A Deep Dive into Memory Access: Extended Roofline for Power Management

Presentation of Bachelor Thesis

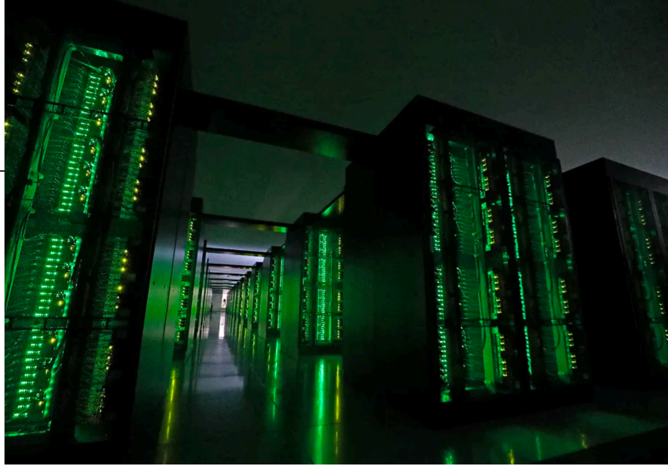
First examiner: Prof. Dr. Matthias Müller
Second examiner: Prof. Dr. Martin Schulz
Adviser: M. Sc. Bo Wang

Lehrstuhl für Hochleistungsrechnen (Informatik 12), IT Center, RWTH Aachen
Lehrstuhl für Rechnerarchitektur & Parallele Systeme. Technische Universität München

Liangkun He, 27.07.2021

Outline

Motivation


[2]

- Supercomputer’s power consumption is significant
  - Fugaku is the top one supercomputer. It consumes 28.3MW to deliver 415.53 Pflops/s [1]
- Large power consumption brings many problems
  - High energy cost
  - Heats dissipation
  - High carbon dioxide emissions
- Exascale supercomputer (1ExaFlop/s=10^3 PFlop/s)
  - Next generation supercomputers that yet to come
- Performance tuning
  - Power consumption cannot exceed the power budget
  - Performance and power consumption optimizations are important

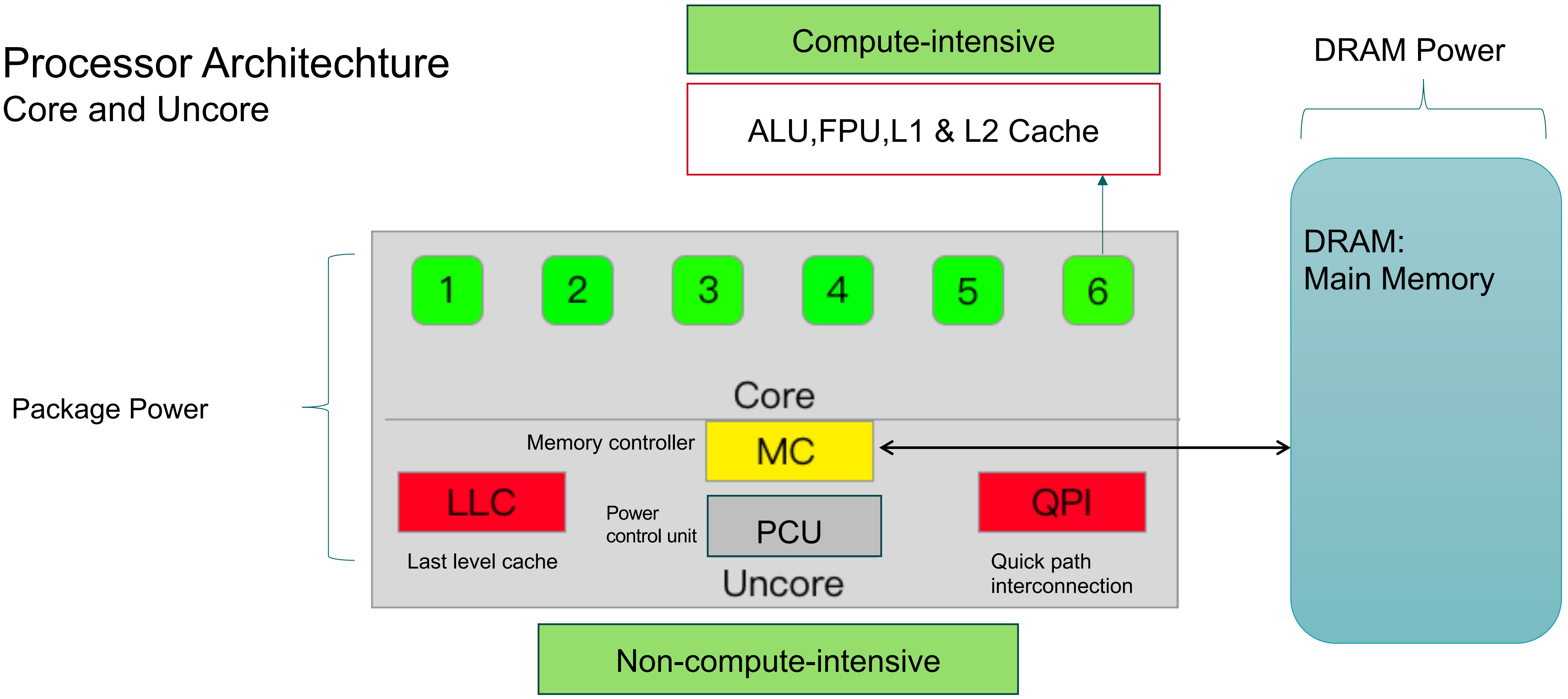
Motivation

Contribution

- Study the processor architecture
  - Core vs. uncore
  - Different components of a processor have their individual frequency
- Implement the Roofline model based on the sampling approach
  - Identify the execution patterns of an application for each sampling intervals
    - Compute bound or memory bound
  - Modify each component's frequency respectively during runtime to optimize the performance and power consumption
  - Propose the time interval analysis method
    - Analyse the activities of CPU and memory for each sampling intervals
    - Extend the Roofline model with the latency bottleneck
- Evaluate the implementation on different benchmarks

Background

- Processor Architechture
- Core and Uncore



Background

Running Average Power Limit(RAPL)

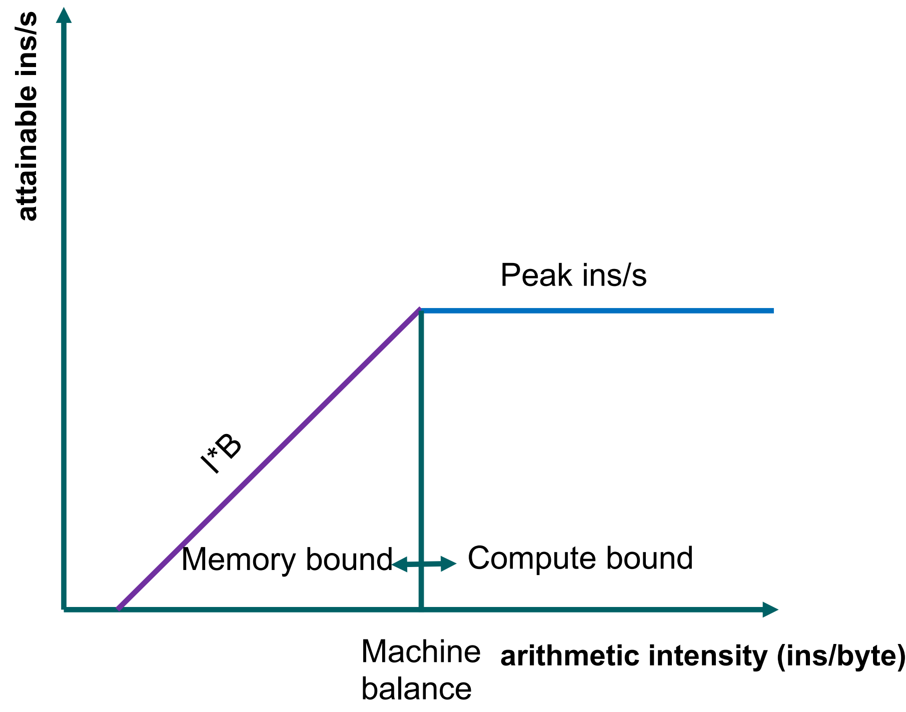
- Set power cap for processors in a time interval
  - Keep the core or uncore frequency low to make sure they do not exceed the power cap

