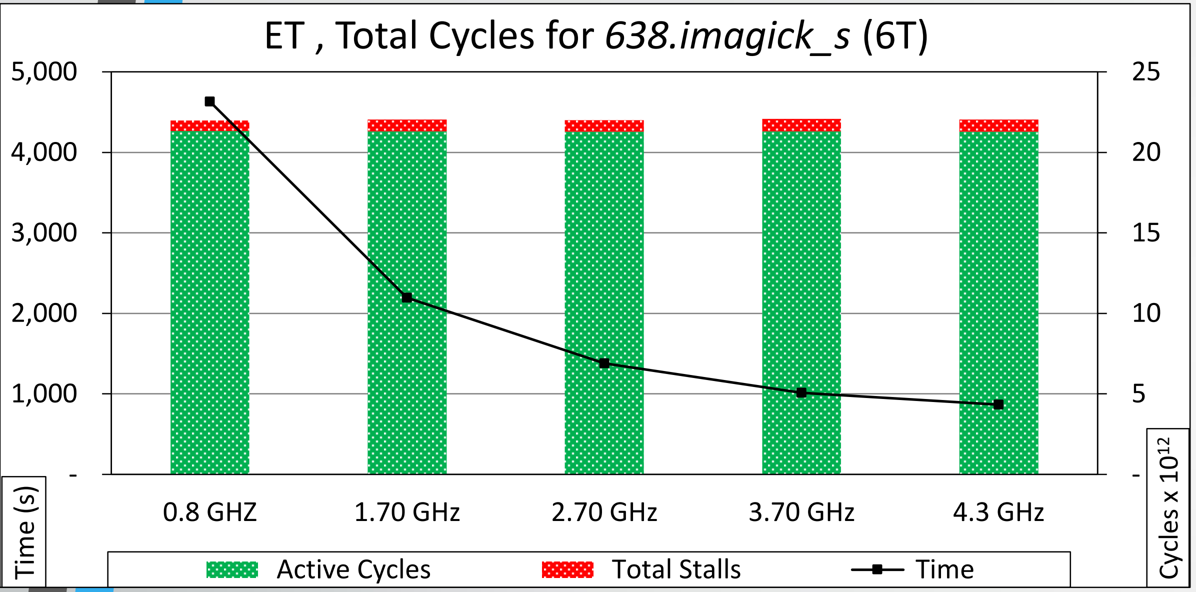
Optimal Frequency Selection

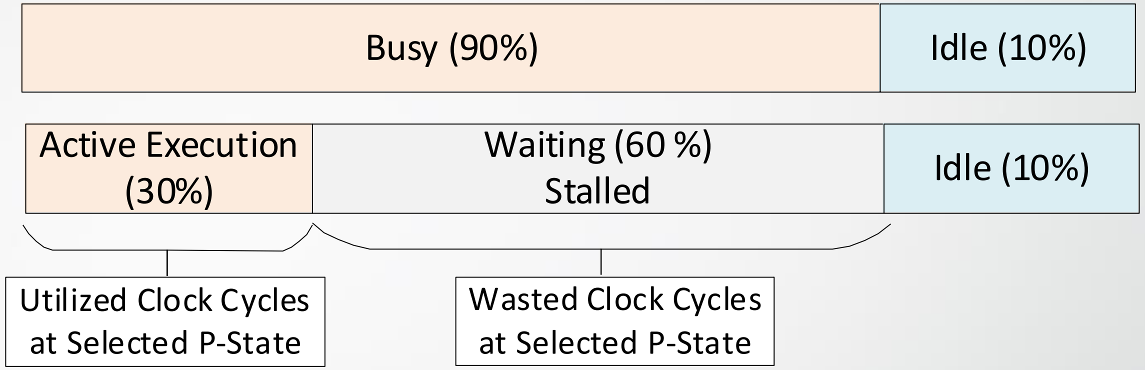
- Performance
  - 4.30 GHz to 2.70 GHz
- Energy Efficiency
  - 1.70 GHz to 1.30 GHz
- PxEE
  - 2.70 GHz to 1.70 GHz



Motivation: Limitations of CPU Utilization Metric

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

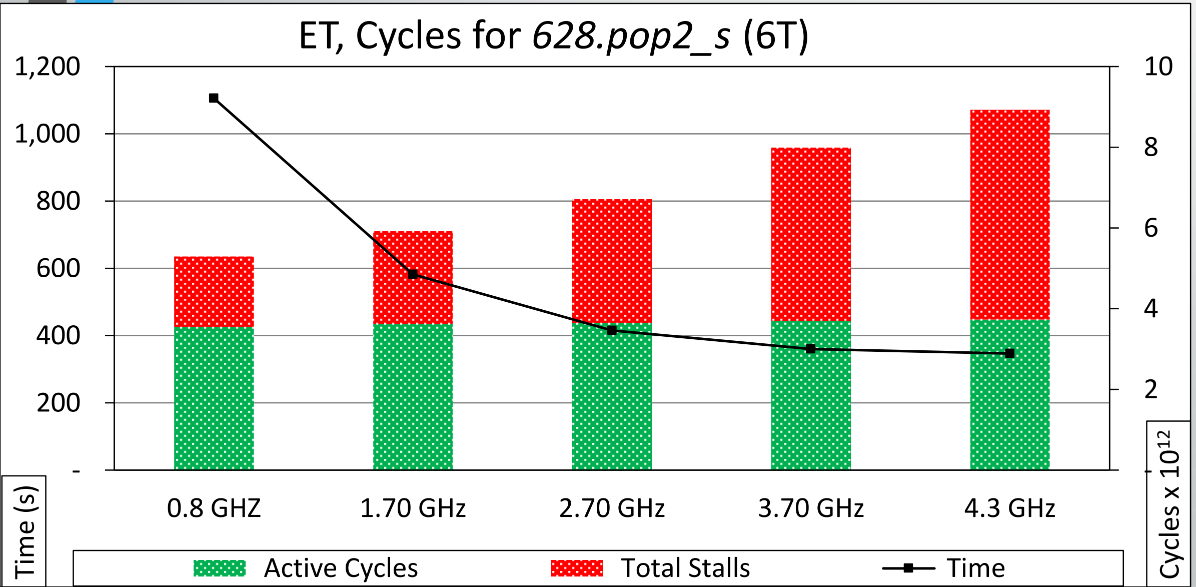

ET , Total Cycles for *638.imagick_s* (6T)

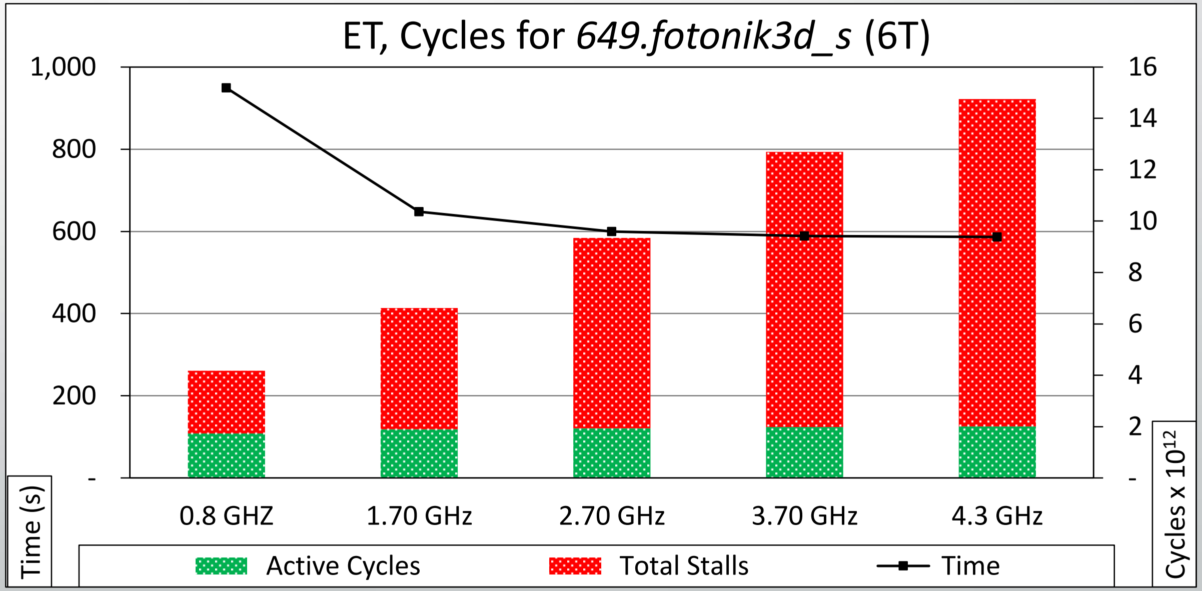

Busy (90%) | Idle (10%)

