Efficient Runahead Threads


Tanausú Ramírez


Alex Pajuelo


Oliverio J. Santana


Onur Mutlu


Mateo Valero

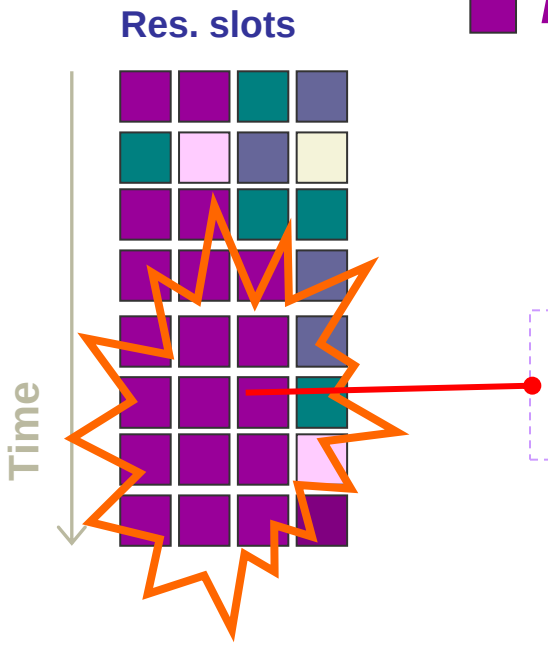
The Nineteenth International Conference on

Parallel Architectures and Compilation Techniques (PACT)

11-15 September, Vienna, Austria

Introduction

- Simultaneous Multithreaded (SMT) processors
  - execute multiple instructions from several threads at the same time
  - resource sharing still hits with the memory wall



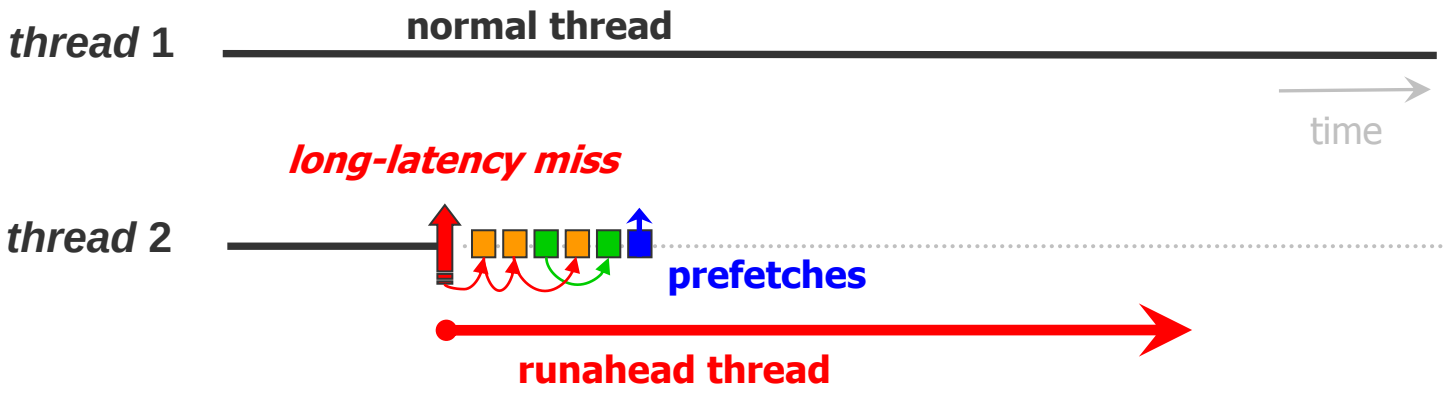
memory-intensive thread

- streams of long latency memory accesses
- holds too many resources

- starves other threads of available resources
- prevents their forward progress as well

Runahead Threads (RaT)

- Multithreaded approach of Runahead paradigm
  - interesting way of using resources in SMT in the presence of long-latency memory instructions
    - speculative thread → runahead thread