- RAPL is not aware of the execution patterns of the running application
  - Compute intensive or memory intensive
  - For memory intensive application such as Stream benchmark, RAPL may reduce the uncore frequency and thus lower the performance

Roofline Model

- $P = \min(P_{Peak}$, I*B) [Ins/s]
  - Performance bottleneck is either
    - Computation: P_{Peak} [Ins/s]
    - Data transfer: I*B [Ins/Byte * Byte/s]
- B: peak memory bandwidth
- I : arithmetic intensity [Ins/Byte]
  - Describe the number of instructions per data transfer
  - Using Ins/Byte has advantages over Flop/Byte in implementation
    - Applications may execute both integer and floating-point operations
    - Ins/Byte is easier to measure and more accurate
- Machine Balance
  - P_{Peak} = I*B
  - Best use of computational performance and memory bandwidth

- I < machine balance → memory bound
- I > machine balance → compute bound



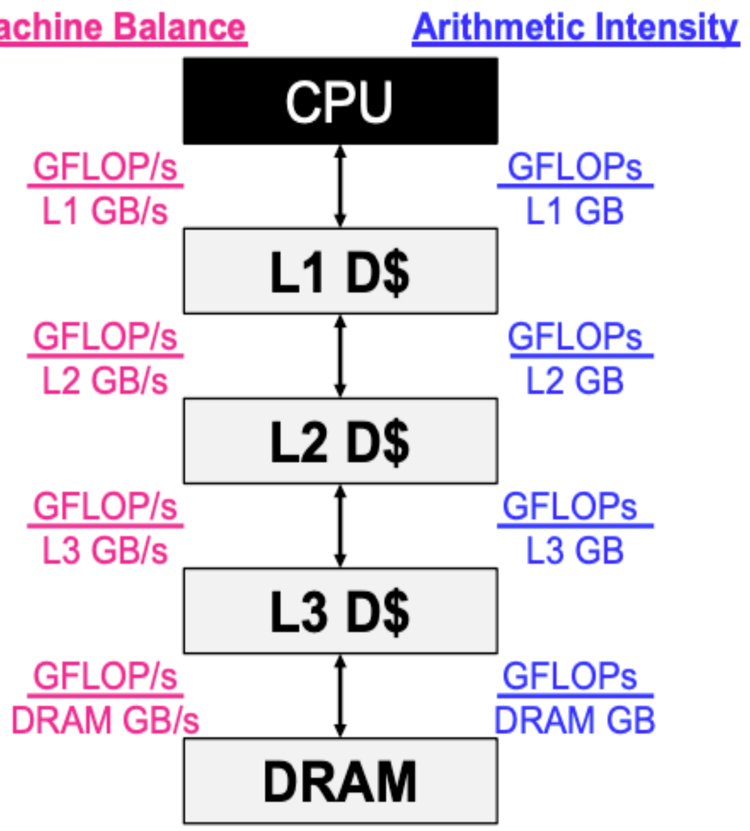
Roofline Model

Memory hierarchy

- Cache
  - Small but fast memory that are integrated in the processor

- Each level has its own bandwidth hence diffenrent machine balances

- The amounts of data that transfered between memory levels are different hence different arithmetic intensities

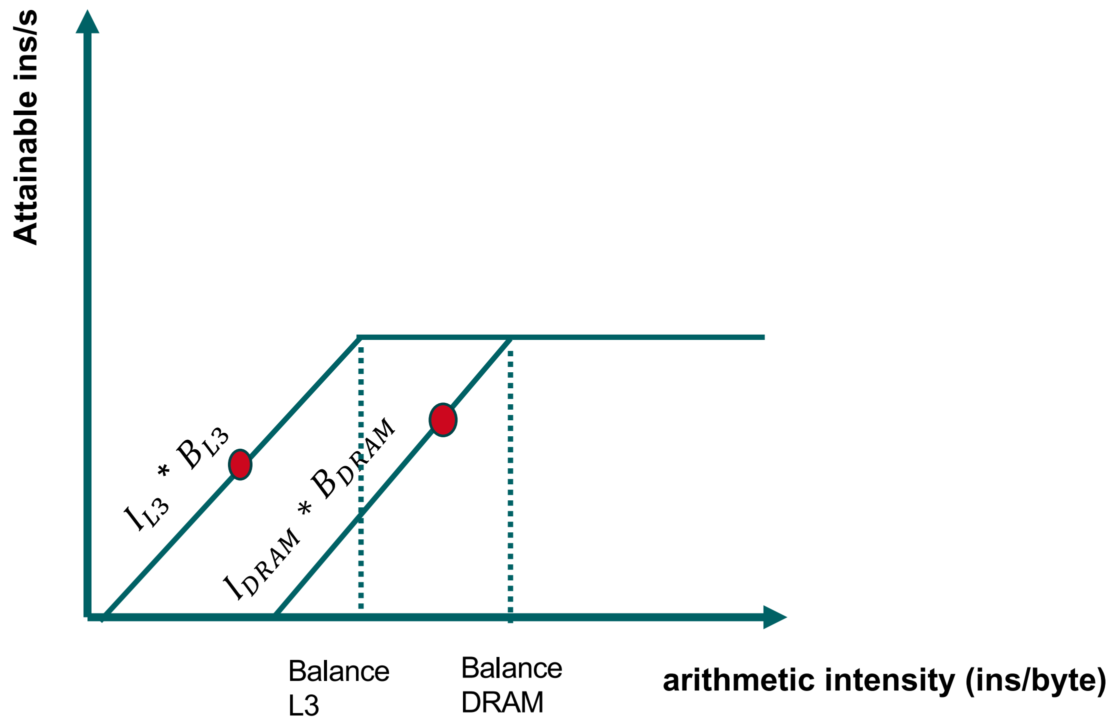


[3]

Roofline Model

This work focuses on L3 cache and DRAM levels

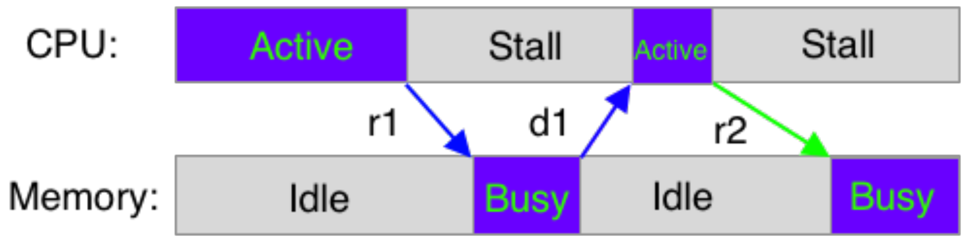
- The performance is bound by the minimal attainable ins/s
  - P= min($P_{Peak}, I_{L3} * B_{L3}$, $I_{DRAM} * B_{DRAM}$)
- Construct two bandwidth ceilings
  - L3 cache and DRAM
- Bound to L3 cache in the example

