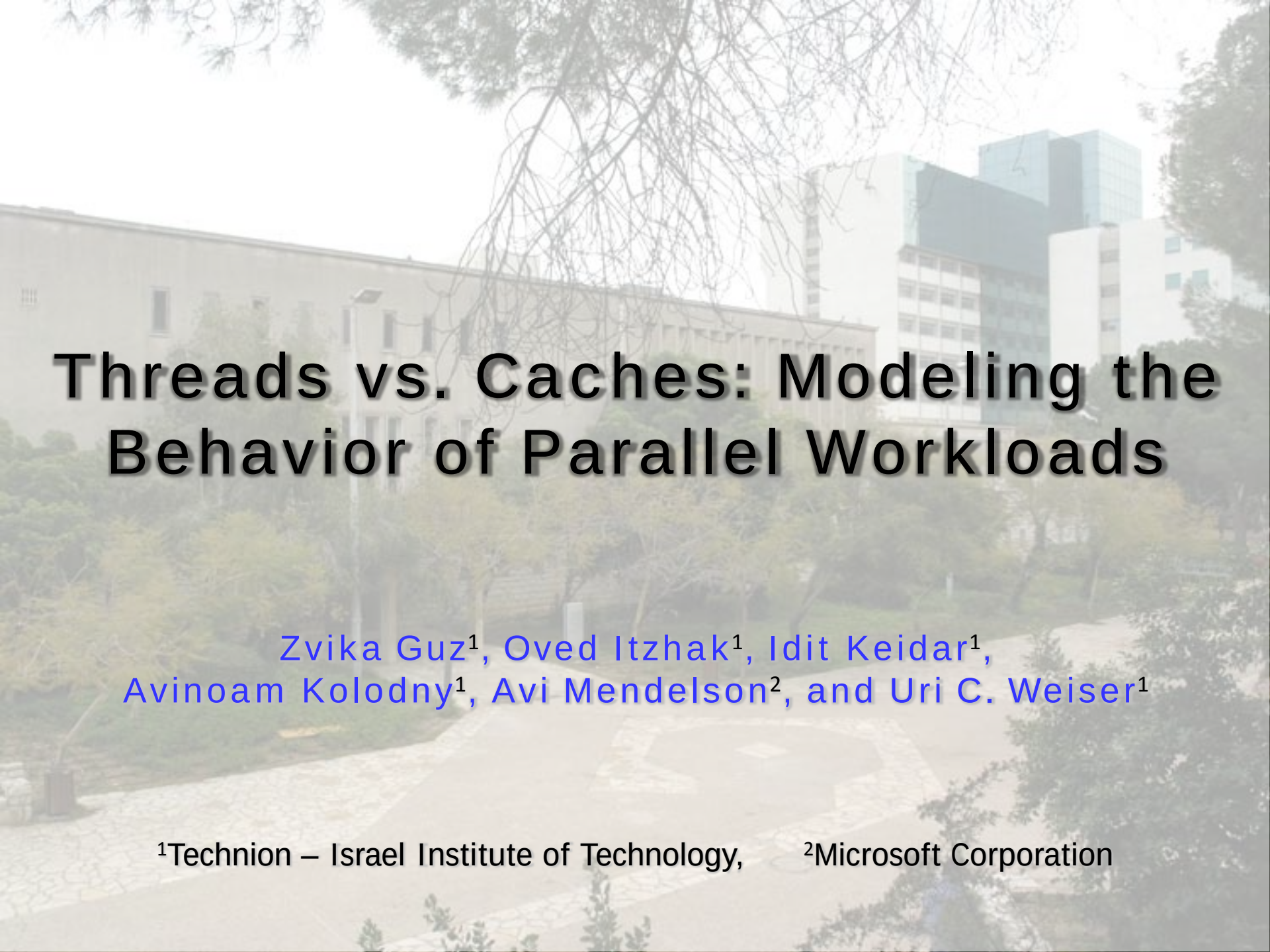


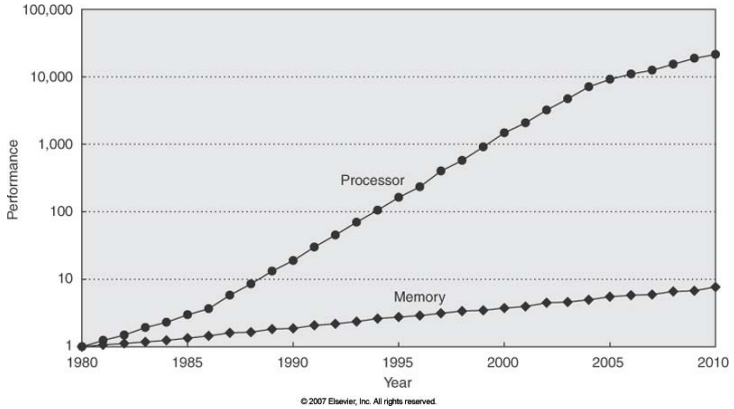
Threads vs. Caches: Modeling the Behavior of Parallel Workloads

Zvika Guz[1], Oved Itzhak[1], Idit Keidar[1],
Avinoam Kolodny[1], Avi Mendelson[2], and Uri C. Weiser[1]

[1]Technion – Israel Institute of Technology, [2]Microsoft Corporation

Chip-Multiprocessor Era

- Challenges:
  - Single-core performance trend is gloomy
    - ➔ Exploit chip-multiprocessors with multithreaded applications
  - The memory gap is paramount
    - Latency, bandwidth, power



- Two basic remedies:
  - Cache – Reduce the number of out-of-die memory accesses
  - Multi-threading – Hide memory accesses behind threads execution
- How do they play together?
- How do we make the most out of them?

[Figure: Hennessy and Patterson, Computer Architecture- A Quantitative approach]

Outline

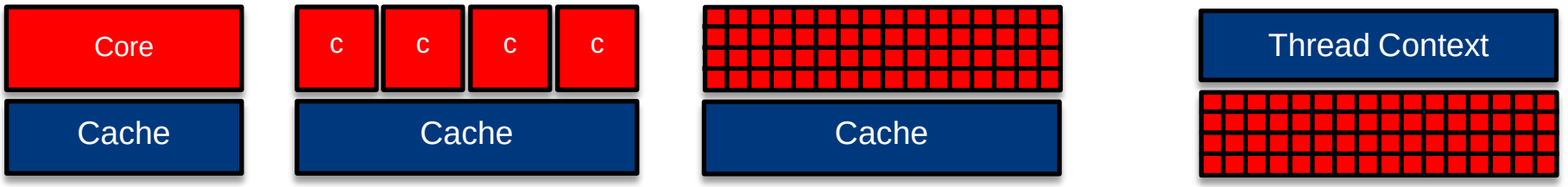
- The many-core span
  - Cache-Machines ↔ MT-Machines
- A high-level analytical model
- Performance curves study
  - Few examples
- Summary

Outline

- **The many-core span**
  - **Cache-Machines ↔ MT-Machines**
- A high-level analytical model
- Performance curves study
  - Few examples
- Summary

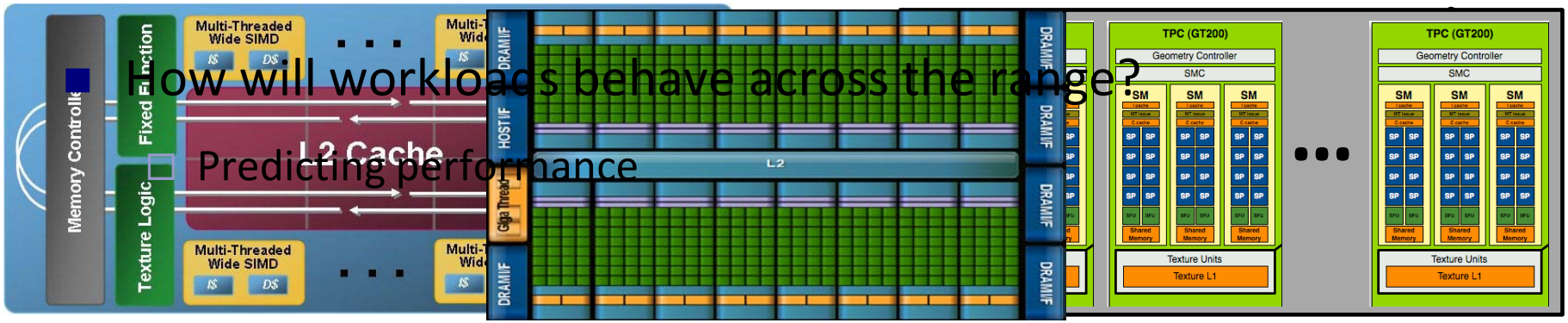
Cache-Machines vs. MT-Machines

- Many-Core – CMP with many, simple cores
  - Tens → hundreds of Processing Elements (PEs)