Active Execution (30%) | Waiting (60 %) Stalled | Idle (10%)

Utilized Clock Cycles at Selected P-State

Wasted Clock Cycles at Selected P-State


ET, Cycles for *628.pop2_s* (6T)


ET, Cycles for *649.fotonik3d_s* (6T)

LaCASA

Overview

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

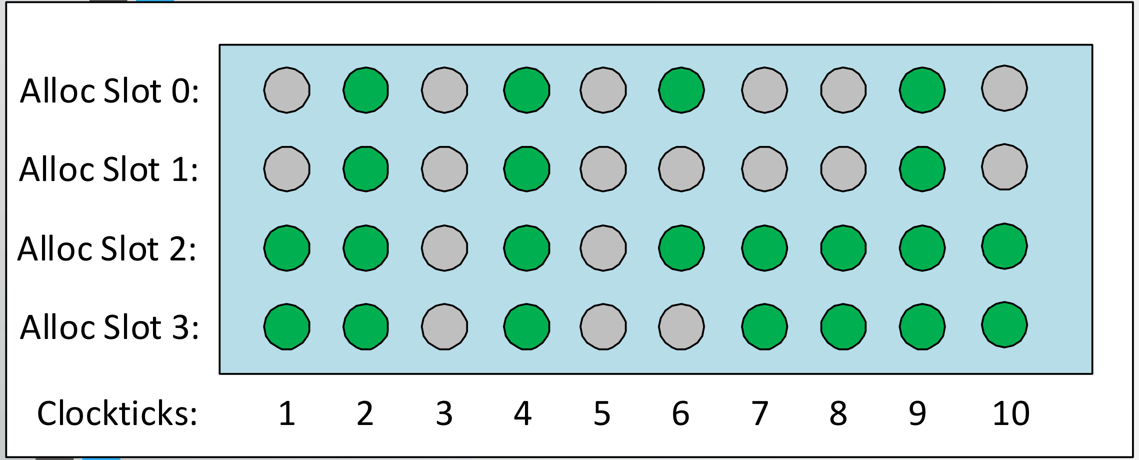
- Introduction
- Background & Motivation
- Experimental Setup
- SPEC CPU2017 Characterization
- Proposed DVFS Techniques
- Results
- Future Work & Conclusions

LaCASA

Proposed PMU-Event-Driven DVFS Techniques

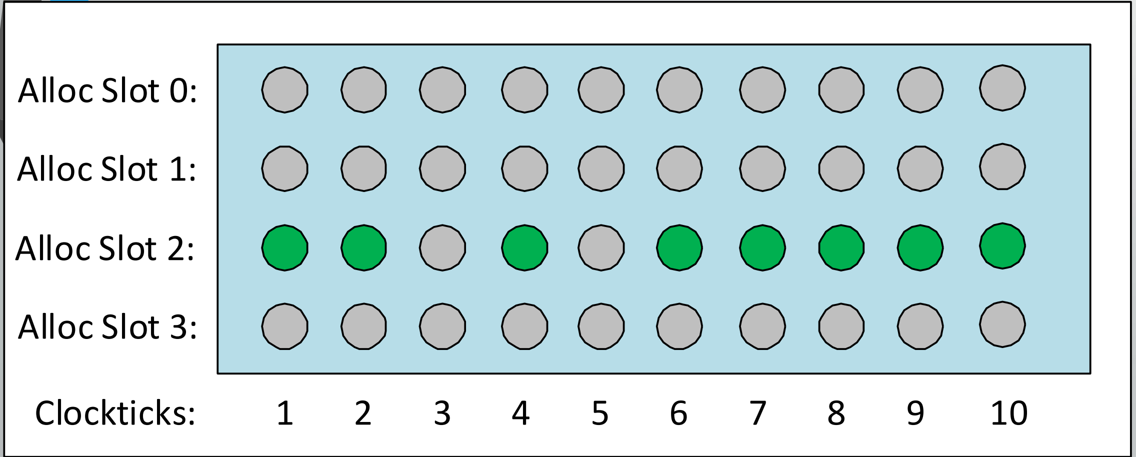
1. Frequency Selection Based on Pipeline Slot Stalls: **FS-PS**
2. Frequency Selection Based on Total Cycle Stalls: **FS-TS**
3. Frequency Selection Based on Memory Related Cycle Stalls: **FS-MS**
4. Frequency Selection Based on LLC- MPKI: **FS-LLCM**

P-State Selection Based on Pipeline Slot Stalls



Alloc Slot 0:	○	●	○	●	○	●	○	○	●	○
Alloc Slot 1:	○	●	○	●	○	○	○	○	●	○
Alloc Slot 2:	●	●	○	●	○	●	●	●	●	●
Alloc Slot 3:	●	●	○	●	○	○	●	●	●	●
Clockticks:	1	2	3	4	5	6	7	8	9	10

Total Available Slots → Issued Pipeline Slots + Unused Pipeline Slots



Alloc Slot 0:	○	○	○	○	○	○	○	○	○	○
Alloc Slot 1:	○	○	○	○	○	○	○	○	○	○
Alloc Slot 2:	●	●	○	●	○	●	●	●	●	●
Alloc Slot 3:	○	○	○	○	○	○	○	○	○	○
Clockticks:	1	2	3	4	5	6	7	8	9	10

Frequency Selection based on Pipeline Slot Stalls (FS-PS)

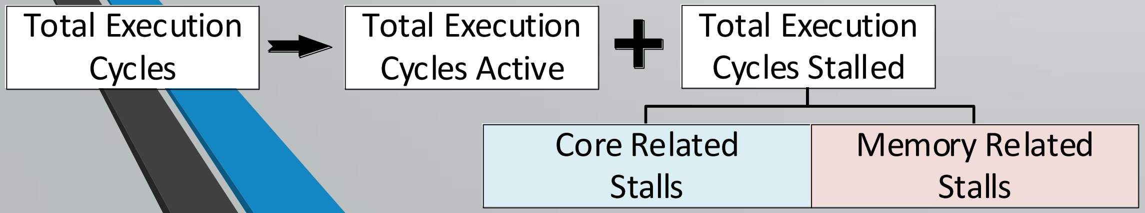
- Example 1: 10 Clock cycles: 40 pipeline slots
  - 40 available pipeline slots (4-wide pipeline)
  - Micro-operations were issues for 22 slots
  - Pipeline Stall ratio: **0.45** ⇒ **P5-State**

- Example 2: 10 Clock cycles
  - 40 available pipeline slots (4-wide pipeline)
  - Micro-operations were issued for 8 slots
  - Pipeline Stall ratio: **0.8 (32/40)** ⇒ **P8-State**

P-State Selection Based on Cycle Stalls

Alloc Slot 0:	○	●	○	●	○	●	○	○	●	○
Alloc Slot 1:	○	●	○	●	○	○	○	○	●	○
Alloc Slot 2:	●	●	○	●	○	●	●	●	●	●
Alloc Slot 3:	●	●	○	●	○	○	●	●	●	●
Clockticks:	1	2	3	4	5	6	7	8	9	10

Alloc Slot 0:	○	○	○	○	○	○	○	○	○	○
Alloc Slot 1:	○	○	○	○	○	○	○	○	○	○
Alloc Slot 2:	●	●	○	●	○	●	●	●	●	●
Alloc Slot 3:	○	○	○	○	○	○	○	○	○	○
Clockticks:	1	2	3	4	5	6	7	8	9	10



- Example1 & 2: 10 Clock cycles
  - 10-clock cycles
  - Total Cycle Stalls: 2
  - Memory-related Cycle Stalls: 1

Frequency Selection based on Total Cycle Stalls (FS-TS)

- Total Cycle Stall ratio: 0.2 ⇒ **P2-State**

Frequency Selection based on Memory-Related Cycle Stalls (FS-MS)