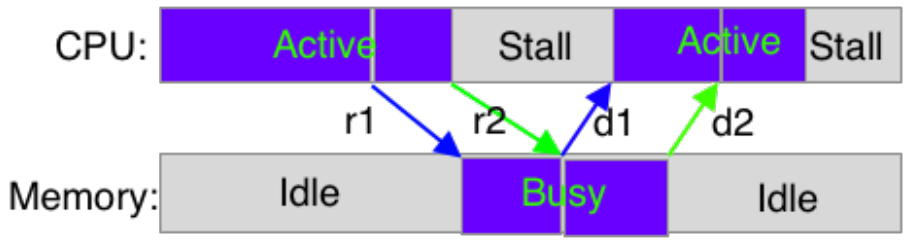


Roofline Model

- Assumption
  - Computation and data transfer between CPU and memory overlap perfectly.(ignore memory latency)
- Fact
  - Memory latency may not be eliminated completely

- Out-of-order execution:

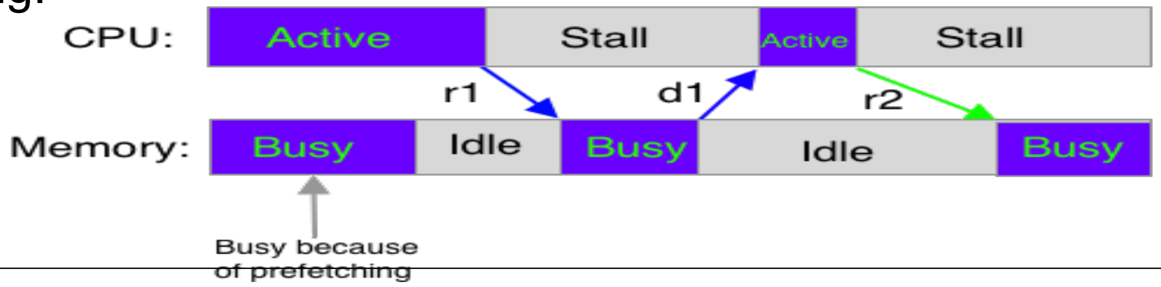




In-order execution

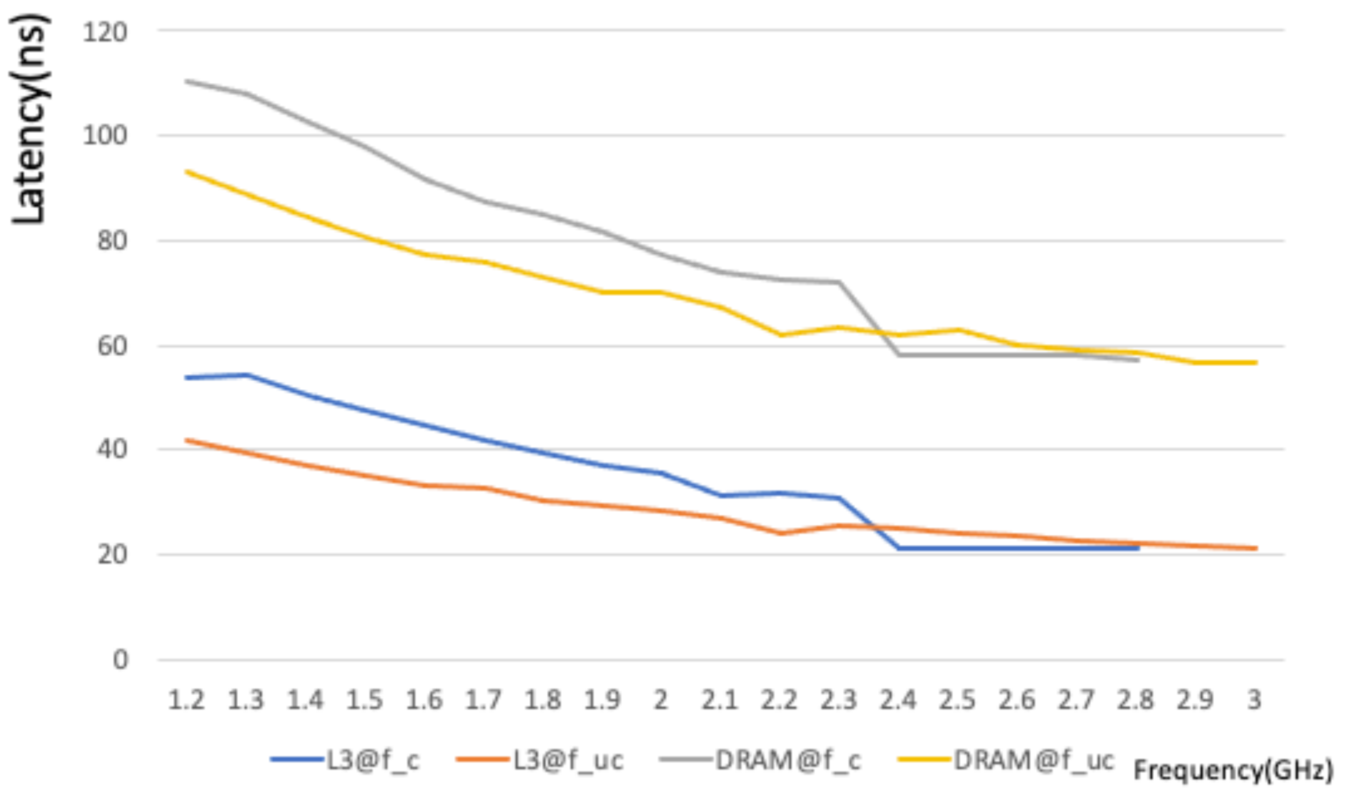
Out-of-order execution

- Hardware prefetching:



Roofline Model

- How core and uncore frequency scaling affects the L3 and DRAM latency
- Latency is measured by tinyBench
- Latency increases when core/uncore frequency decreases