Uni-Processor Region | **Multi-Core Region** | **Cache Architecture Region** | **MT Architecture Region**

Intel's Larrabee | Nvidia's Fermi | Nvidia's GT200 | **# of Threads**



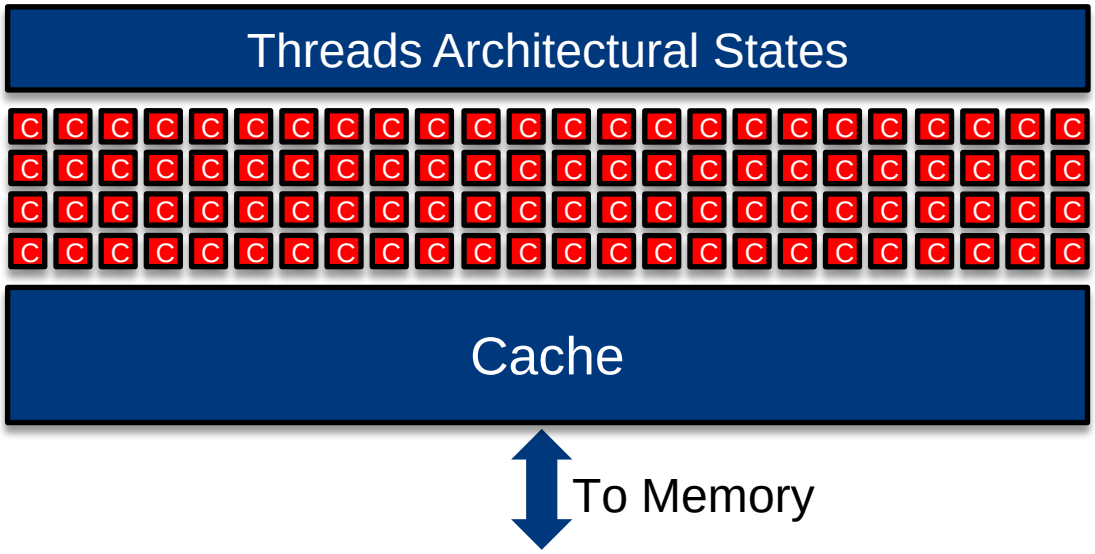
- What are the basic tradeoffs?
- How will workloads behave across the range?
  - Predicting performance

Outline

- The many-core span
  - Cache-Machines ↔ MT-Machines
- **A high-level analytical model**
- Performance curves study
  - Few examples
- Summary

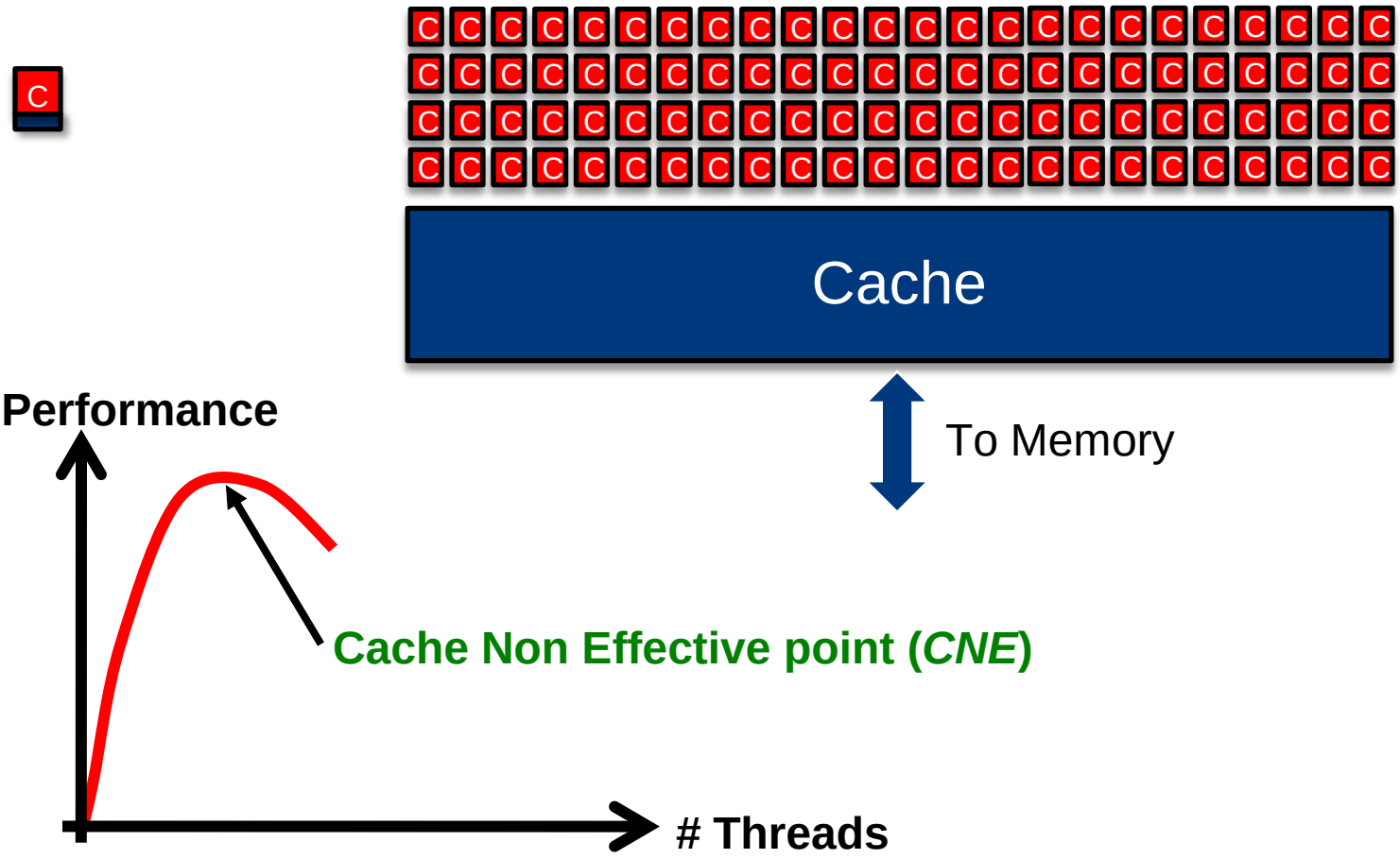
A Unified Machine Model

- Use both cache and many threads to shield memory access
  - The uniform framework renders the comparison meaningful
  - We derive simple, parameterized equations for performance, power, BW,..