  - a runahead thread executes speculative instructions while a long-latency load is resolving
    - follow the program path instead of being stalled
    - overlap prefetch accesses with the main load

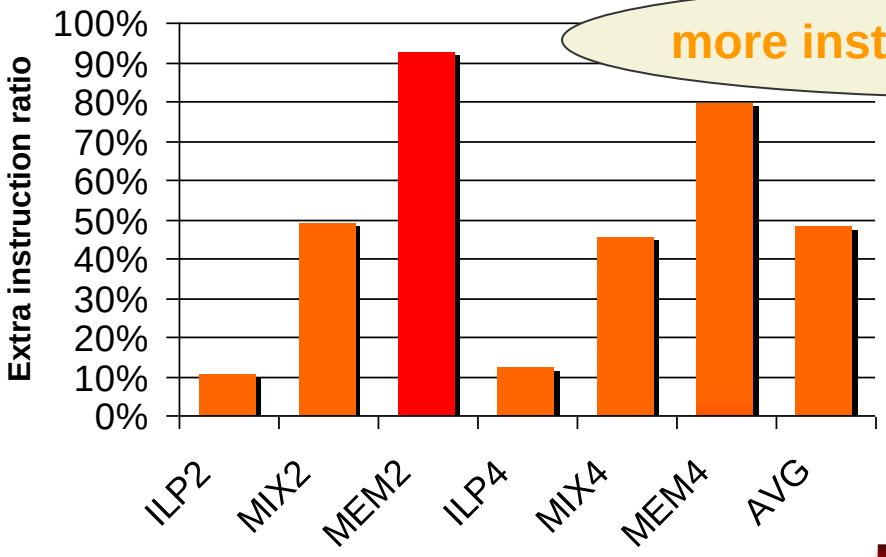
RaT benefits

1. Allow memory-bound threads going speculatively in advance doing **prefetching**
   - provide speedup increasing *memory-level parallelism*
     - improve individual performance of this thread

2. Prevent memory bounded thread **monopolization**
   - transform a resource hoarder thread into a light-consumer thread
     - shared resources can be fairly used by other threads

RaT shortcoming

- RaT causes extra work due to more extra executed instructions



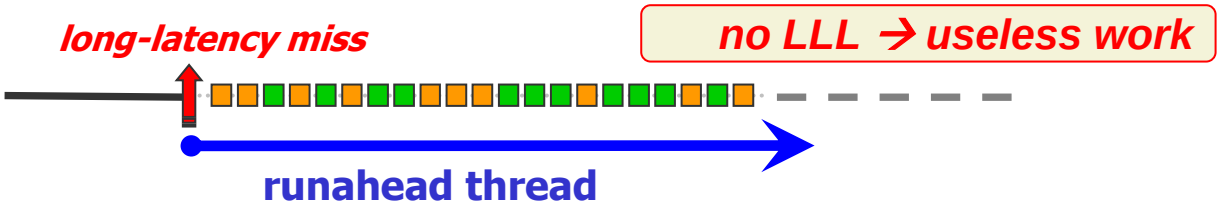
more instructions → more energy

RaT increase ~50% power consumption

Impact on performance-energy efficiency

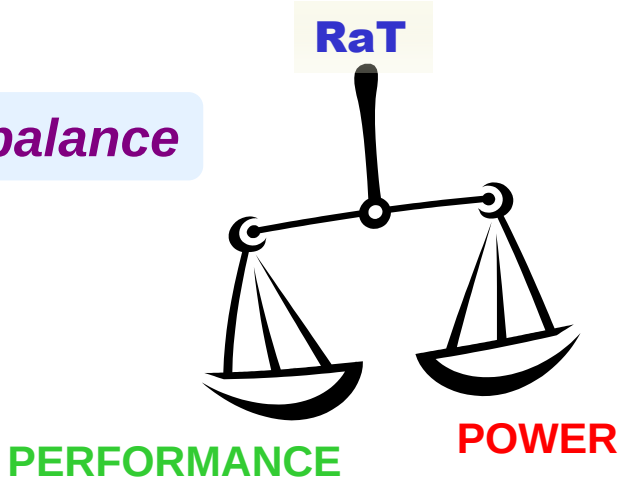
Proposal

- Improving RaT efficiency
  - avoid inefficient cases: no prefetching → useless instructions