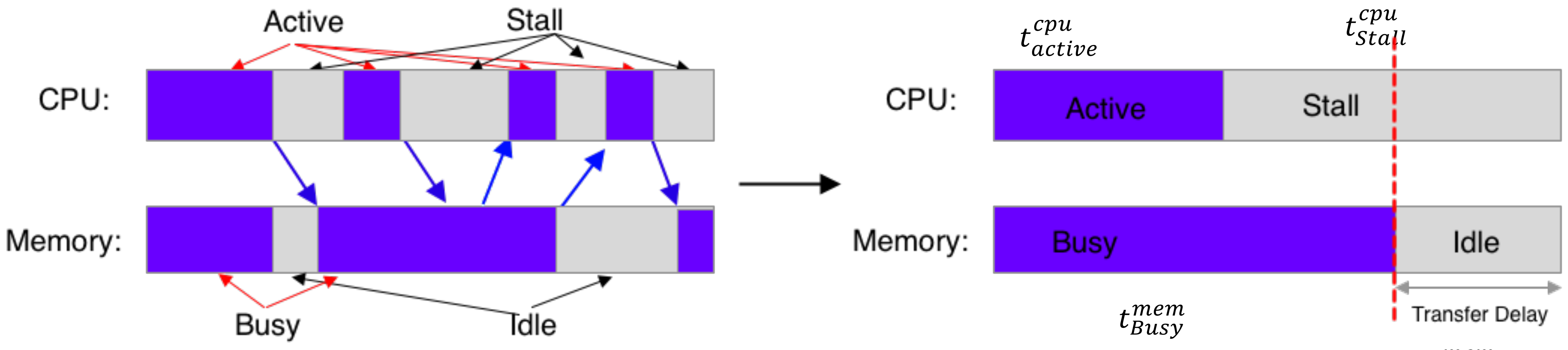


Extended Roofline Model

Time Interval Analysis

- Transfer delay: Part of memory latency that cannot be eliminated

- Time interval analysis
  - Calculate the transfer delay based on the activities of CPU and memory in a sampling interval
  - The summarized activities can be derived from hardware events



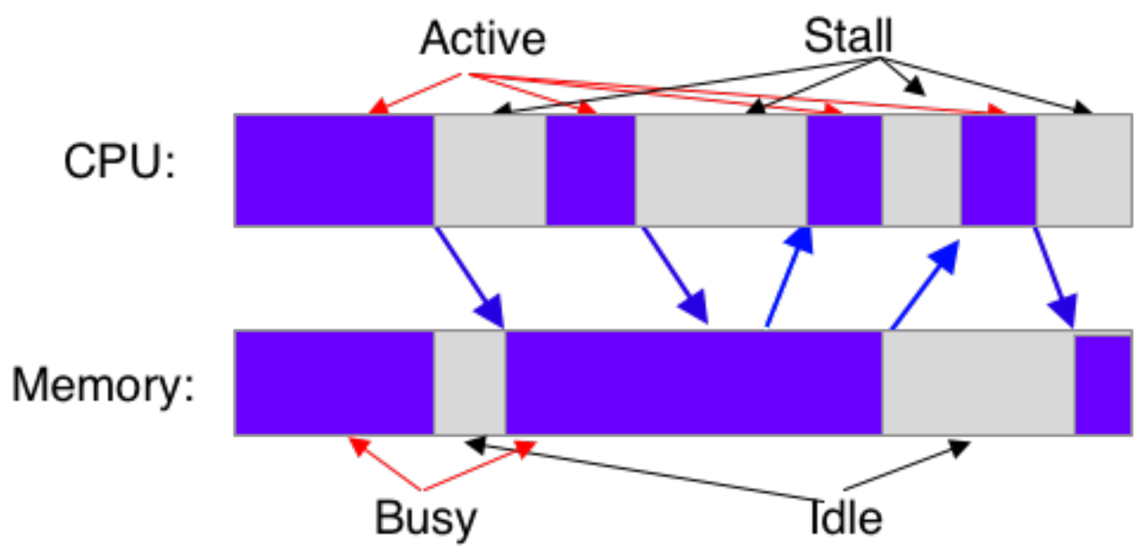
Real time activities

Summarized activities

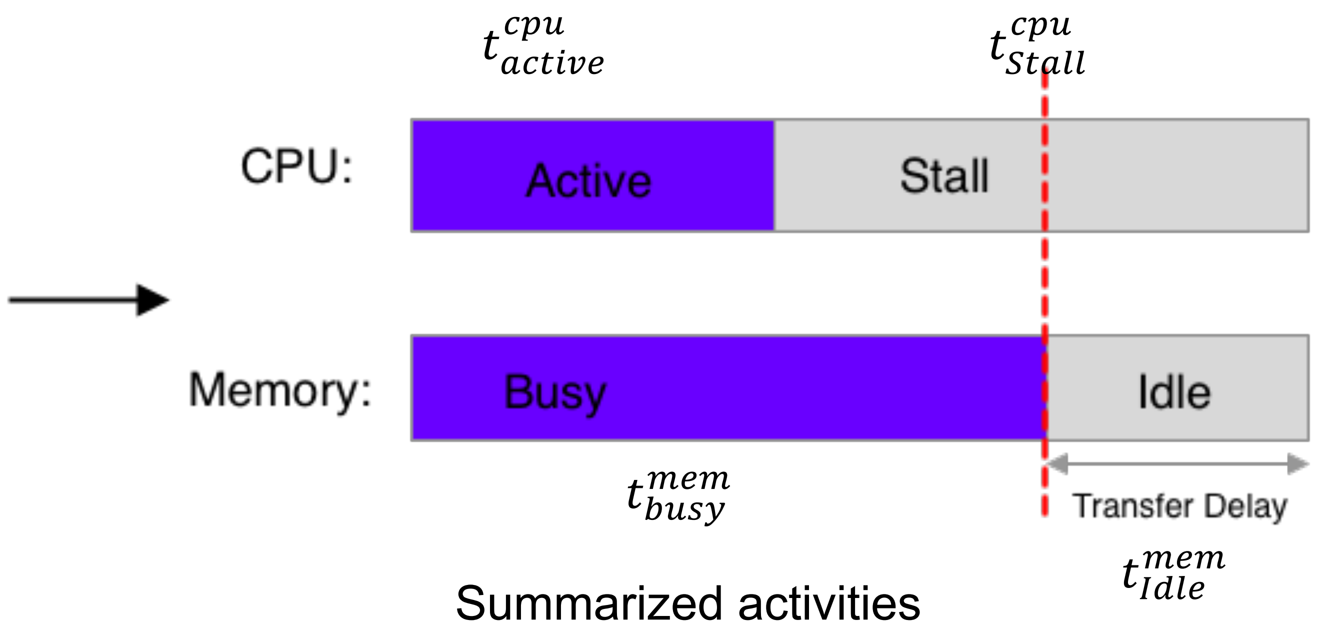
Extended Roofline Model

- Transfer delay is approximated to $\min(t_{Stall}^{cpu}, t_{Idle}^{mem})$
  - Time for synchronization between CPU and memory
  - When CPU stalls and memory is idle

- Transfer delay bound with memory domination
  - Slightly decrease the core frequency



Real time activities

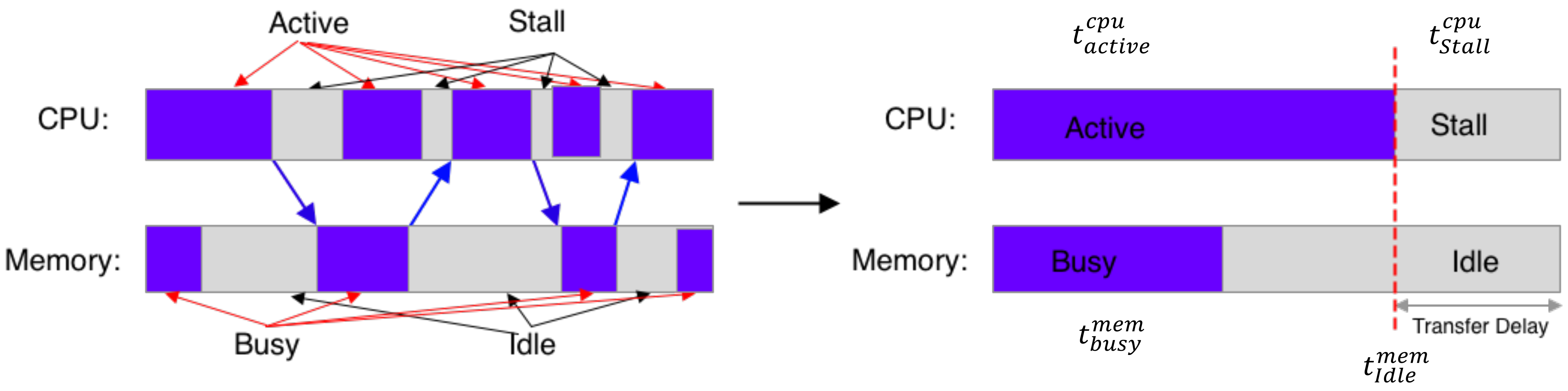


Summarized activities

Extended Roofline Model

- Transfer delay is approximated to min(t_{Stall}^{cpu}, t_{Idle}^{mem})