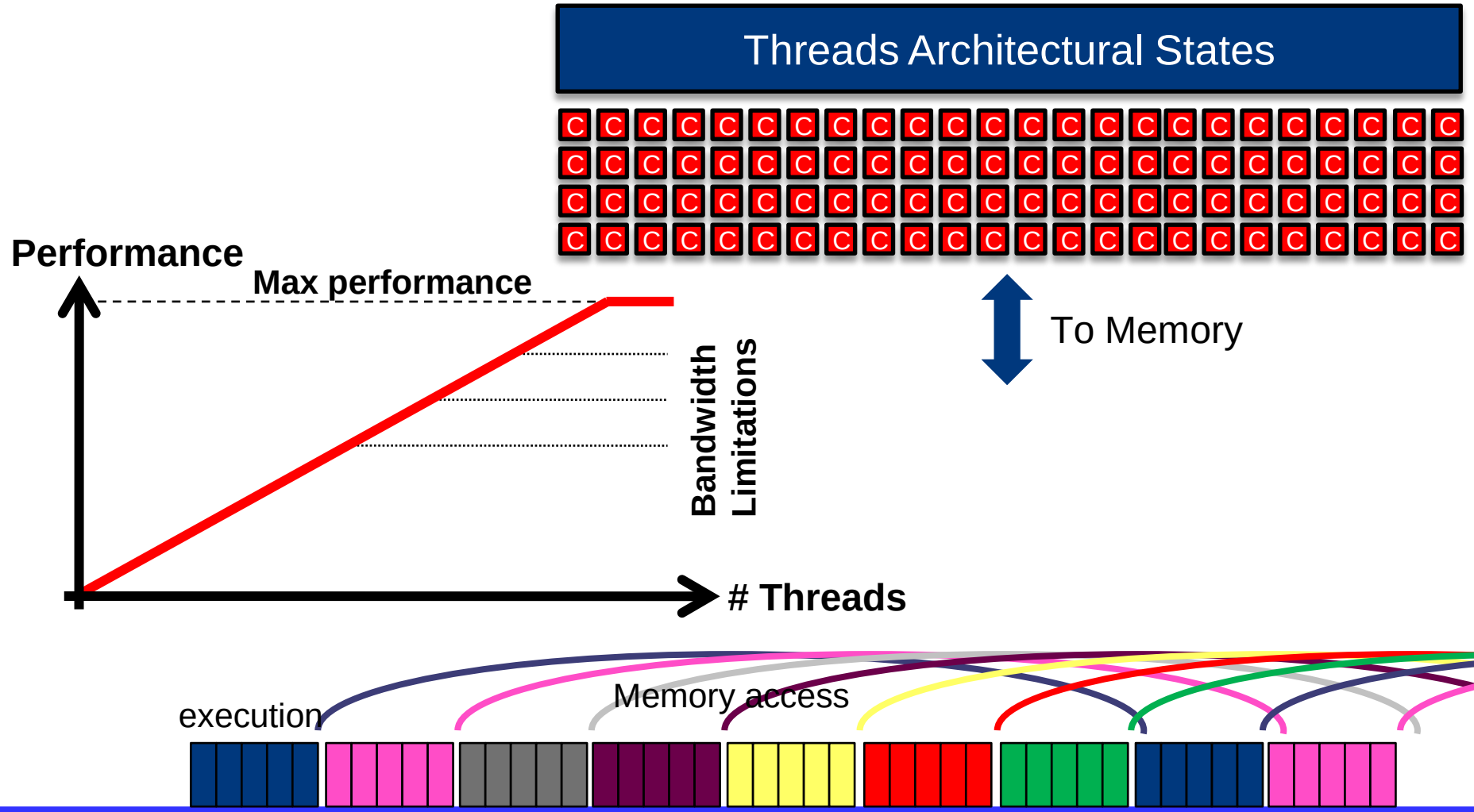
Cache Machines

- Many cores (each may have its private L1) behind a shared cache



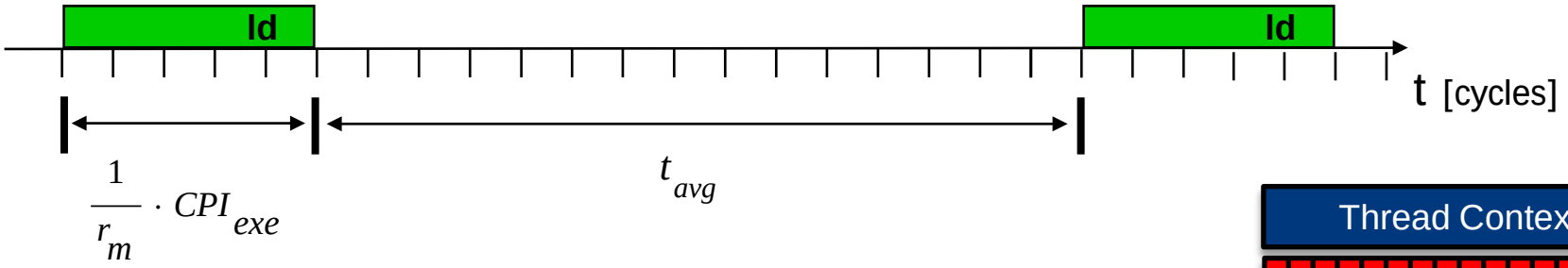
Multi-Thread Machines

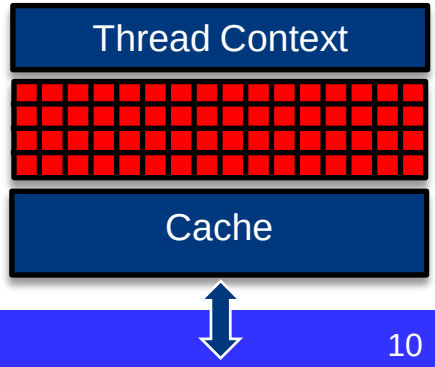
- Memory latency shielded by multiple thread execution



Analysis (1/3)

- Given a ratio of memory access instructions r_m ($0 \leq r_m \leq 1$)
- Every $1/r_m$ instruction accesses memory
  - A thread executes $1/r_m$ instructions
  - Then stalls for t_{avg} cycles
    - t_{avg}=*Average Memory Access Time* (*AMAT*) [cycles]

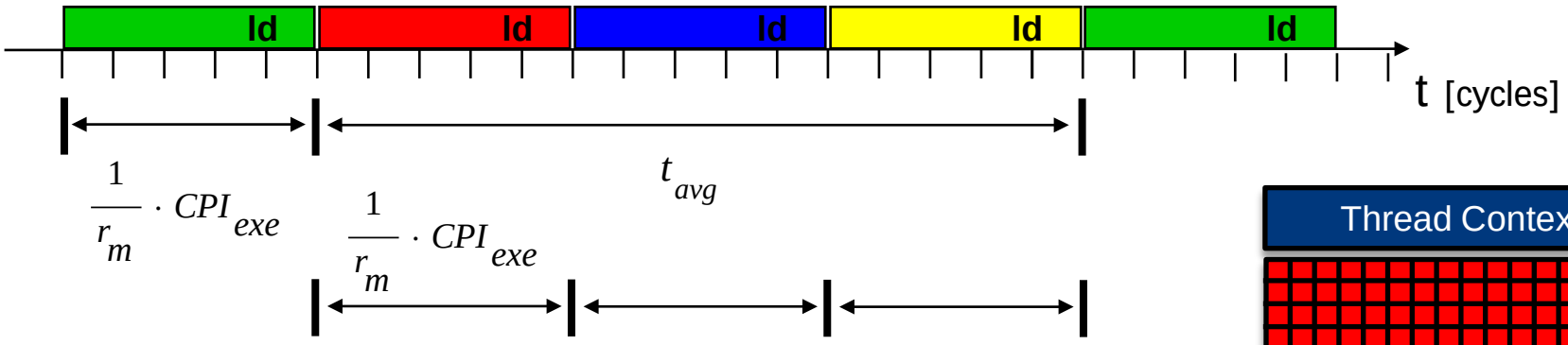


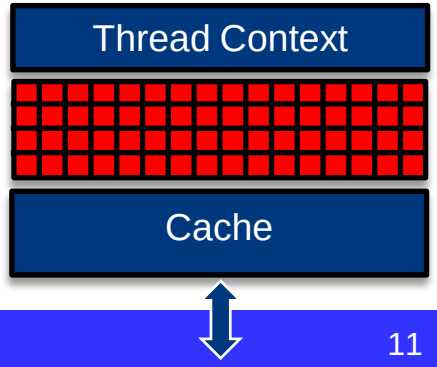


Analysis (2/3)