  - only useful executions that maximize the performance
    - control the number of *useless* instructions executed by runahead threads

- Our goal → ***energy-performance balance***
  - reduce instruction without losing performance
  - reduce power



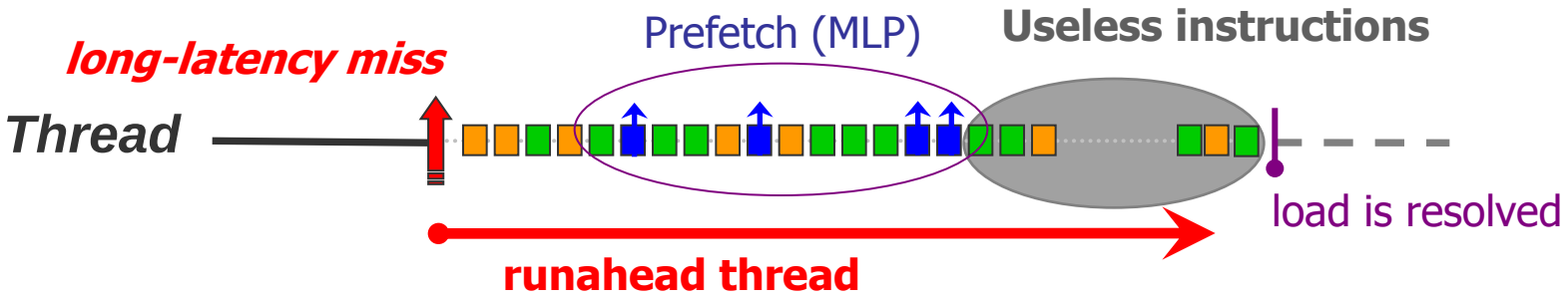
Talk outline

- **Introduction**
  - RaT problem
  - Proposal
- **Efficient Runahead Threads**
  - **Useful Runahead Distance**
  - **Runahead Distance Prediction (RDP)**
- **Evaluation**
  - Framework
  - Results
- **Conclusions**

Our approach

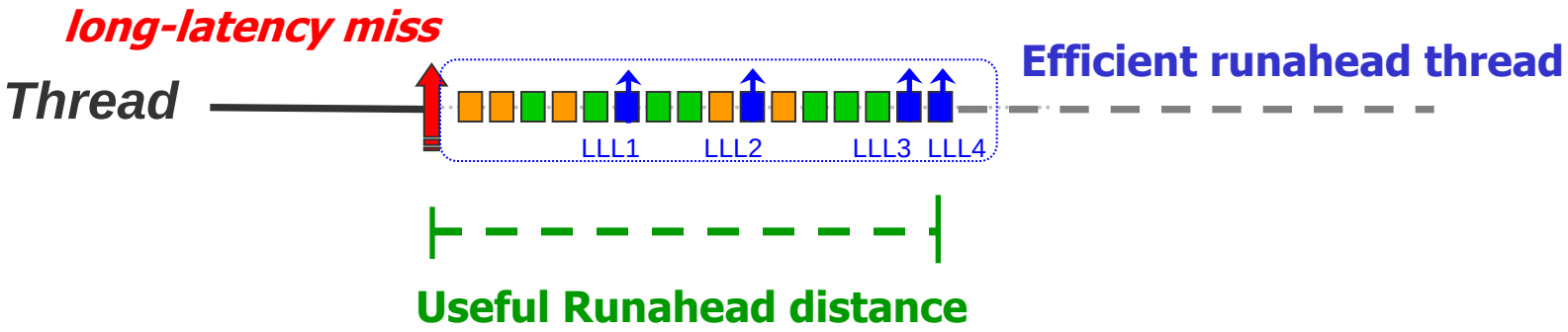
- Propose a mechanism to improve RaT efficiency
  - based on features of own runahead threads