- Memory-related Cycle Stall ratio: 0.1 ⇒ **P1-State**

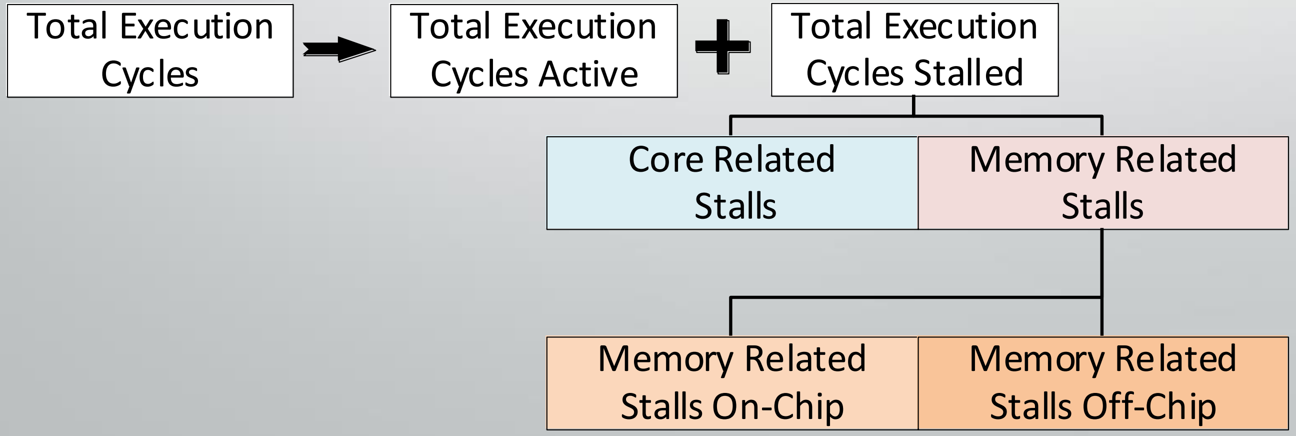
P-State Selection Based on LLC Misses PKI

- Targeted DVFS for applications with significant Off-chip requests
- LLC miss denotes off-chip requests
- An LLC miss is followed by stalls associated by data fetch from main-memory

Frequency Selection based on LLC Miss PKI (FS-LLCM)

- Map LLC-MPKI to available P-states

- Practical Considerations
  - Range of MPKI observed on real life workload shows a range of 0-100 MPKI

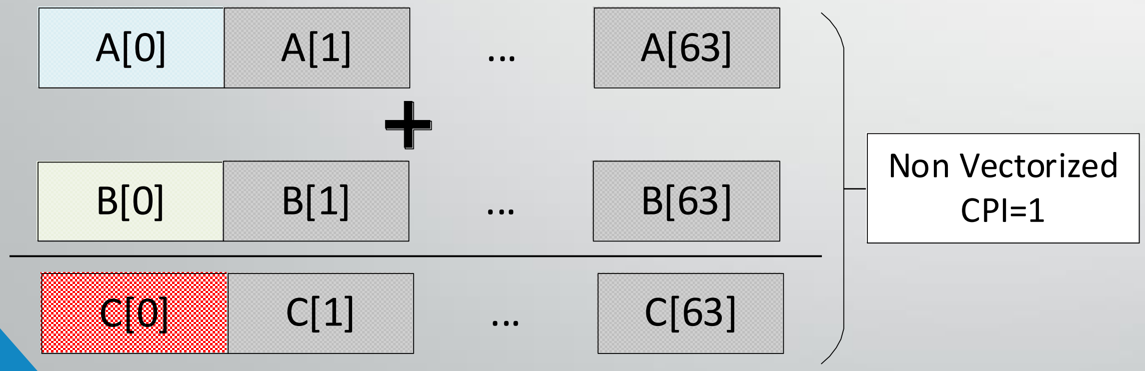

Total Execution Cycles
Total Execution Cycles Active
+
Total Execution Cycles Stalled
Core Related Stalls
Memory Related Stalls
Memory Related Stalls On-Chip
Memory Related Stalls Off-Chip

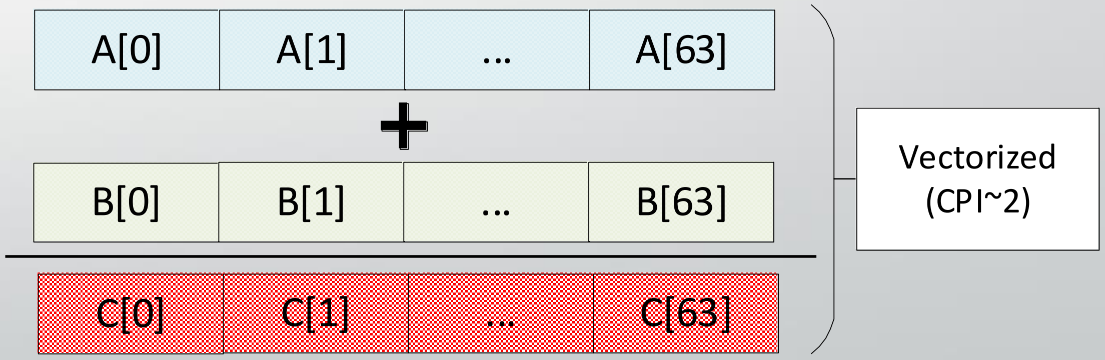
FS-CPI: Implementation and Its Shortcomings

Frequency Selection based on CPI (FS-CPI)

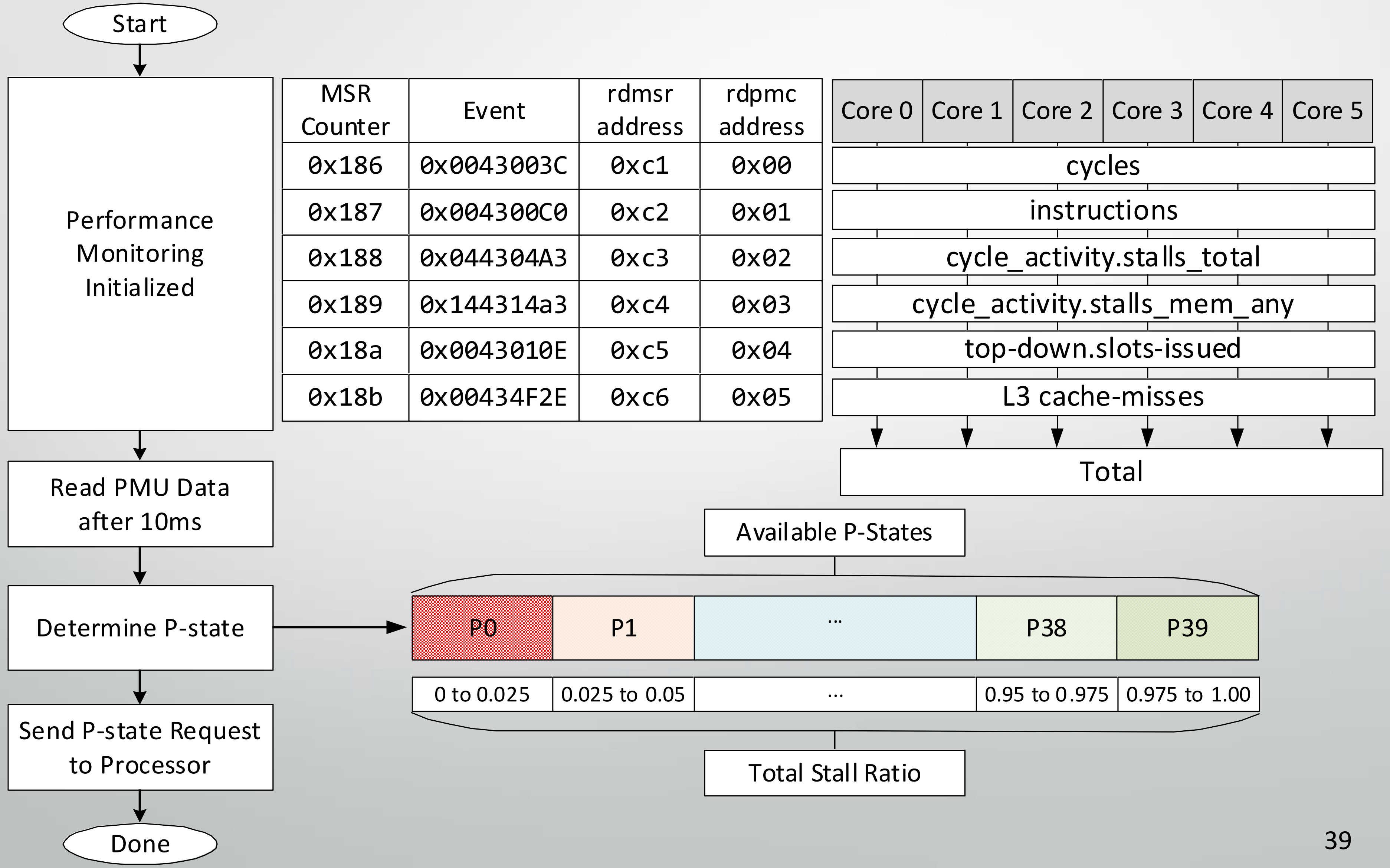
- Map CPI to available P-states
- Mapping range selected based on real-life workload (0-6)