- PE stays idle unless filled with instructions from other threads
- Each thread occupies the PE for additional $\frac{1}{r_m} \cdot CPI_{exe}$ cycles

➔ $1 + \frac{t_{avg}}{\left(\frac{CPI_{exe}}{r_m}\right)}$ threads needed to fully utilize each PE





Analysis (3/3)

- Machine utilization:

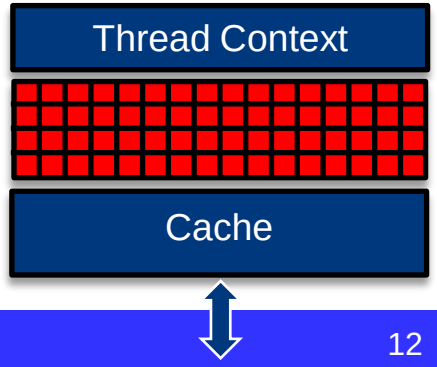
$$\eta = \min\left(1, \frac{n_{threads}}{N_{PE} \cdot \left(1 + t_{avg} \cdot \frac{r_m}{CPI_{exe}}\right)}\right)$$

Number of available threads

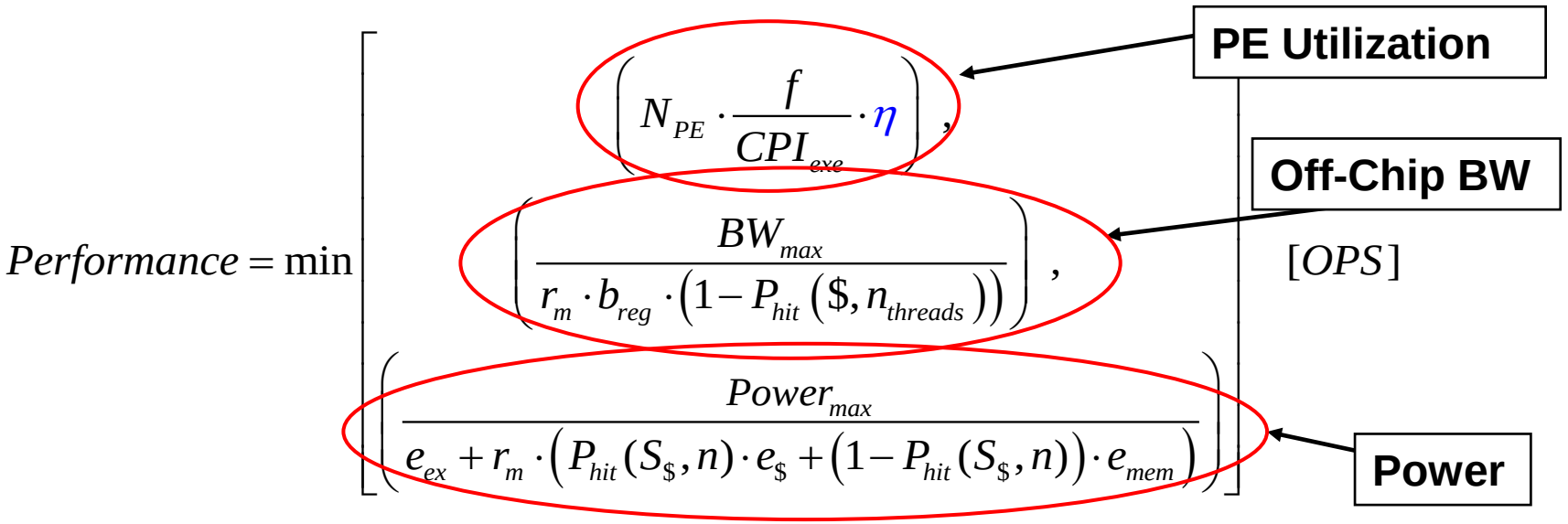
#Threads needed to utilize a single PE

- Performance in Operations Per Seconds [OPS]:

$$Performance = \underbrace{N_{PE} \cdot \frac{f}{CPI_{exe}} \cdot \eta}_{\text{Peak Performance}} \quad [OPS]$$



Performance Model



$$Machine\ Utilization = \eta = \min\left[1\ ,\ \frac{n_{threads}}{N_{PE} \cdot \left(1 + t_{avg} \cdot \frac{r_m}{CPI_{exe}}\right)}\right]$$

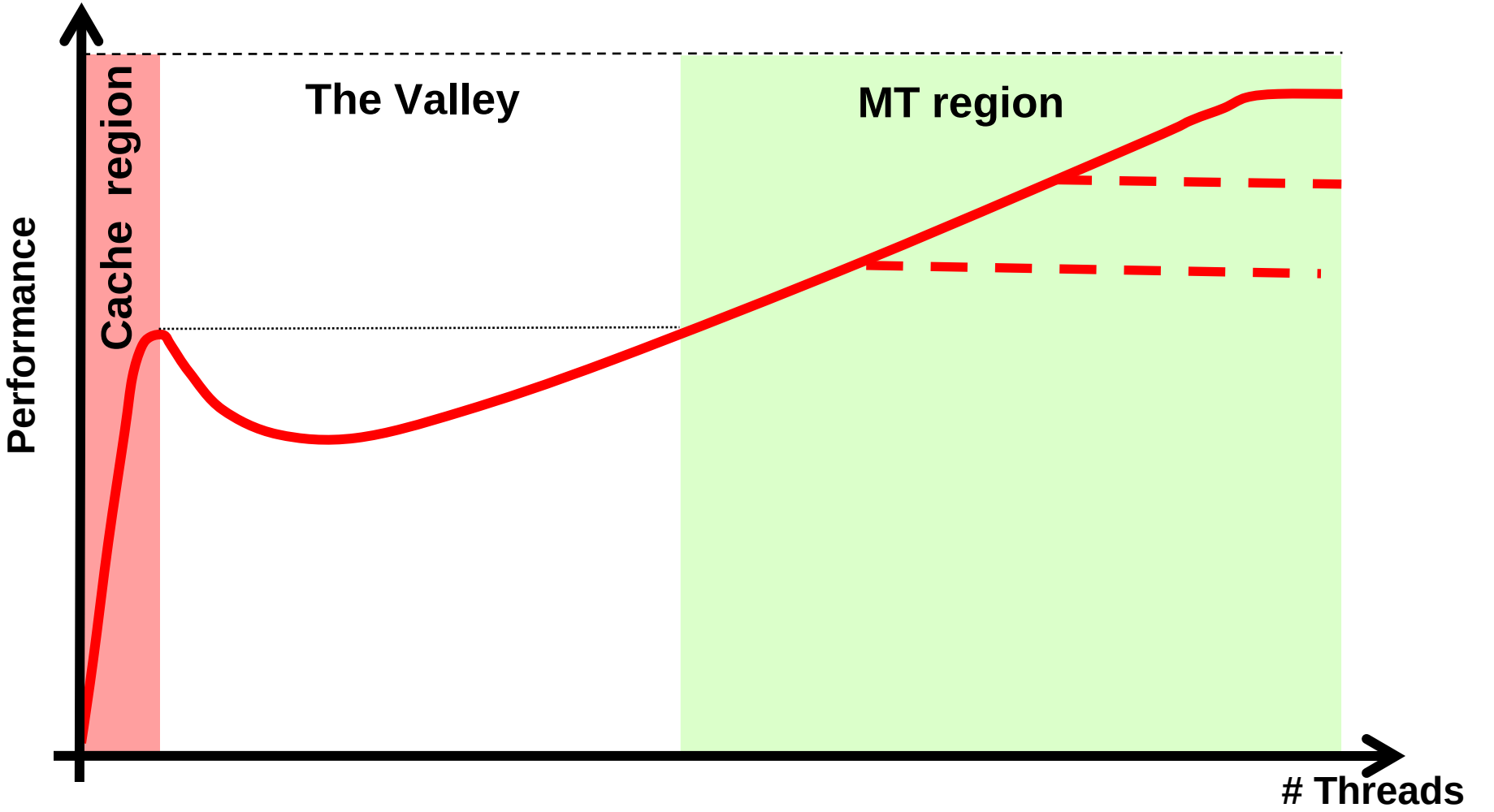
$$t_{avg} = AMAT = P_{hit}\left(\$, n_{threads}\right) \cdot t_\$ + \left(1 - P_{hit}\left(\$, n_{threads}\right)\right) \cdot t_m \quad [cycles]$$