- Idea
  - capture the useful lifetime of a runahead thread
    - to exploit as much MLP as possible
    - with the minimum number of speculative instructions



    - balance between prefetching and useless instructions

Useful Runahead Distance



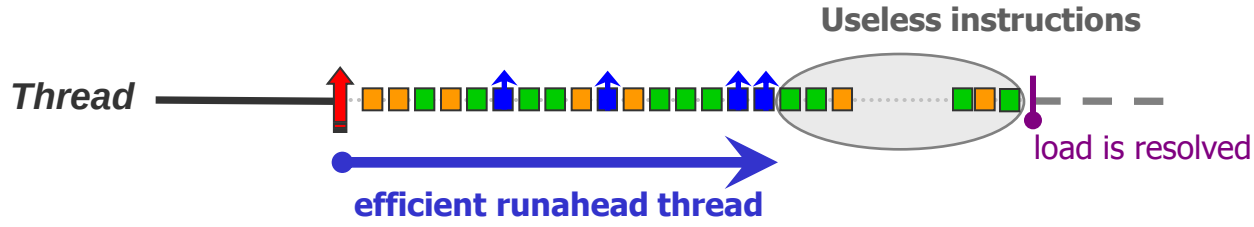
how far thread should run ahead speculatively such that execution will be efficient

- By limiting the runahead execution of a thread, we generate **efficient runahead threads** that
  - avoid unnecesary speculative execution and
  - enhance runahead threads efficiency
    - control the number of useless speculative instructions while obtaining the maximum performance for a given thread

Using useful runahead distance

- With the useful runahead distance we can help the processor to decide
  - **whether or not** a runahead thread should be initiated on an L2-miss load
    - eliminate useless runahead threads

  - indicate **how long** the runahead execution period should be when a runahead thread is initiated
    - reduce unnecessary extra work done by useful runahead threads



Runahead Distance Prediction

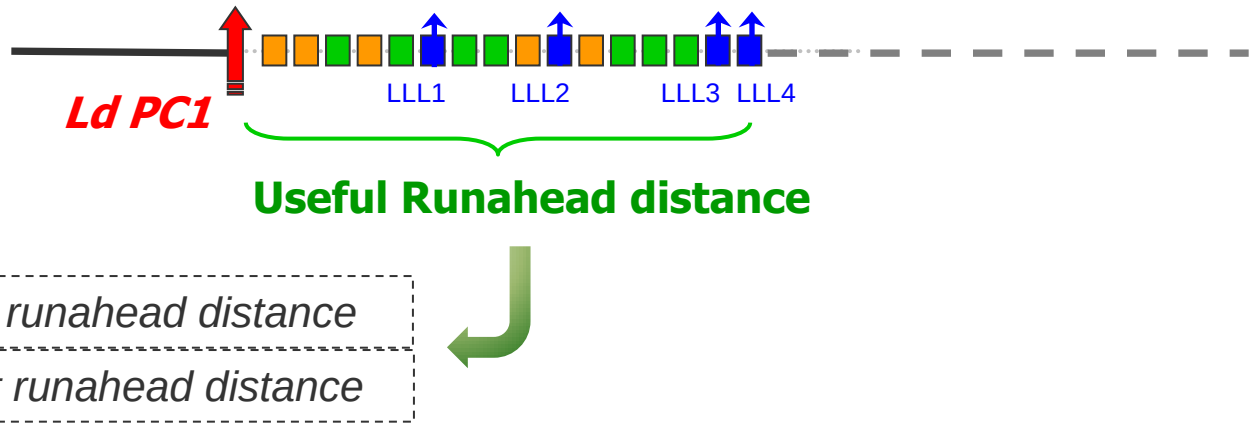
- RDP operation
  - calculate the useful runahead distances of runahead thread executions
  - use that useful runahead distance for predicting future runahead periods caused by the same static load instruction
- RDP implementation
  - use two useful runahead distances (full and last)
    - improve reliability of the useful distance information
  - runahead thread information is associated to its causing long-latency load
    - store this information in a table