- Why CPI is not ideal?
- Metric too sensitive to operating frequency
- Vectorized Code:
  - Vectorized code may have higher CPI but can get more work done in fewer instructions.
  - Does not capture the impact of vectorization





Implementation of Proposed Techniques



Start

Performance Monitoring Initialized

MSR Counter	Event	rdmsr address	rdpmc address
0x186	0x0043003C	0xc1	0x00
0x187	0x004300C0	0xc2	0x01
0x188	0x044304A3	0xc3	0x02
0x189	0x144314a3	0xc4	0x03
0x18a	0x0043010E	0xc5	0x04
0x18b	0x00434F2E	0xc6	0x05

Core 0 Core 1 Core 2 Core 3 Core 4 Core 5

cycles

instructions

cycle_activity.stalls_total

cycle_activity.stalls_mem_any

top-down.slots-issued

L3 cache-misses

Total

Read PMU Data after 10ms

Available P-States

Determine P-state

P0	P1	…	P38	P39
0 to 0.025	0.025 to 0.05	…	0.95 to 0.975	0.975 to 1.00

Total Stall Ratio

Send P-state Request to Processor

Done

Implementation Overhead & Invocation Rate

- Performance monitoring period: 10 ms
- OpenMP to read PMU from all core counters
- Implementation overhead: +3 ms
- Total duration: Idle CPU, running at 0.8 GHz: 13 ms

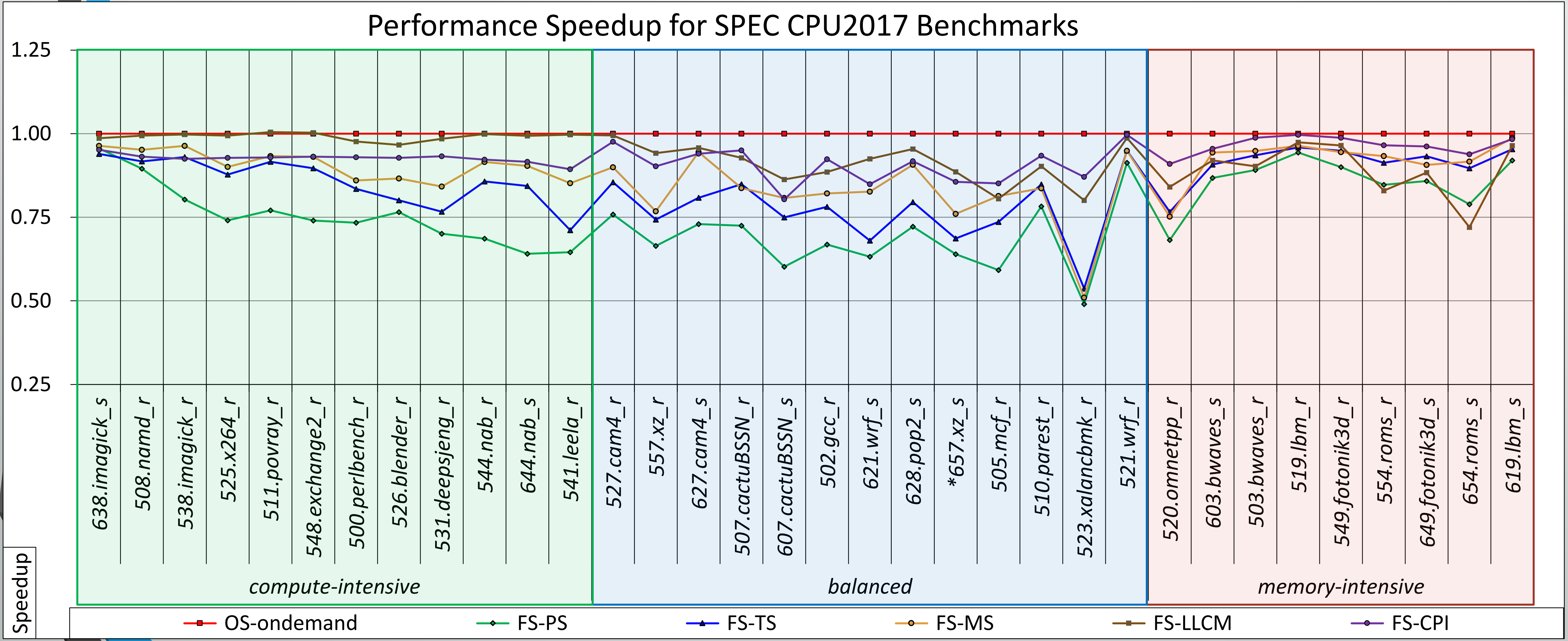
	PKG Power (W)	PP0-Power (W)
Ref (OS-ondemand)	4.50	1.00
FS-50ms (20 Hz)	6.05	2.55
FS-100ms (10 Hz)	5.25	1.75

Overview

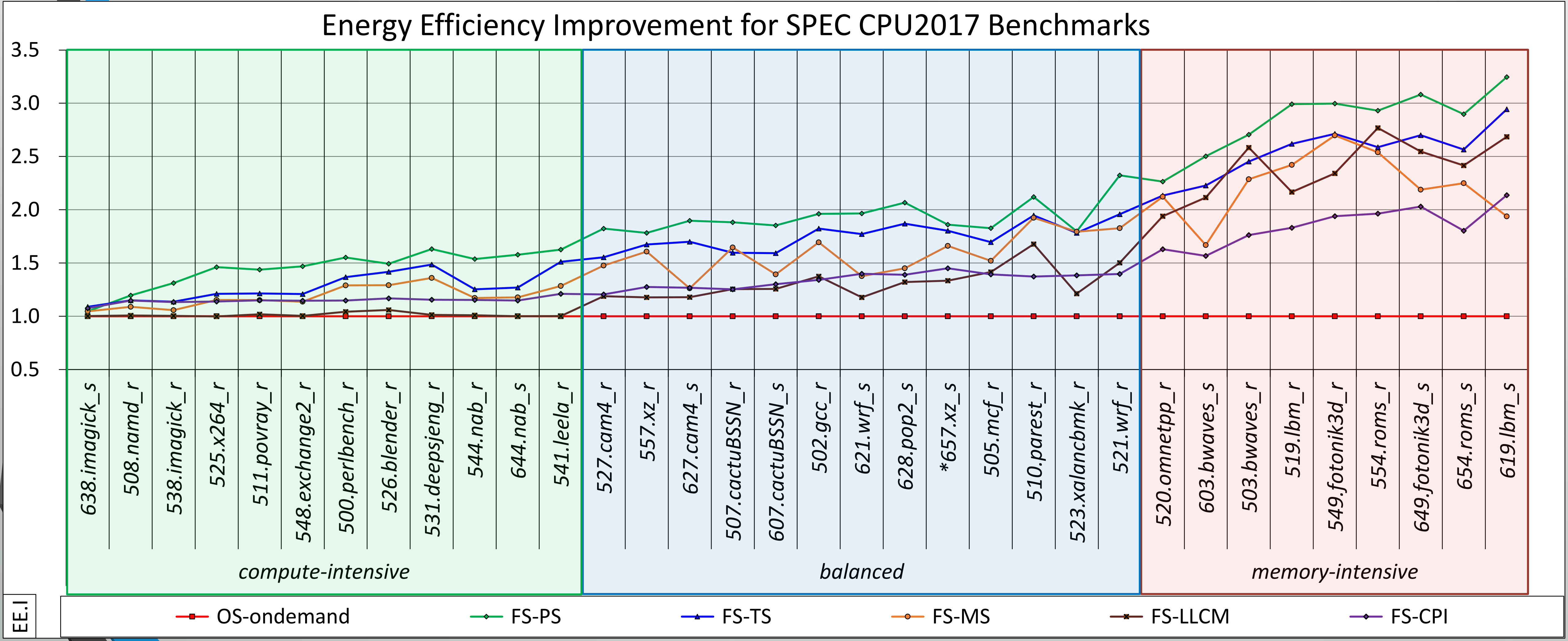
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Introduction
- Background & Motivation
- Experimental Setup
- SPEC CPU2017 Characterization
- Proposed DVFS Techniques
- **Results**
- Conclusions