Outline

- The many-core span
  - Cache-Machines ↔ MT-Machines
- A high-level analytical model
- **Performance curves study**
  - **Few examples**
- Summary

Unified Machine Performance

- 3 regions: Cache efficiency region, The Valley, MT efficiency region



HW/SW Assumptions

Workloads:

- Can be parallelized into large number of threads
- No serial part
- Threads are independent of each other
  - No wait time/synchronization
- No data sharing:
  - Cache capacity divided among all running threads
  - Cache hit rate function:

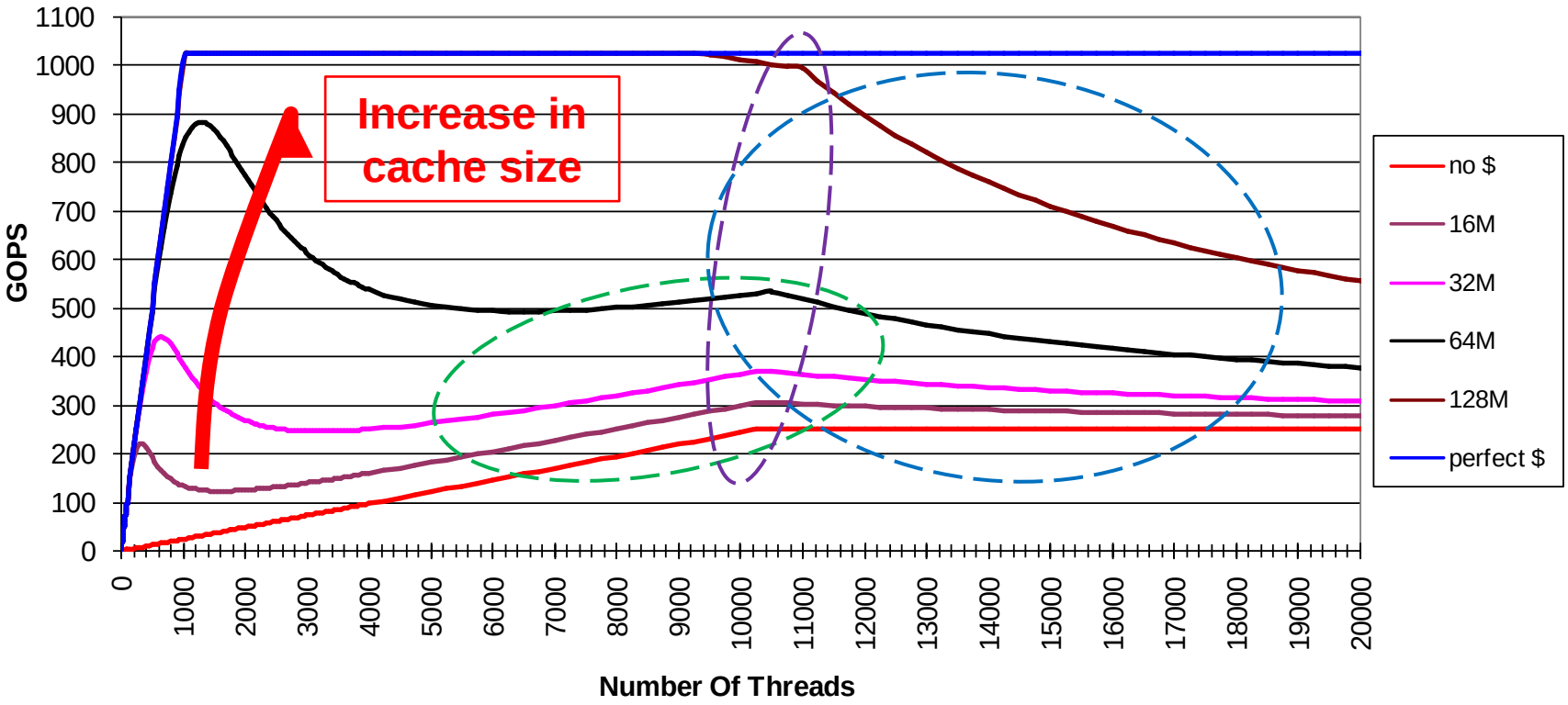
$$P_{hit} = 1 - \left(\frac{S_{\$}}{n_{threads} \cdot \beta} + 1 \right)^{-(\alpha - 1)} \quad : \alpha > 1, \beta > 0$$

Hardware:

Parameter	Value
N_{PE}	1024
$S_{\$}$	16 MByte
CPI_{exe}	1
f	1 GHz
t_m	200 cycles
r_m	0.2

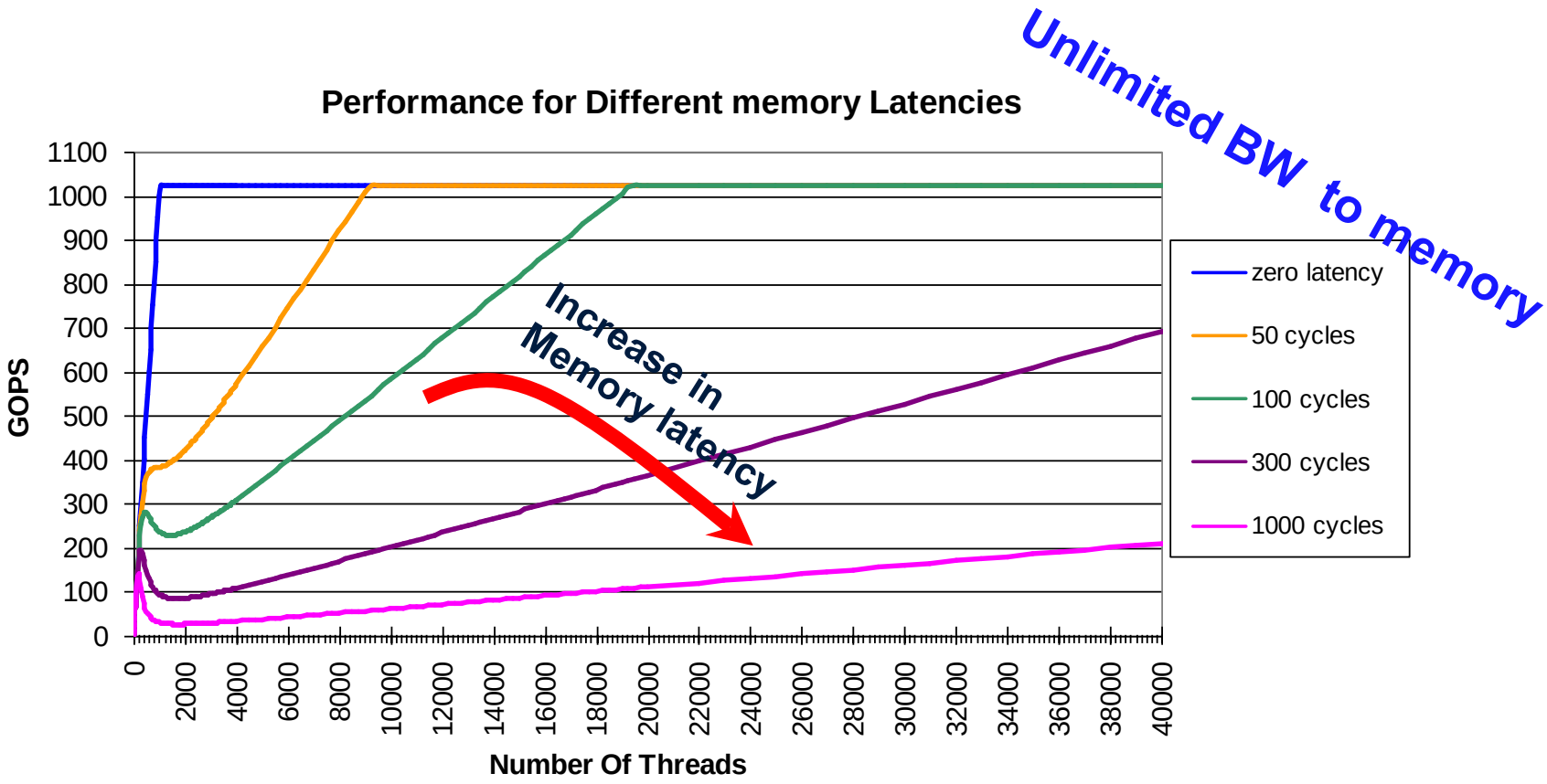
Cache Size Impact

- Increase in cache size→ cache suffices for more in-flight threads → Extends the $ region ..AND also→ Valuable in the MT region
  - Caches reduce off-chip bandwidth→ delay the BW saturation point