- Transfer delay bound with compute domination
  - Slightly decrease the uncore frequency



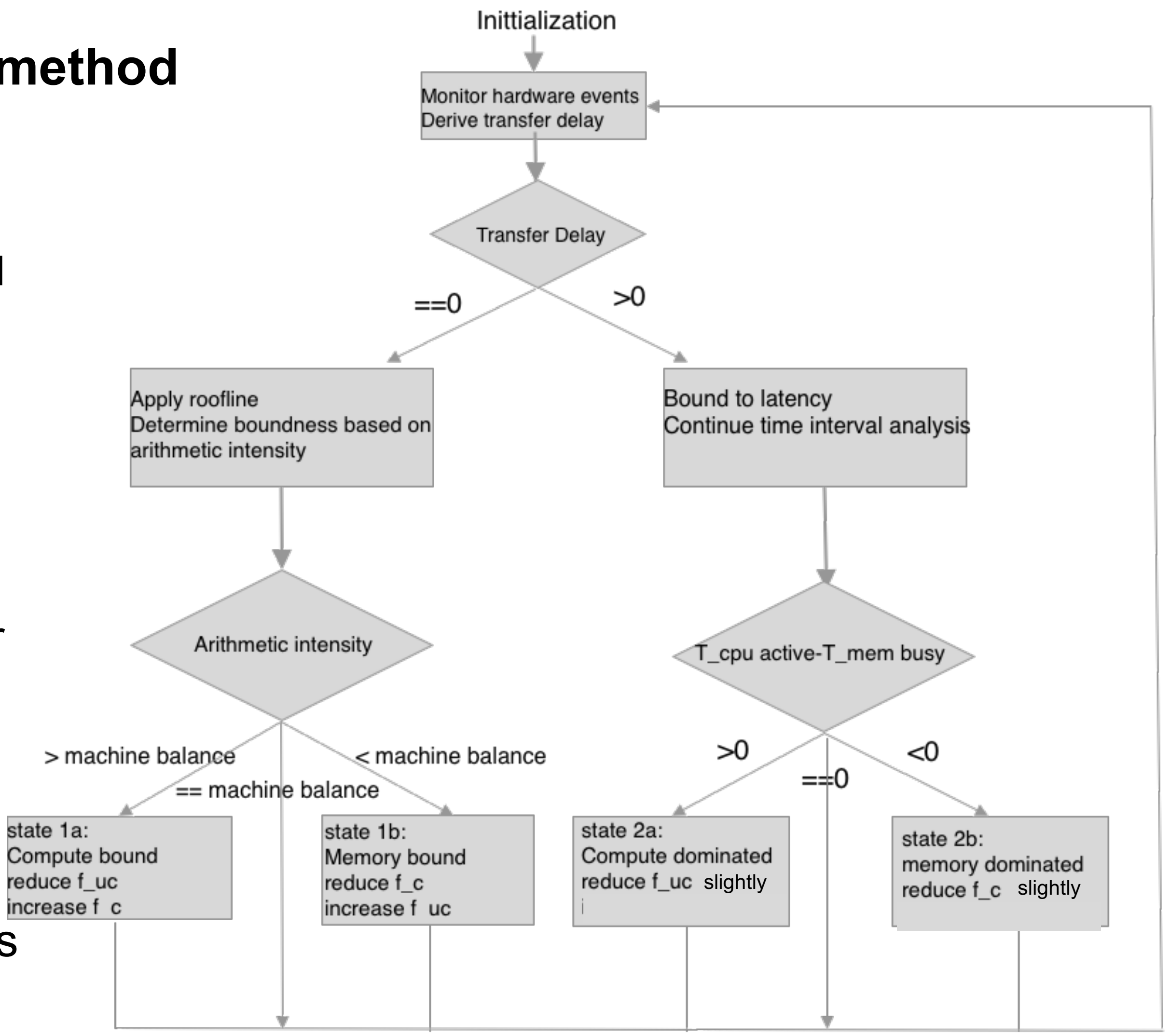
Implementation

Extended Roofline Model based on sampling method

- Sampling interval: 0.3s
  - Less over head
  - Detects some applications that vary from compute bound to memory bound.

- Initiation
  - Core and uncore frequency is set to maximum
- Measure hardware events
  - Core/uncore frequency, instructions, cycles, data transfer of LLC and DRAM, PKG energy and power…
  - Derive the transfer delay, CPU active and memory busy time, arithmetic intensities…

- If transfer delay approximates to zero → Roofline
- If transfer delay > 0 → Continue time interval analysis



Implementation

Hardware Platform and Tools for Measuring Hardware Events

- Haswell Intel (R) Xeon (R) processor E5 v3 @ 2.3GHz
  - Core frequency: 1.2 GHz ~ 2.8 GHz
  - Uncore frequency: 1.2 GHz ~ 3.0 GHz
  - L3 cache: 45 MB
  - DRAM: 32 GB
- Perf
  - Measuring core events and DRAM read/write
  - Lack uncore support
- Libmsr
  - Measuring uncore activities: LLC read/write

Evaluation

Benchmarks

- Stream (to evaluate the boundary cases)
  - Stream_L3 : LLC intensive
    - Total memory required = 20.6 MiB < L3 cache size: suitable for L3 cache testing
  - Stream_DRAM : DRAM intensive
    - Total memory required >> L3 cache size: suitable for DRAM testing

- NAS ***Parallel Benchmarks***
  - Memory intensive: Conjugate Gradient(CG) Multi-Grid (MG) Fourier Transform (FT)
  - Compute intensive: Embarrassingly Parallel (EP)
  - LLC intensive: SP
  - Different execution patterns: BT