SPEC CPU2017 Performance

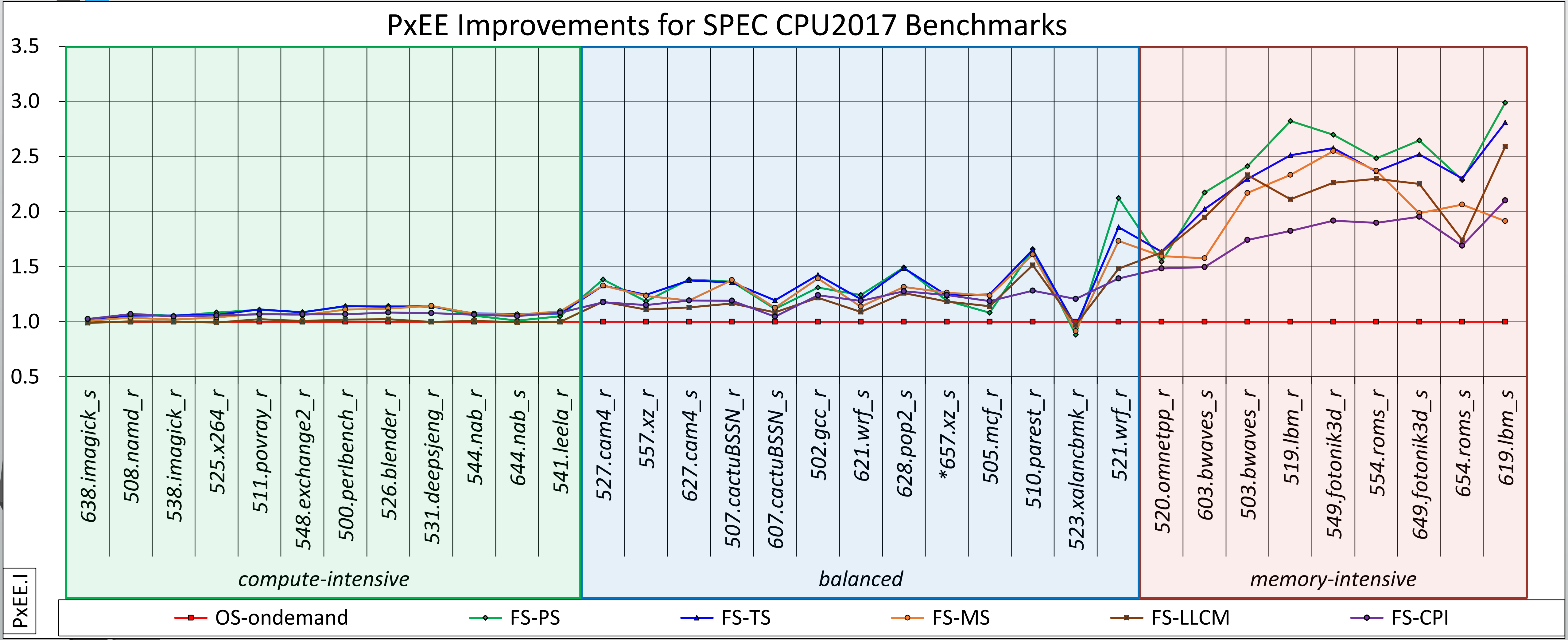


SPEC CPU2017 Energy Efficiency

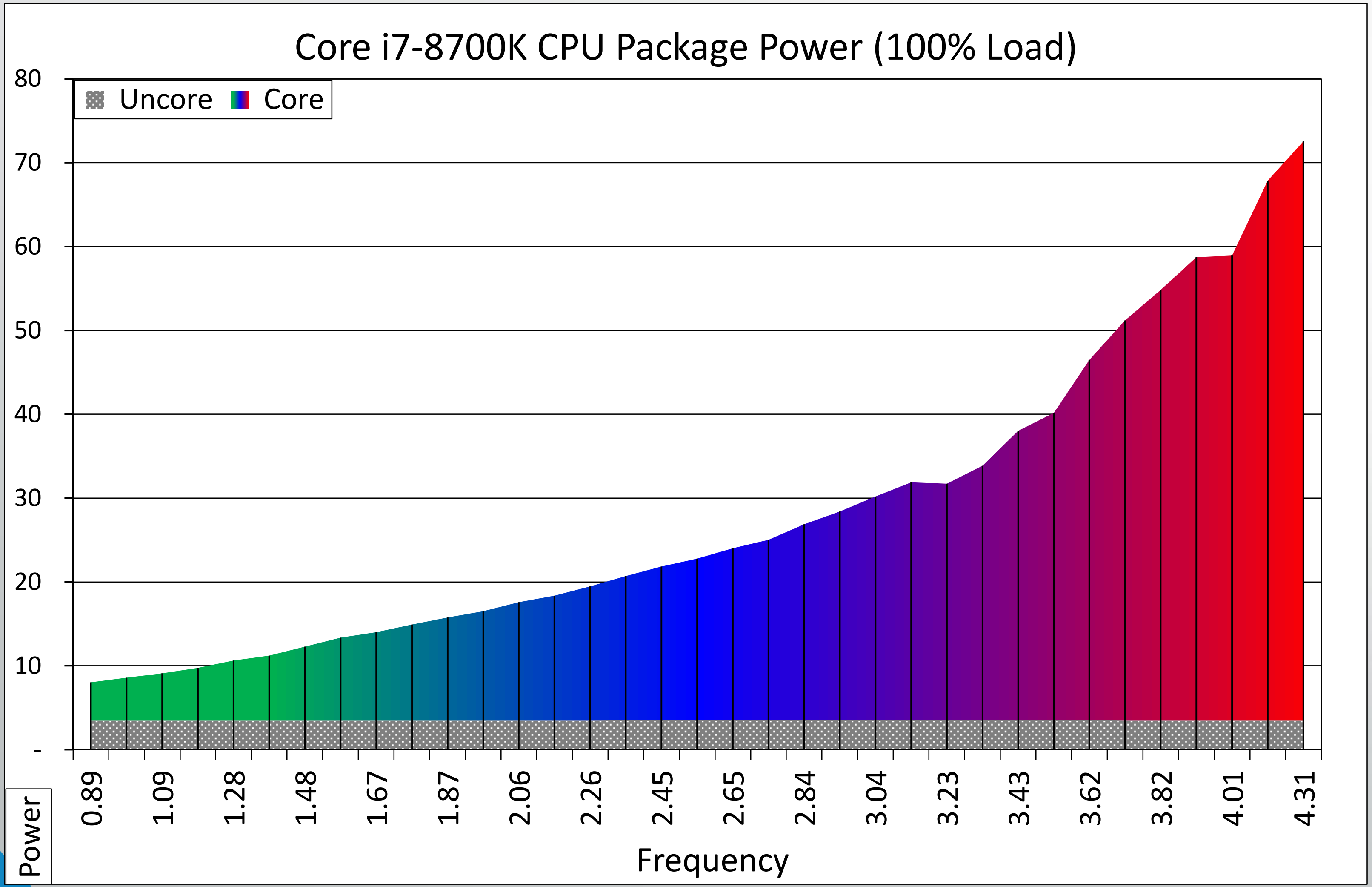


PxEE Summary for SPEC CPU2017 Benchmarks

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE



Future Work



- Mapping Techniques
- Hybrid Models
- Alternate Workloads

Conclusions

- DVFS is one of the most important tools in regulating processor power consumption
- OS implementation relies on CPU utilization ⇒ favors performance
- Existing FS-CPI can be unstable (vectorized code), limited benefits
- This dissertation proposes alternate techniques that significantly improve energy-efficiency of memory intensive applications