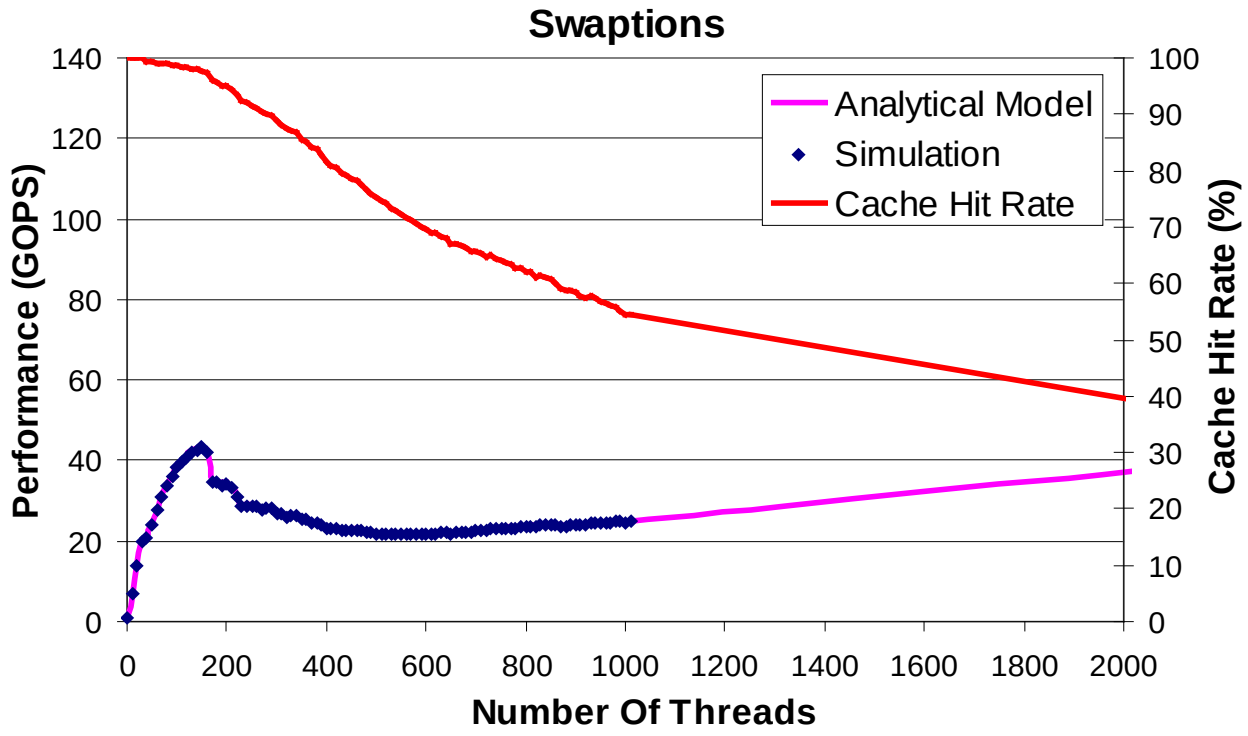
Memory Latency Impact

- Increase in memory latency ➔ Hinders the MT region
  ➔ Emphasise the importance of caches



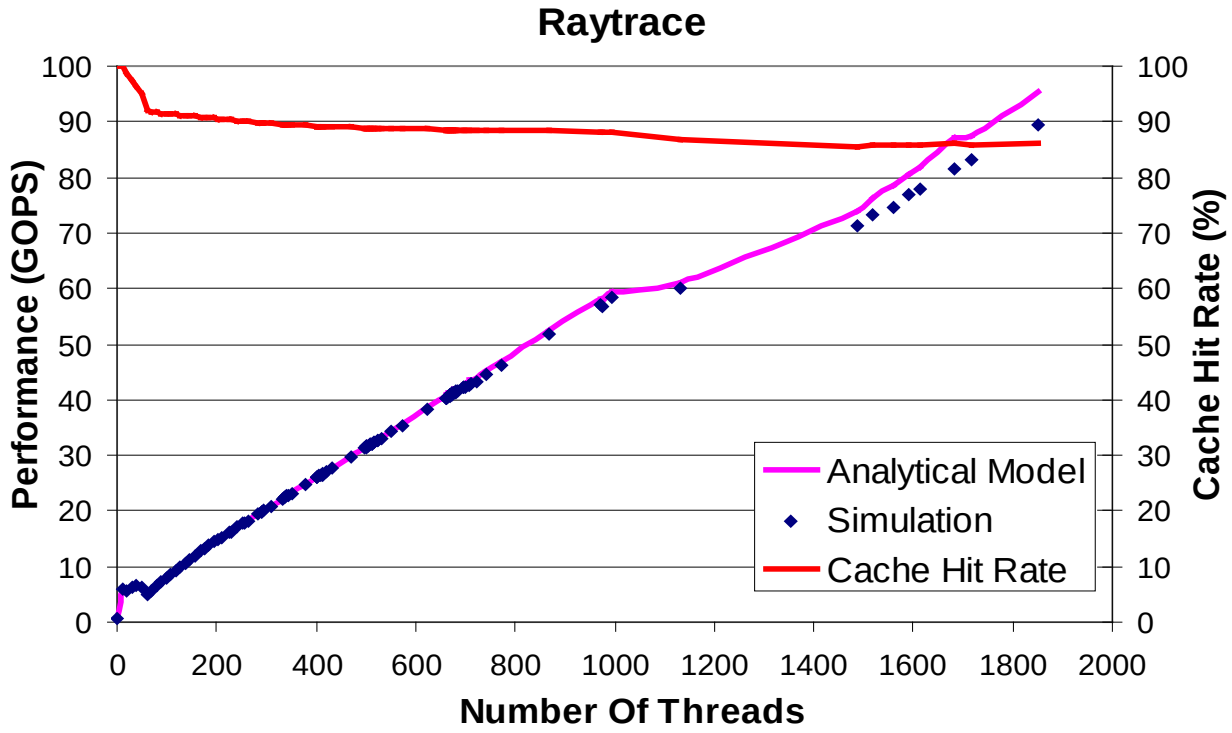
Hit Rate Function Impact

- Simulation results from the PARSEC workloads kit
- ***Swaptions***:
  - Perfect Valley