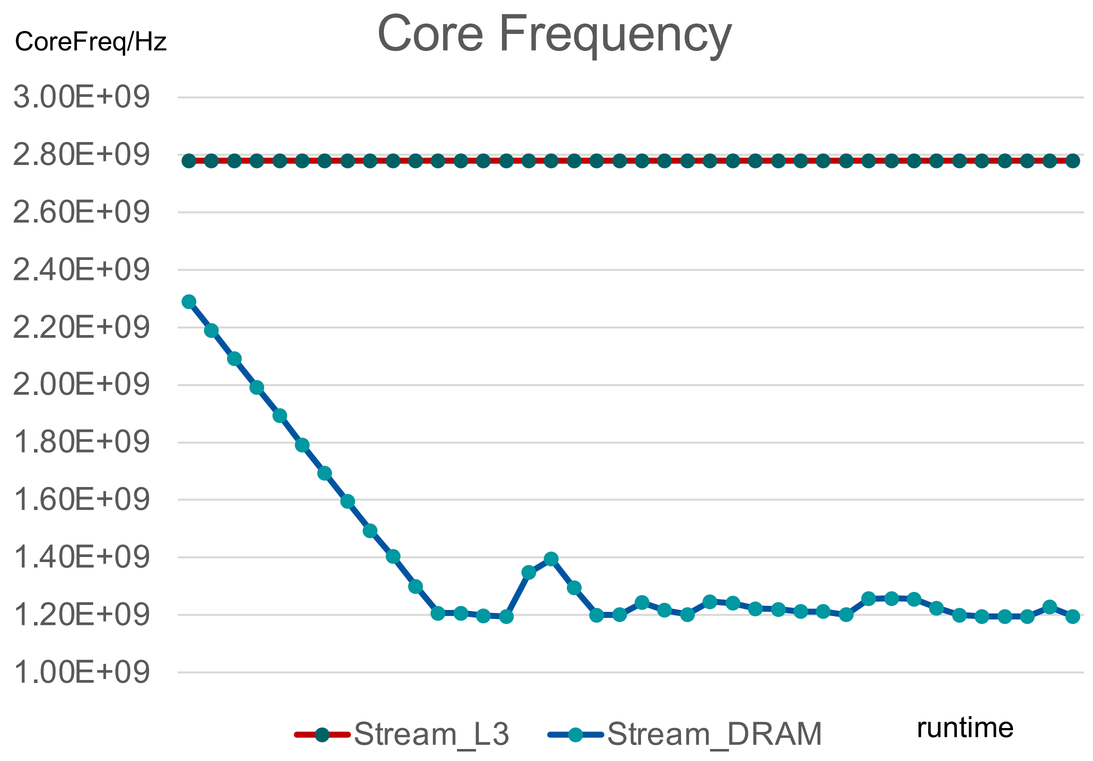
Evaluation

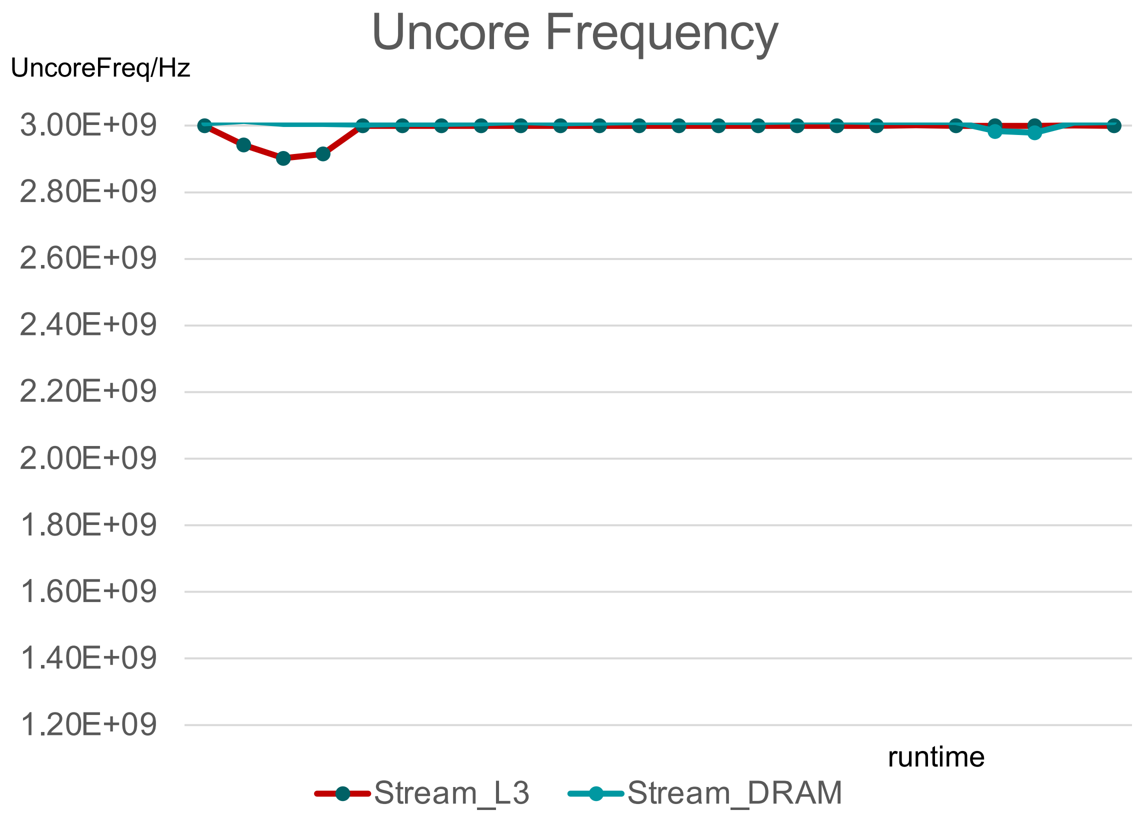
- Stream (without power cap):
  - Measure core and uncore frequency during runtime by the extended Roofline model
  - PKG power saving, energy saving and slowdown compared to the default (f_c = 2.8GHz, f_{uc} = 3.0GHz)

- NAS Parallel Benchmarks:
  - Energy optimization (without power cap):
    - PKG power and energy saving, and slowdown compared to the default (f_c = 2.8GHz, f_{uc} = 3.0GHz)

  - Performance and energy optimization (under a power cap (69W)):
    - Speedup and energy saving compared to the default