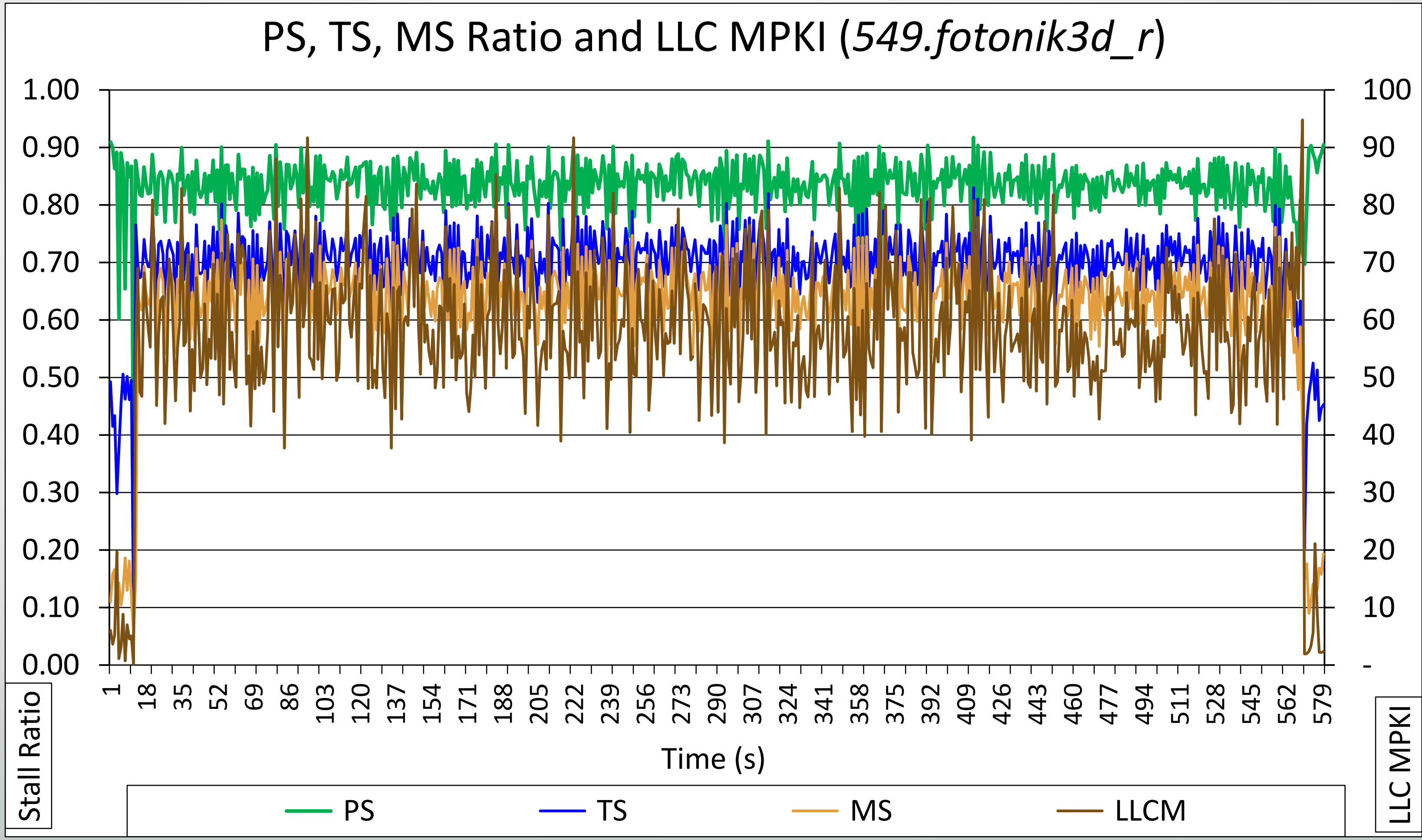
Technique	PxEE Gains w.r.t state-of-the-art
FS-PS	141%
FS-TS	132%
FS-MS	104%
FS-LLCM	110%

Outline

- Introduction
- Background
- Experimental Setup
- SPEC CPU2017 Characterization
- Proposed DVFS Techniques
- Results
- Conclusions
- Backup

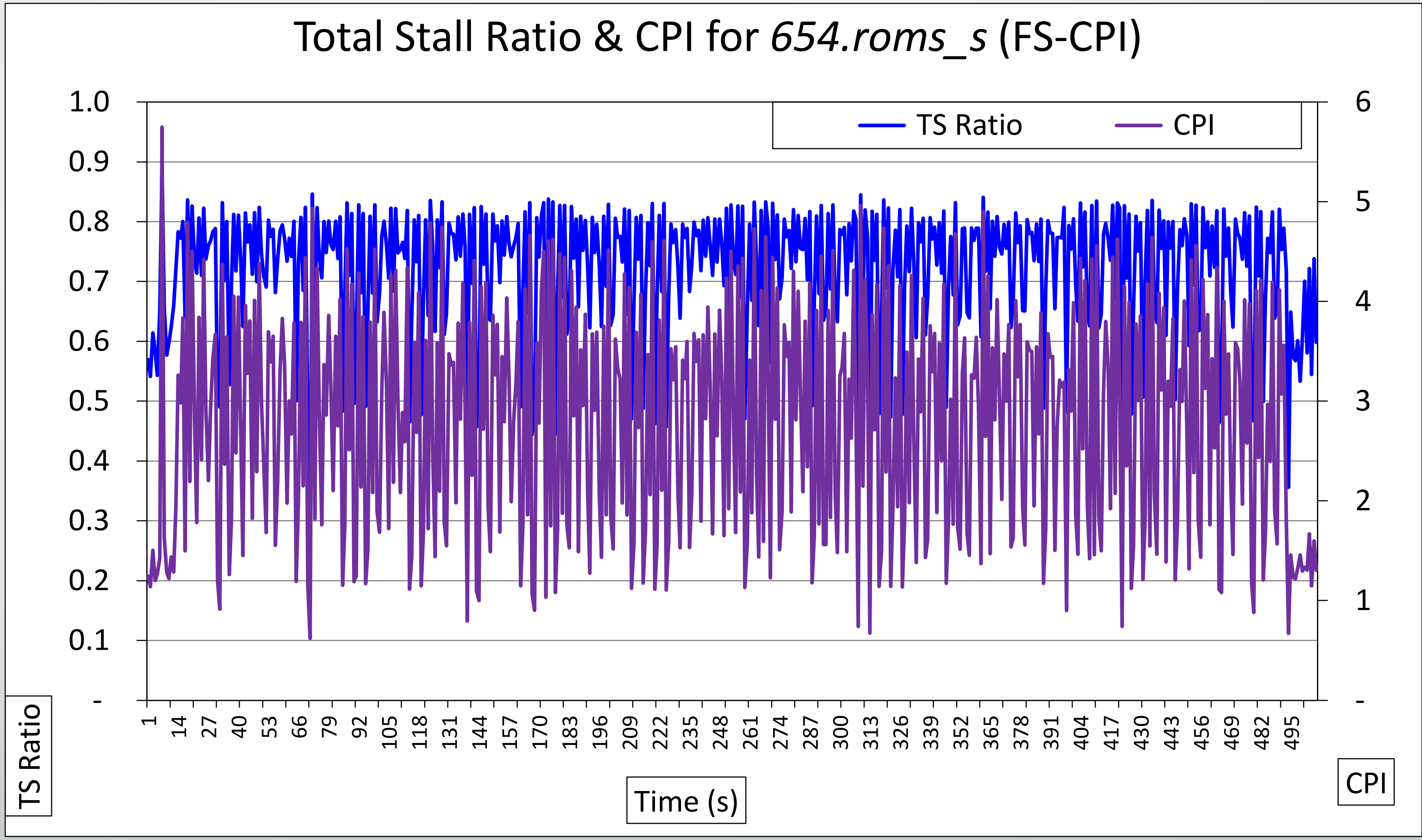
Runtime Metrics Measurements for *549.fotonik3d_r*