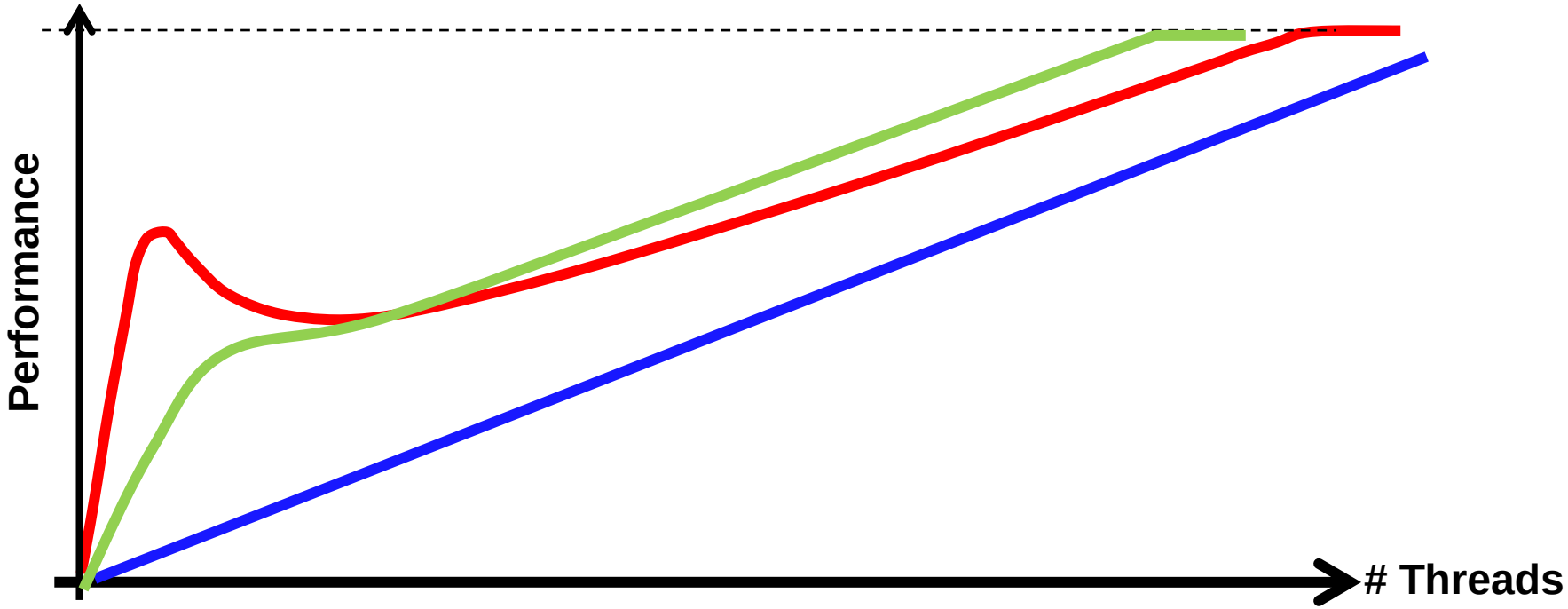
Hit Rate Function Impact

- Simulation results from the PARSEC workloads kit
- ***Raytrace***:
  - Monotonically-increasing performance



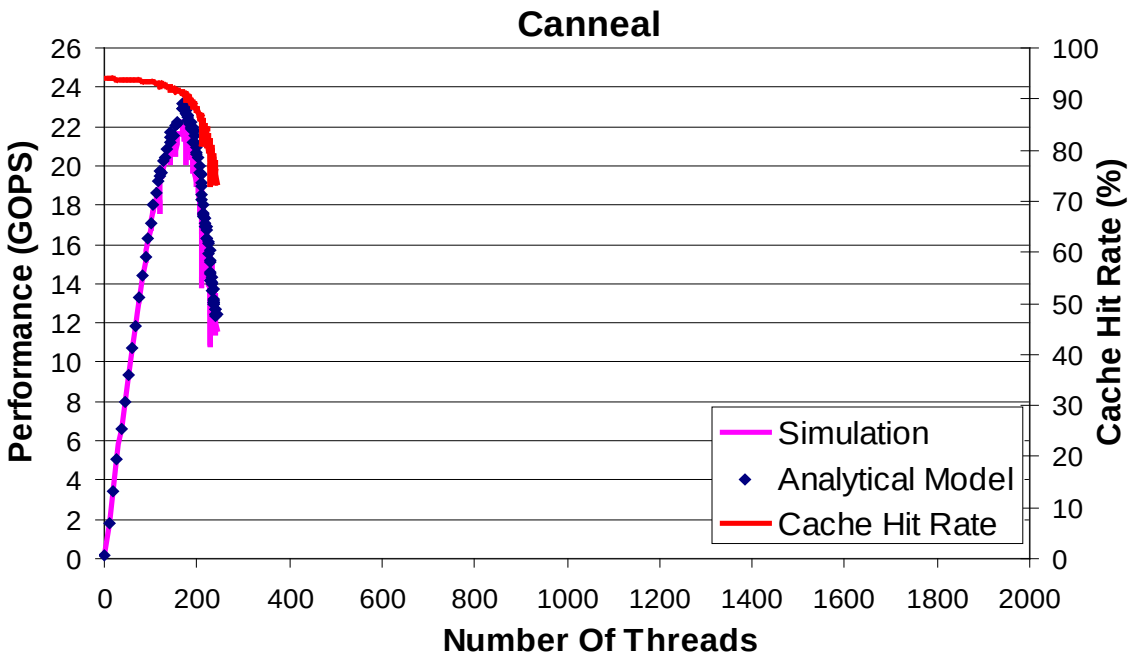
Hit Rate Dependency – 3 Classes

- Three applications families based on cache miss rate dependency:
  - A “strong” function of number of threads – $f(N^q)$ when $q>1$
  - A “weak” function of number of threads - $f(N^q)$ when $q\leq 1$
  - Not a function of number of threads



Workload Parallelism Impact

- Simulation results from the PARSEC workloads kit
- ***Canneal***
  - Not enough parallelism available



Outline

- The many-core span
  - Cache-Machines ↔ MT-Machines
- A high-level analytical model
- Performance curves study
  - Few examples
- **Summary**

Summary

- A high-level model for many-core engines
  - A unified framework for machines and workloads from across the range
  - Validated by simulations with PARSEC workloads
- A vehicle to derive intuition
  - Qualitative study of the tradeoffs
  - A tool to understand parameters impact
  - Identifies new behaviors and the applications that exhibit them
  - Enables reasoning of complex phenomena
- First step towards escaping the valley
  - Current work: architectural mechanisms to bridge the valley

Thank You!

Backup

Model Parameters

Model Parameters

- Machine parameters:

Parameter	Description
N_{PE}	Number of PEs (in-order processing elements)
$S_{\$}$	Cache size *[Bytes]*
N_{max}	Maximal number of thread contexts in the register file
CPI_{exe}	Average number of cycles required to execute an instruction assuming a perfect (zero-latency) memory system *[cycles]*
f	Processor frequency *[Hz]*
$t_{\$}$	Cache latency *[cycles]*
t_m	Memory latency *[cycles]*
BW_{max}	Maximal off-chip bandwidth *[GB/sec]*
b_{reg}	Operands size *[Bytes]*

Model Parameters

- Workload parameters:

Parameter	Description
n	Number of threads that execute or are in ready state (not blocked) concurrently
r_m	Fraction of instructions accessing memory out of the total number of instructions [$0 \leq r_m \leq 1$]
$P_{hit}(s, n)$	Cache hit rate for each thread, when n threads are using a cache of size s