Stream

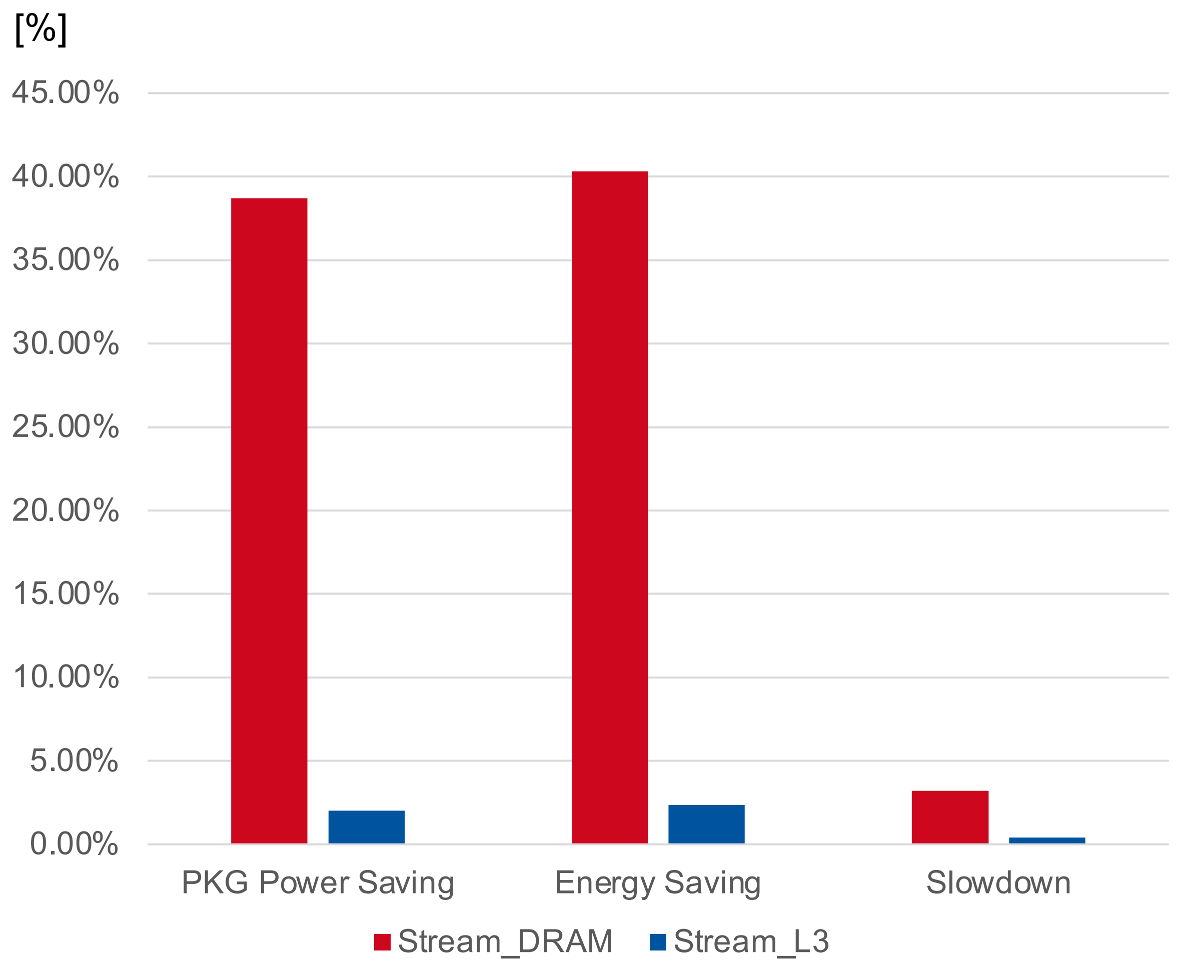

Core Frequency
CoreFreq/Hz
3.00E+09
2.80E+09
2.60E+09
2.40E+09
2.20E+09
2.00E+09
1.80E+09
1.60E+09
1.40E+09
1.20E+09
1.00E+09
Stream_L3
Stream_DRAM
runtime


Uncore Frequency
UncoreFreq/Hz
3.00E+09
2.80E+09
2.60E+09
2.40E+09
2.20E+09
2.00E+09
1.80E+09
1.60E+09
1.40E+09
1.20E+09
runtime
Stream_L3
Stream_DRAM

Stream

PKG Power, energy saving, and slowdown [%] wrt. default

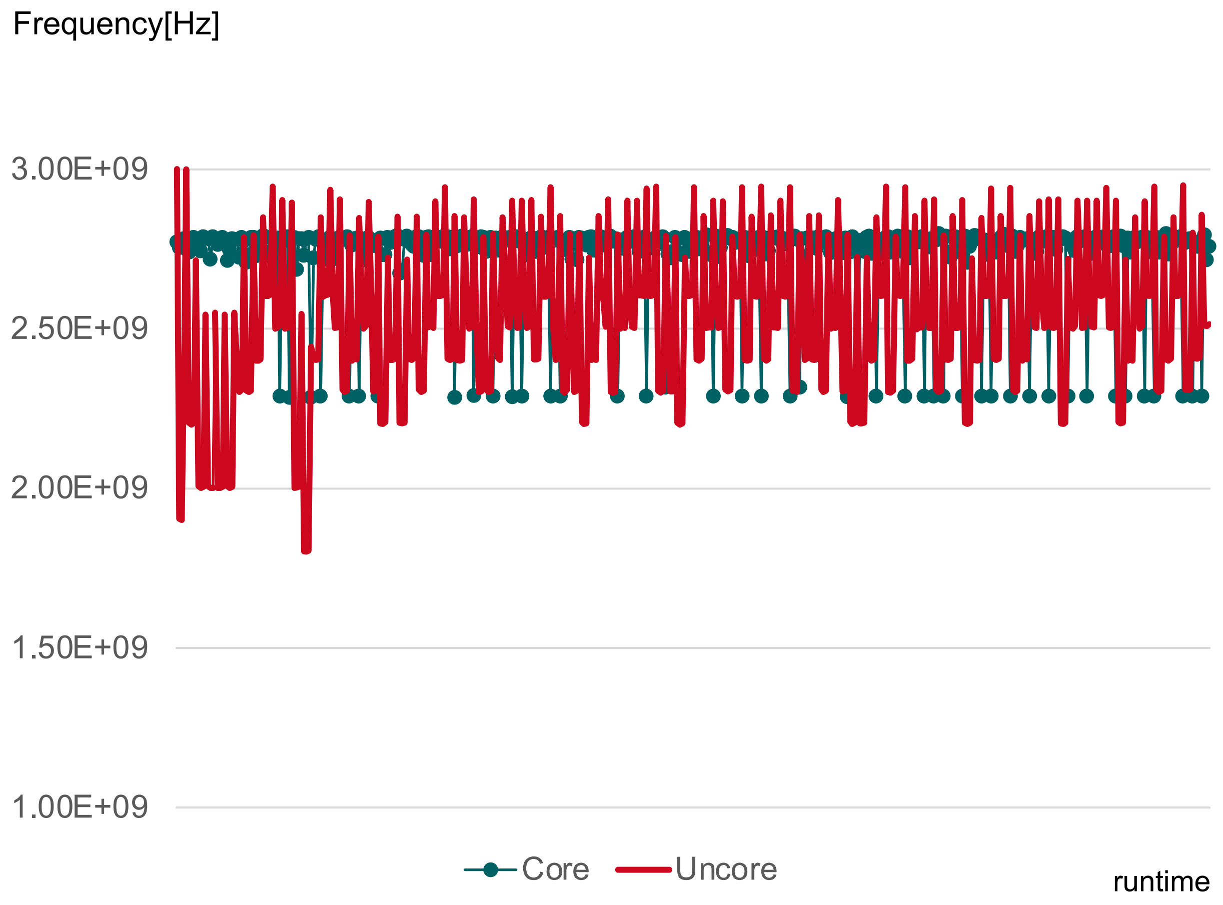
- For DRAM
  - up to 40% of PKG power or energy saving with less than 5% slowdown
- For L3 Cache
  - Marginal impacts on L3 cache



NAS *Parallel Benchmarks*

BT's execution patterns (no power cap)

- Bounds to transfer delay but varies from compute domination to memory domination
- Core frequency: 2.3 ~ 2.8 GHz
- Uncore frequency : 2.2 ~ 3.0 GHz