	FS-PS	FS-TS	FS-MS	FS-LLCM
P.S	0.90	0.95	0.94	0.96
EE.I	3.00	2.71	2.70	2.34
PxEE	2.70	2.58	2.55	2.26

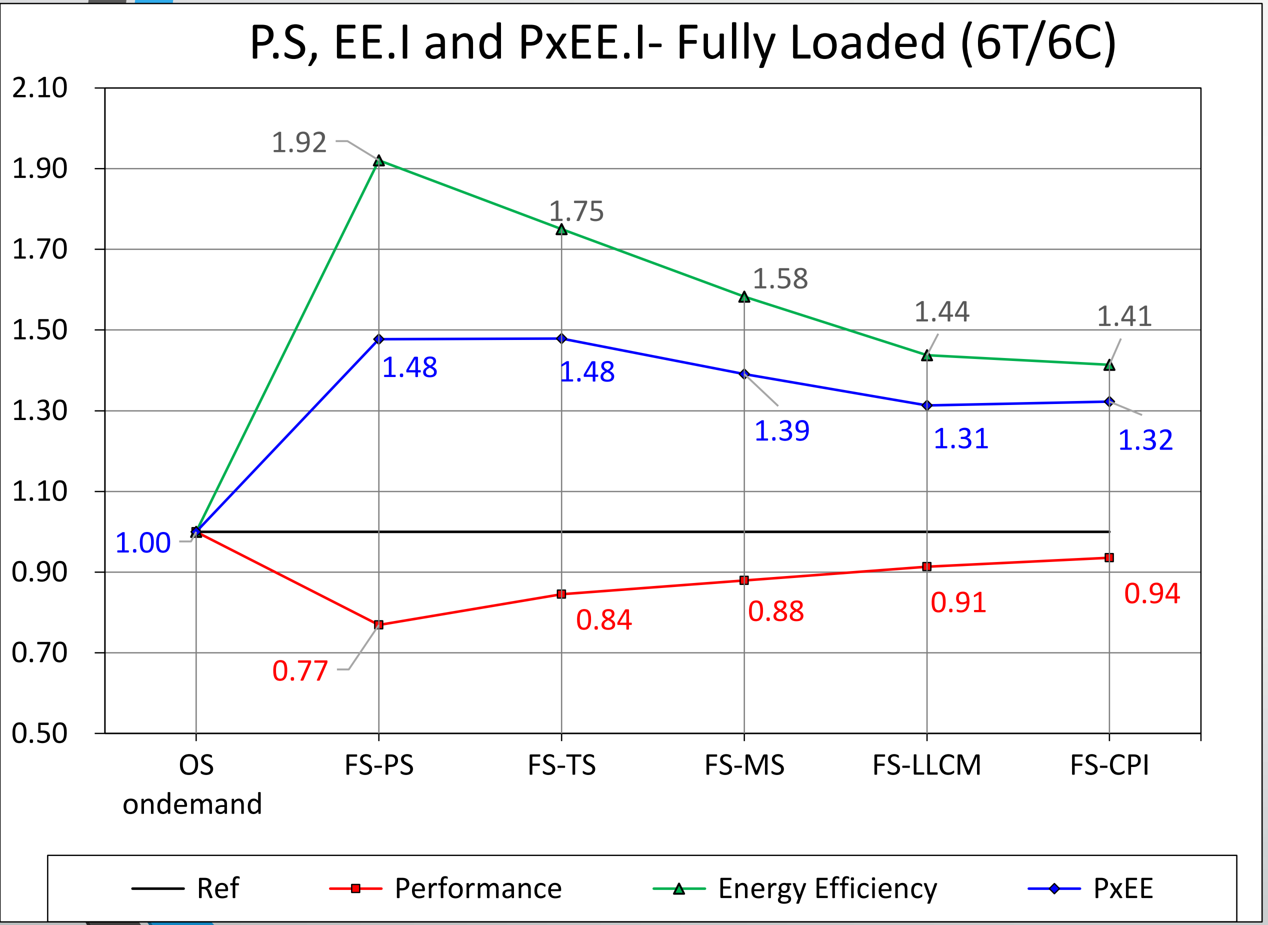


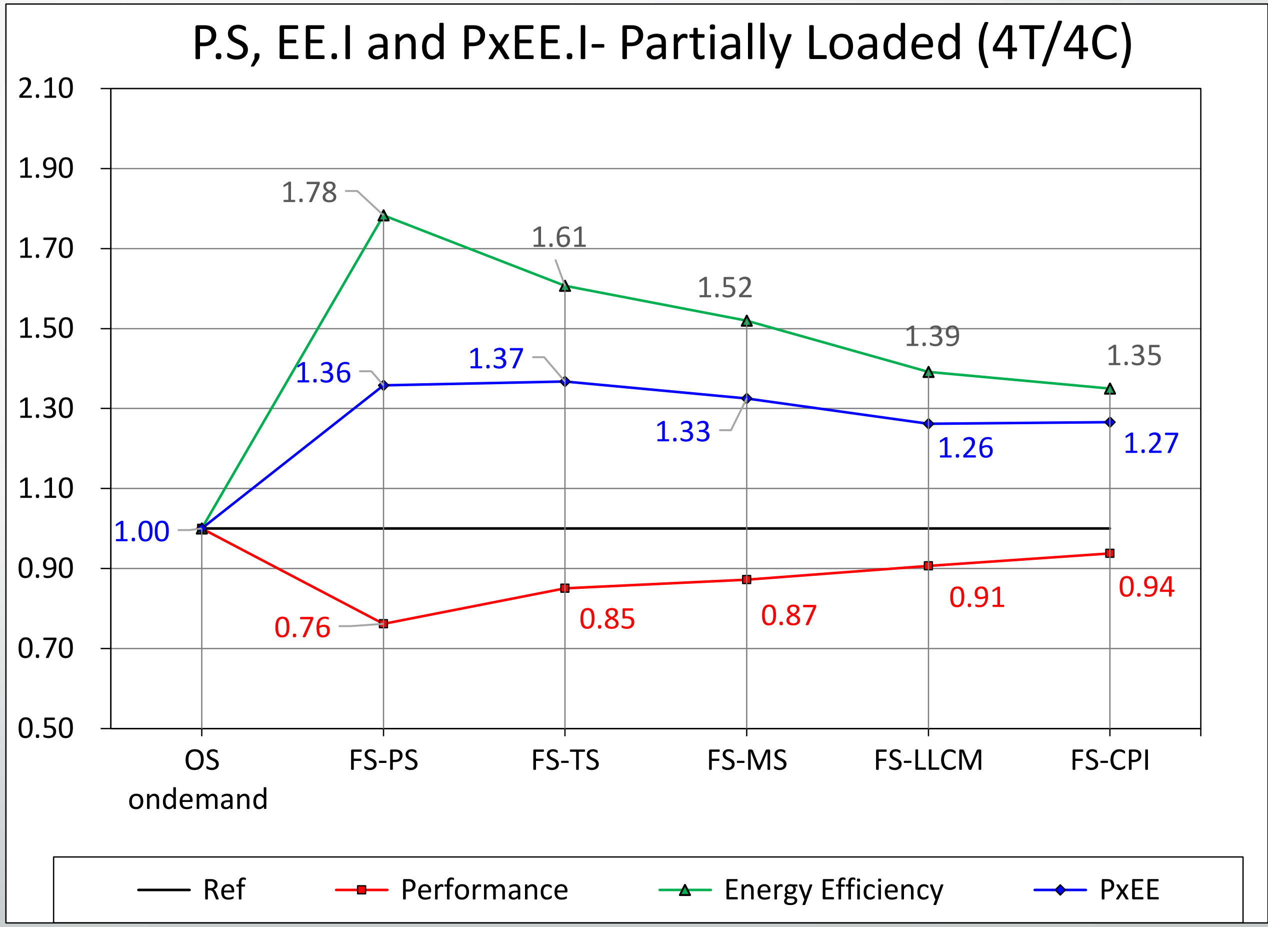
Runtime Measurements of the Total Stall Ratio and the Average CPI for *654.rom_s*