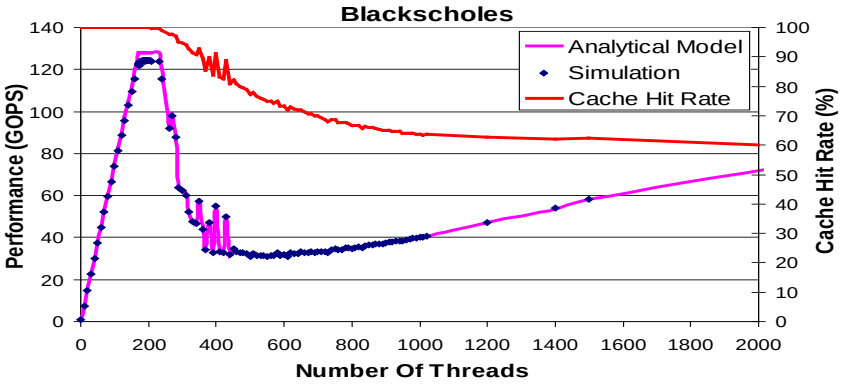
Model Parameters

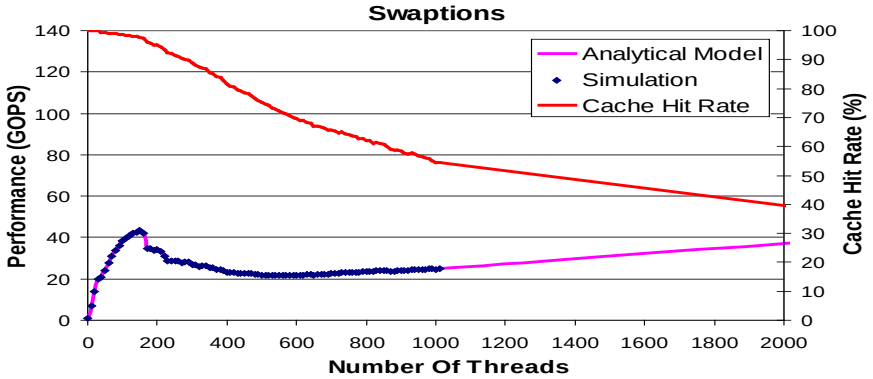
- Power parameters:

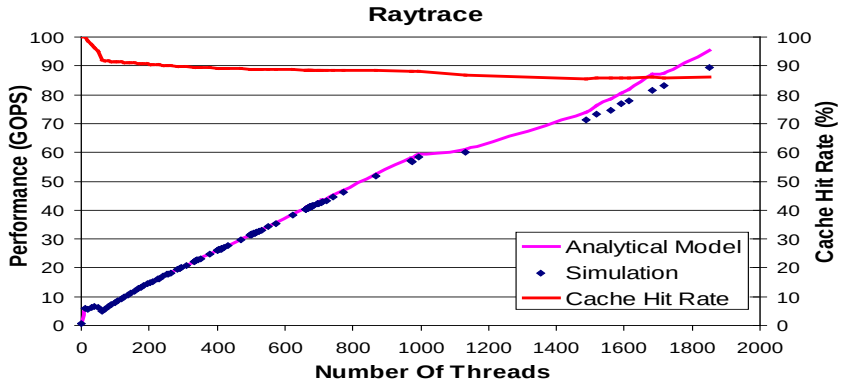
Parameter	Description
e_{ex}	Energy per operation *[j]*
$e_{\$}$	Energy per cache access *[j]*
e_{mem}	Energy per memory access *[j]*
$Power_{leakage}$	Leakage power *[W]*

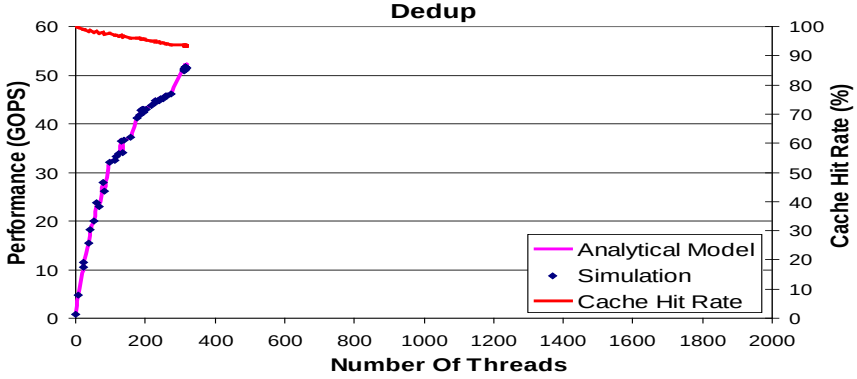
Parsec Workloads

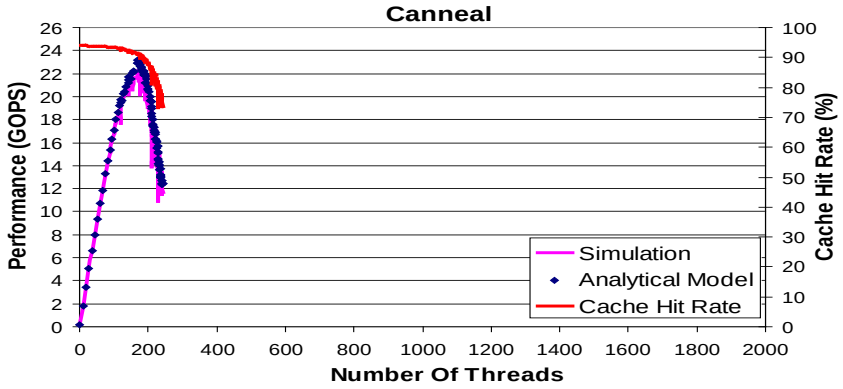
Model Validation, PARSEC Workloads

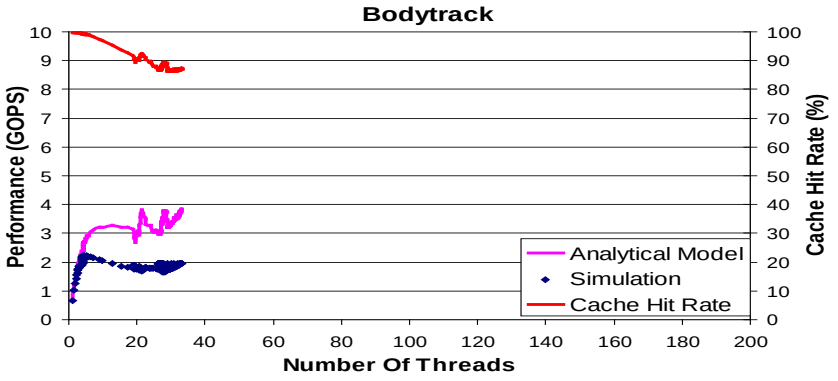






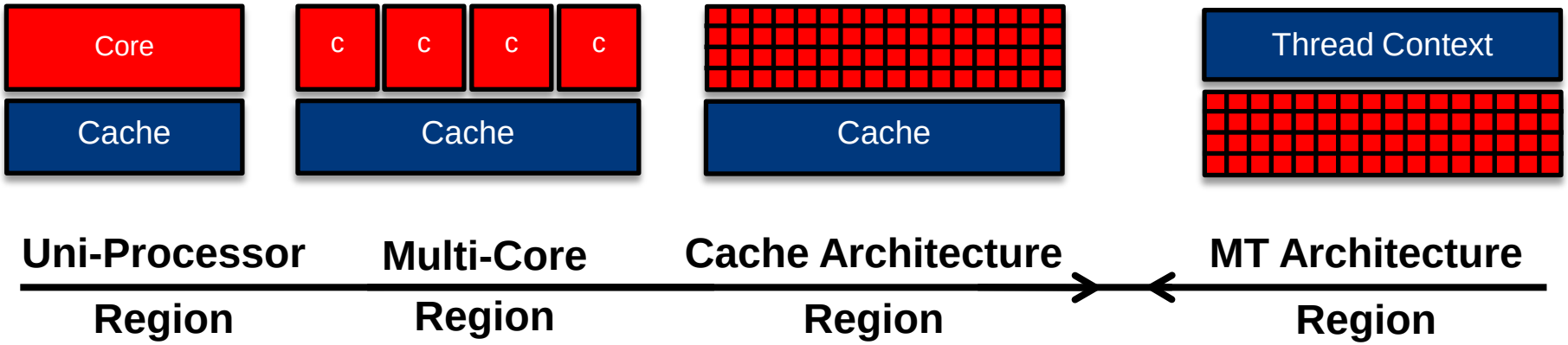






Related Work

Related Work



- Similar approach of using high level models:
  - Agarwal, TPDS-1992
  - Saavedra-Barrera and Culler, Berkeley 1991
  - Sorin et al., ISCA-1998
  - Marculescu et al., CA Letters 2005
  - Hill and Michael, IEEE Computer 2008
  - Eyerman and Eeckhout, ISCA-2010
  - Hong and Kim, ISCA-2009
  - Baghsorkhi et al., PPoPP-2010