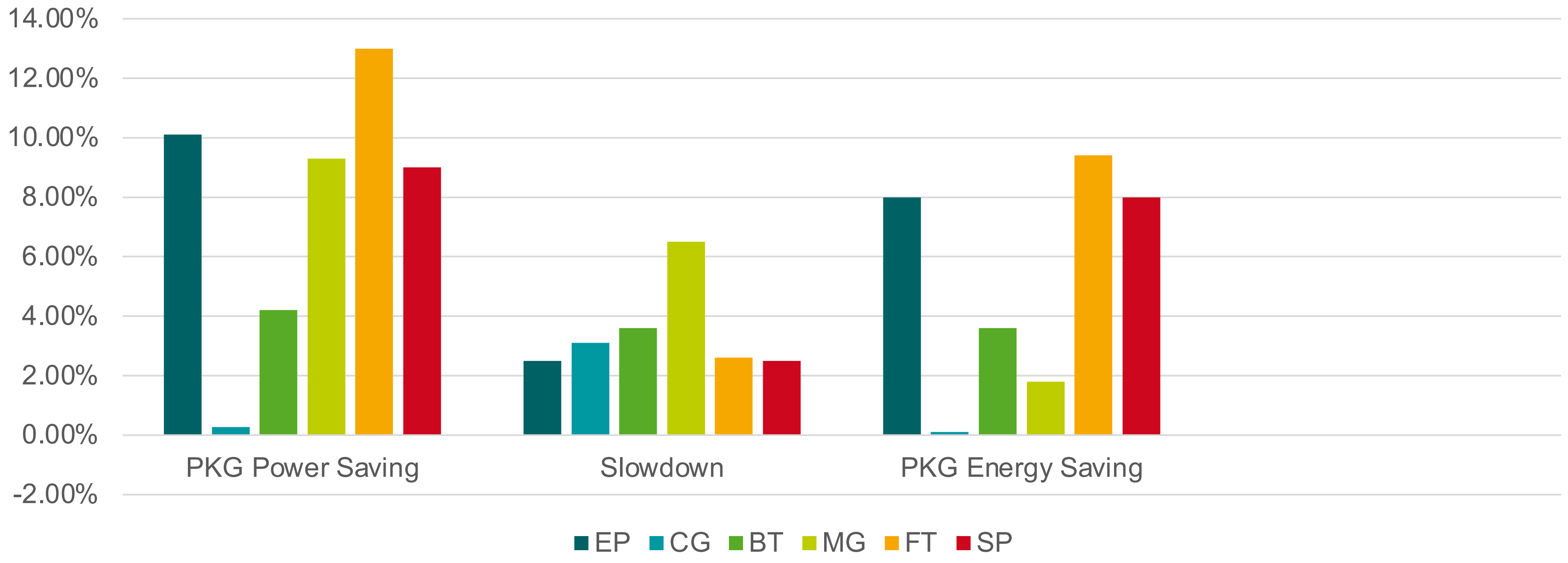
NAS *Parallel Benchmarks*

PKG power saving, slowdown and PKG energy saving [%] wrt. default (no power cap)

Up to 13% PKG power saving

Slowdown under 4% for most benchmarks
Worst case 6.5%

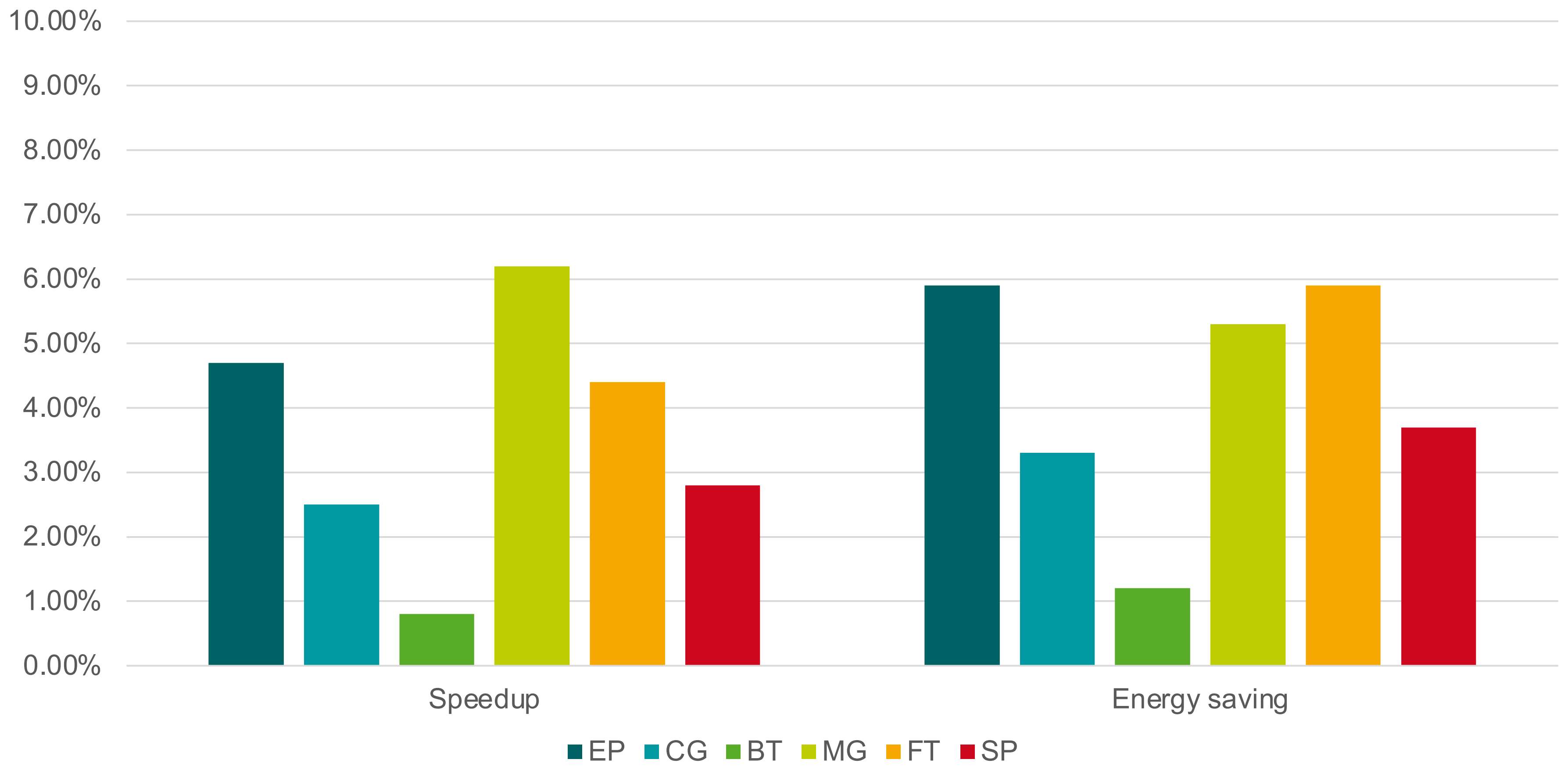
Up to 9.4% energy saving



NAS *Parallel Benchmarks*

Speed up and energy saving [%] wrt. default under a power cap:69W

- Up to 6.2% speedup
- Up to 5.9% energy saving



Summary

- Implement the Roofline model based on the sampling approach

- Extend the Roofline model with the time inteval analysis method
  - Charachterize different performance bottlenecks: compute-, memory- or transfer delay-bound

- The Extended Roofline achieves:
  - Without power cap:
    - Up to 13% package power saving and 9.4% energy saving
    - 2.5% ~ 6.5% slowdown

  - Under power cap:
    - Up to 6.2% speedup and 5.9% energy saving

- *Future Works:*
  - *Sampling by instructions*
    - *Sampling by instructions* triggers the interrupts only if there are program activities
    - More accurate and less overhead

Thanks for your attention

Citation

- [1] https://www.top500.org/lists/top500/
- [2] https://techwireasia.com/2020/06/japans-fugaku-is-the-worlds-fastest-supercomputer/
- [3] https://crd.lbl.gov/assets/Uploads/ECP18-Roofline-1-intro.pdf