- FS-TS
  - P.S: 0.90
  - EE.I: 2.57
  - PxEE.I: 2.30
- FS-CPI
  - P.S: 0.94
  - EE.I: 1.80
  - PxEE.I: 1.69

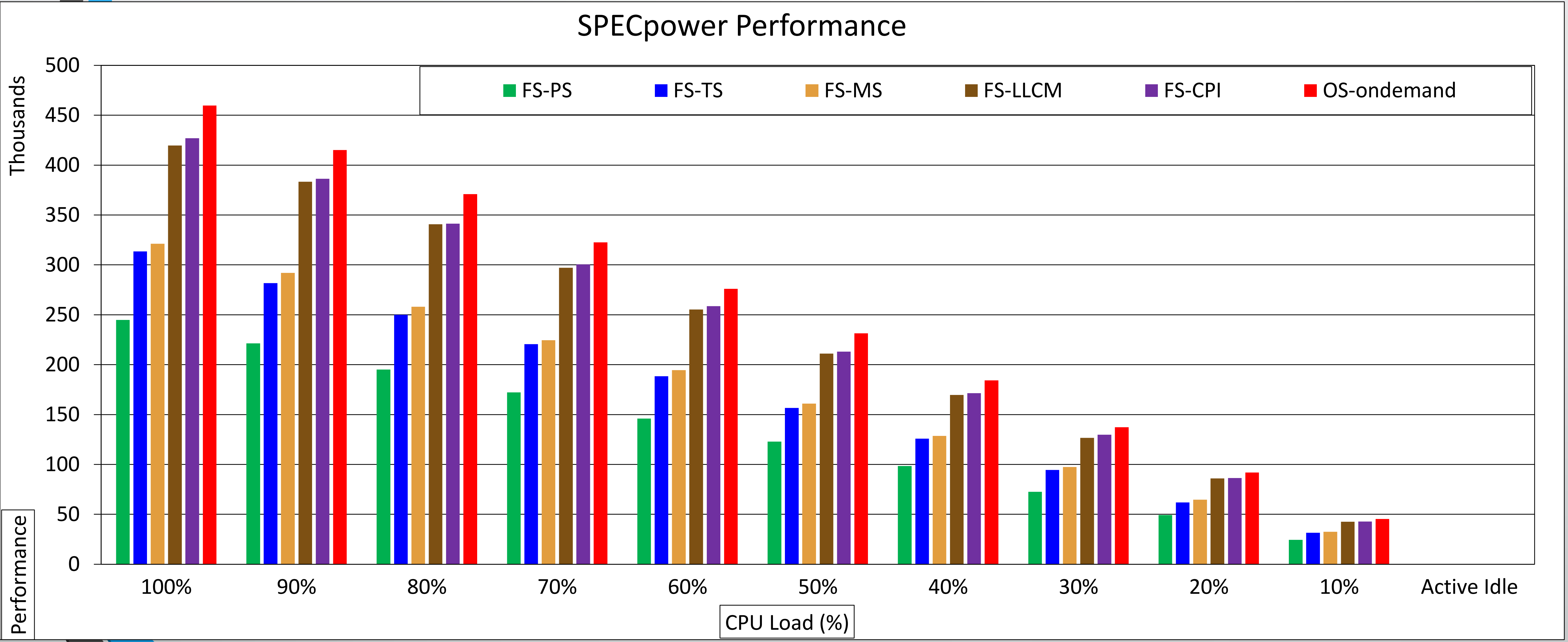

Total Stall Ratio & CPI for *654.roms_s* (FS-CPI)

SPEC CPU2017 Summary

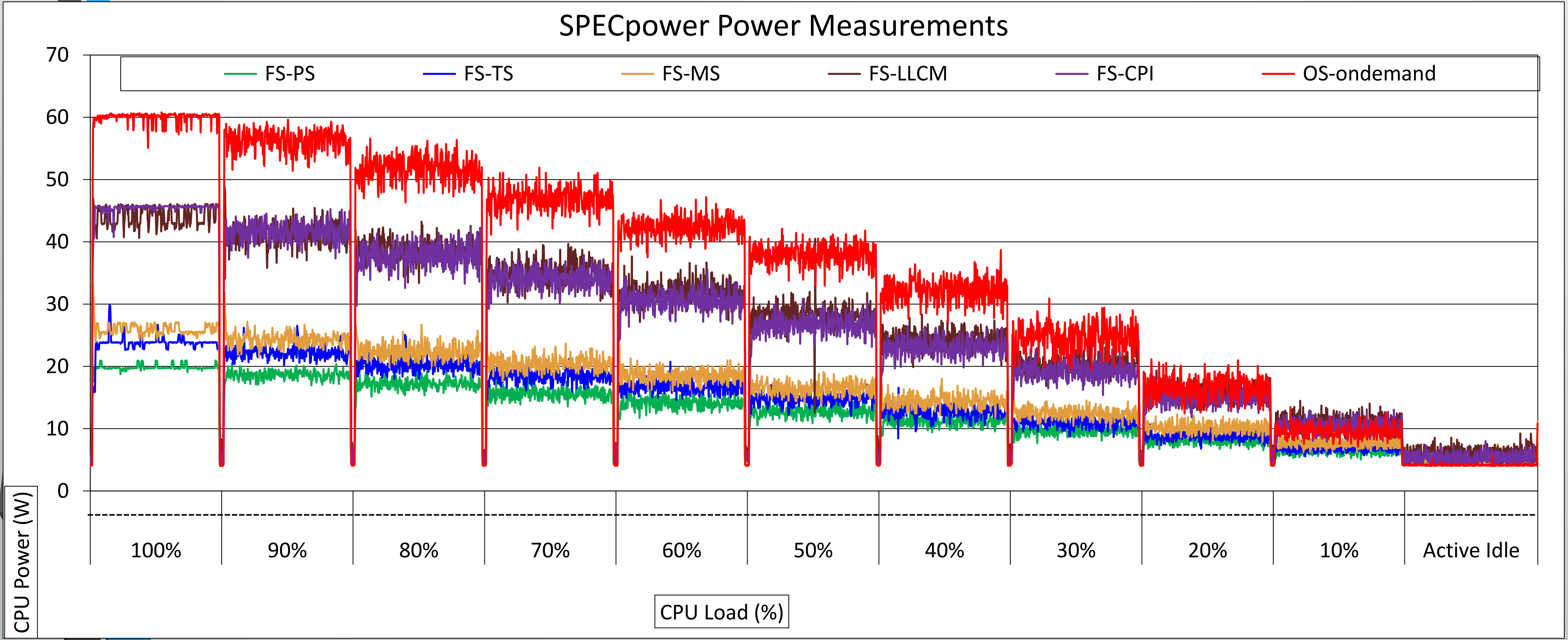




SPECpower: Performance

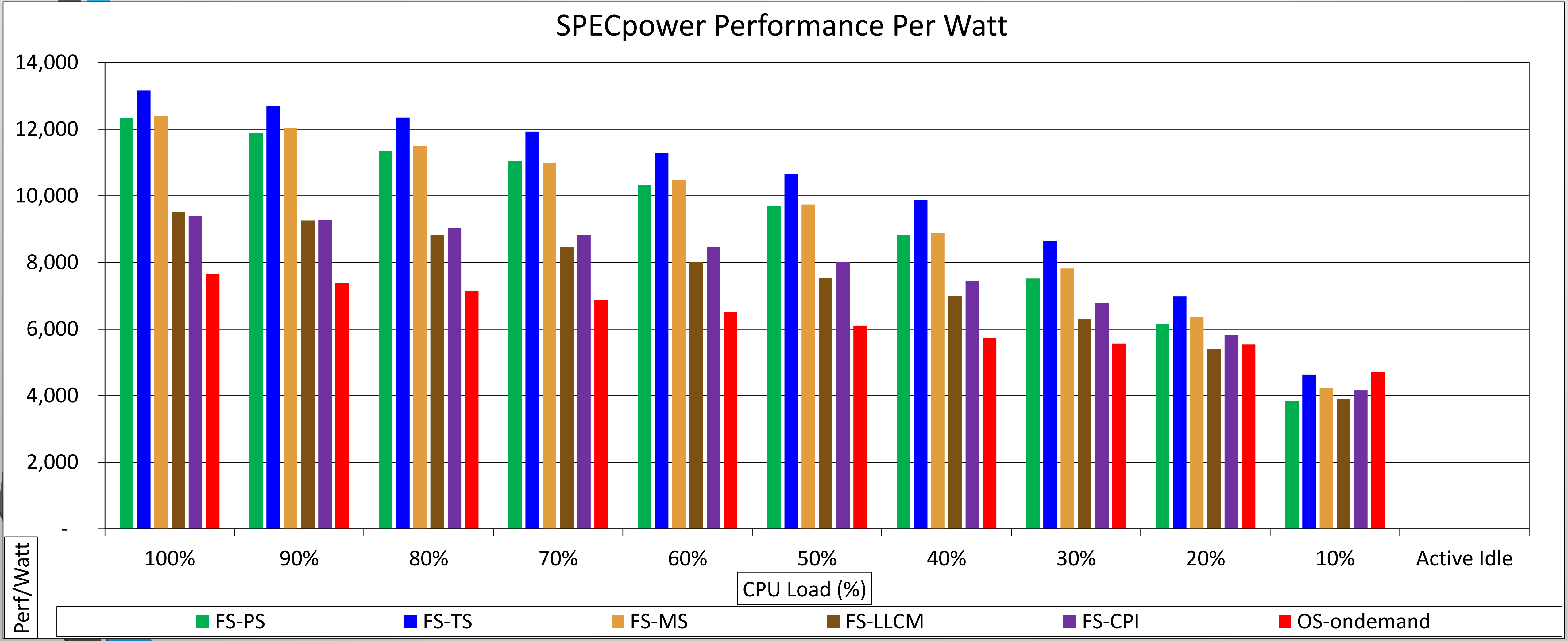


SPECpower: Power



SPECpower: Performance/Watt

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE



LaCASA