Runahead Distance Information Table (RDIT)

tag	thid	last	full	counter
tag	thid	last	full	counter
11bits	2bits	10bits	10bits	2bits

RDP mechanism (I)

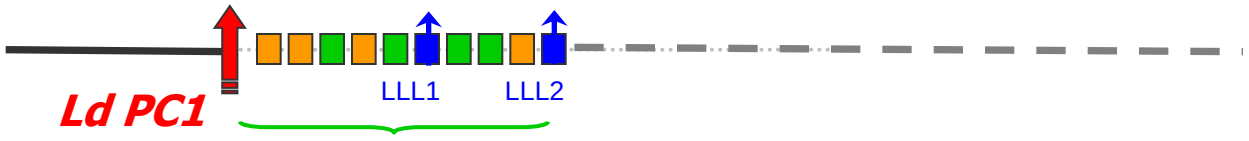
- First time: no useful distance information for a load
  - execute fully a runahead thread and compute the new useful runahead distance



  - both full and last distances are updated with the obtained useful runahead distance

RDP mechanism (II)

- Next times: check whether distances are reliable
  - **No** → when full and last distances are very different
    - based on a threshold: *Distance Difference Threshold*
    - full – last > DDT
      - → execute fully the runahead thread



new Useful Runahead distance

full runahead distance

last runahead distance

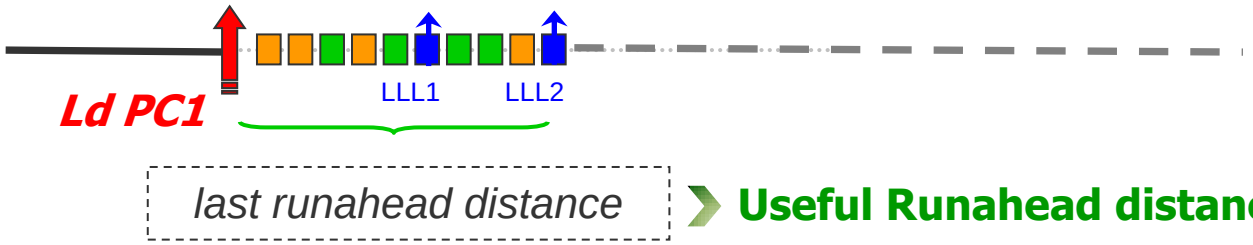
- → and update both (full and last) runahead distances

RDP mechanism (III)

- Next times: check whether distances are reliable
  - **Yes** → full - last <= DDT
  - take the last distance as useful runahead distance

❶ decide to start the runahead thread ?

– if last < activation threshold → do not start runahead thread



❷ how long executes the runahead thread

– else → start the runahead thread executing as many instructions as last useful distance indicates

RDP mechanism (IV)

- Ensuring reliability of small usefull distances
  - some loads have initial small runahead distances…
  - but useful runahead distance becomes larger later
    - 26% of loads increase its useful runahead distance
- Use a simple heuristic
  - provide an additional confidence mechanism that determine whether or not a small distance is reliable
    - while useful runahead distance < activation threshold during N times (N=3) → execute full runahead execution

Talk outline

- **Introduction**
  - RaT problem
  - Proposal
- **Efficient Runahead Threads**
  - Useful Runahead Distance
  - Runahead Distance Prediction (RDP)
- **Evaluation**
  - **Framework**
  - **Results**
- **Conclusions**

Framework

- **Tools**
  - derived SMTSIM exec-driven simulator
  - enhanced Wattch power model integrated
  - CACTI
- **Benchmarks**
  - Multiprogrammed workloads of 2 and 4 threads mix from SPEC 2000 benchmarks
    - ILP,MEM,MIX
- **Methodology**
  - FAME
    - ensure multithreaded state continuously
    - threads fairly represented in the final measurements

Framework

- SMT model with a complete resource sharing organization

Processor core	
Processor depth	10 stages
Processor width	8 way (2 or 4 contexts)
Reorder buffer size	64 entries per context
INT/FP registers	48 regs per context
INT / FP /LS issue queues	64 / 64 / 64
INT / FP /LdSt unit	4 / 4 / 2
Hardware prefetcher	Stream buffers (16)
Memory subsystem	
Icache	64 KB, 4-way, pipelined
Dcache	64 KB, 4-way, 3 cyc latency
L2 Cache	1 MB, 8-way, 20 cyc latency
Caches line size	64 bytes
Main memory latency	300 cycles minimum

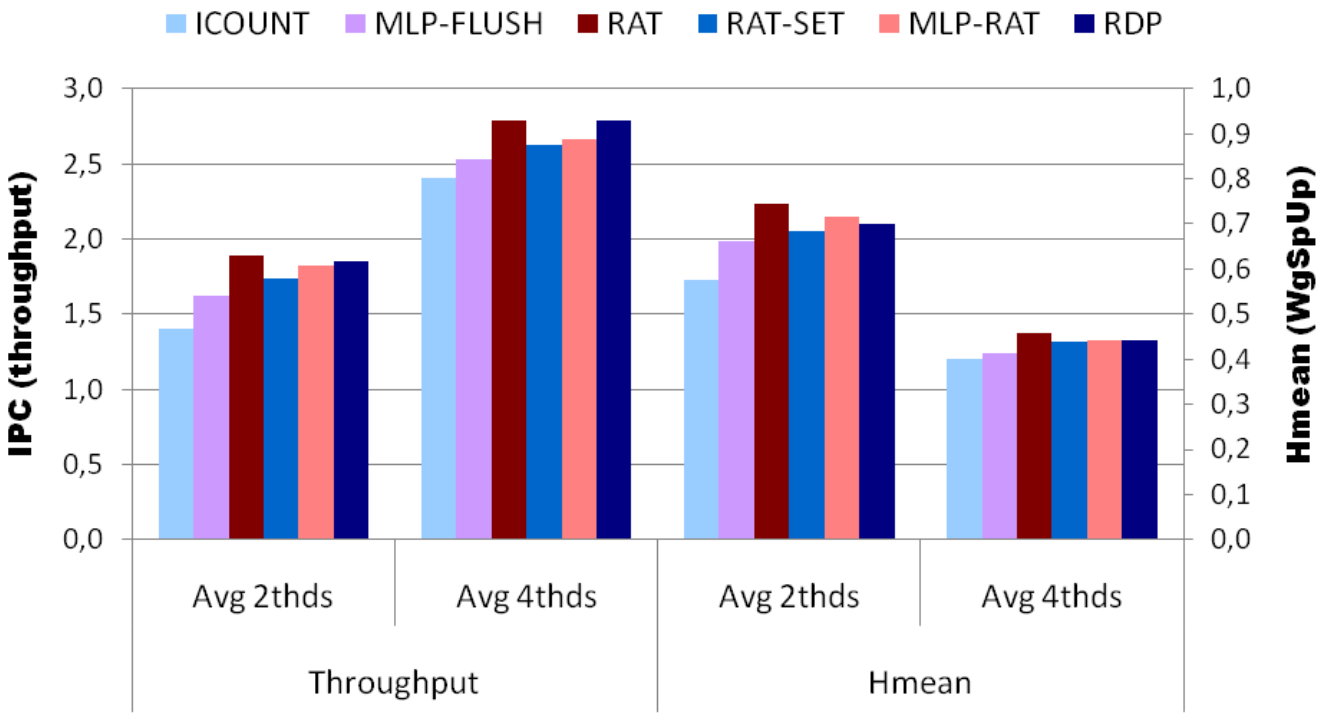
Evaluation

- Performance and energy-efficiency analysis
  - throughput, hmean of thread weighted speedups
  - extra instructions, power, ED^2
- Comparison with state-of-the-art mechanisms
  - MLP-aware Flush (MLP-Flush)
  - MLP-aware Runahead thread (MLP-RaT)
  - RaT + single-thread execution techniques (RaT-SET)
  - Efficient Runahead Threads (RDP)

Performance

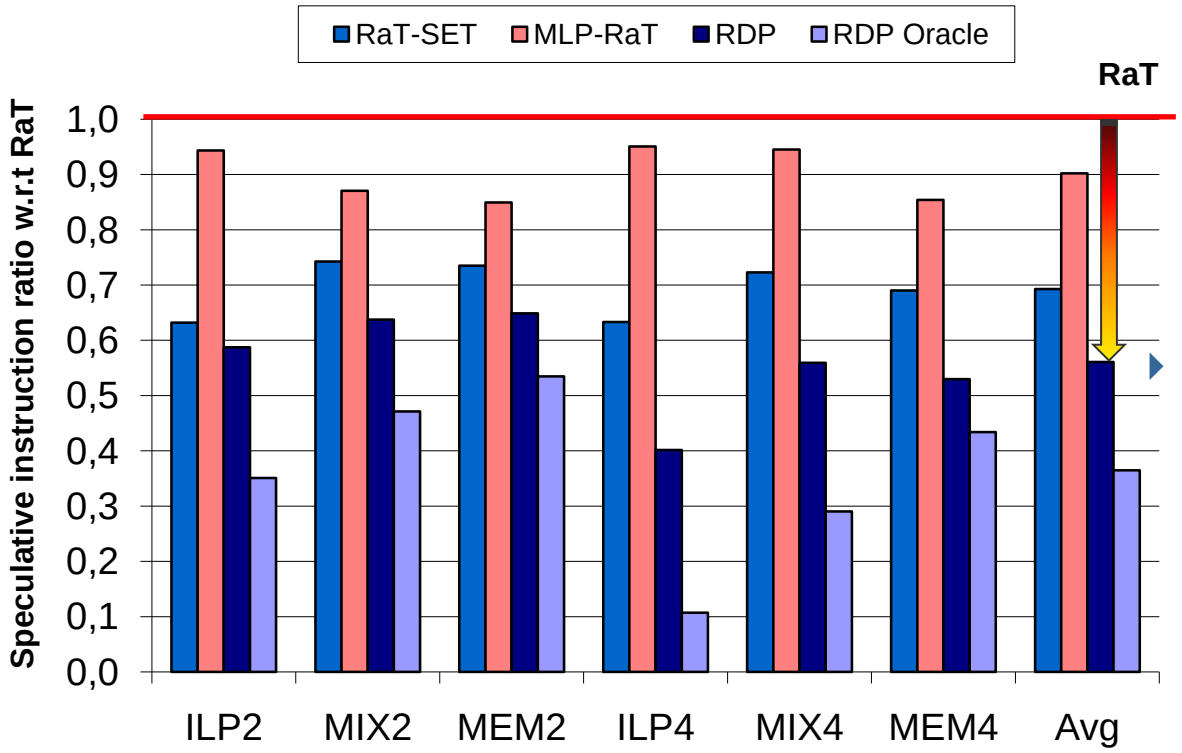
- IPC throughput and Hmean
  - RDP preserves the performance of RaT better than other mechanisms

    within ~2% throughput and ~4% hmean deviation



Extra Instructions

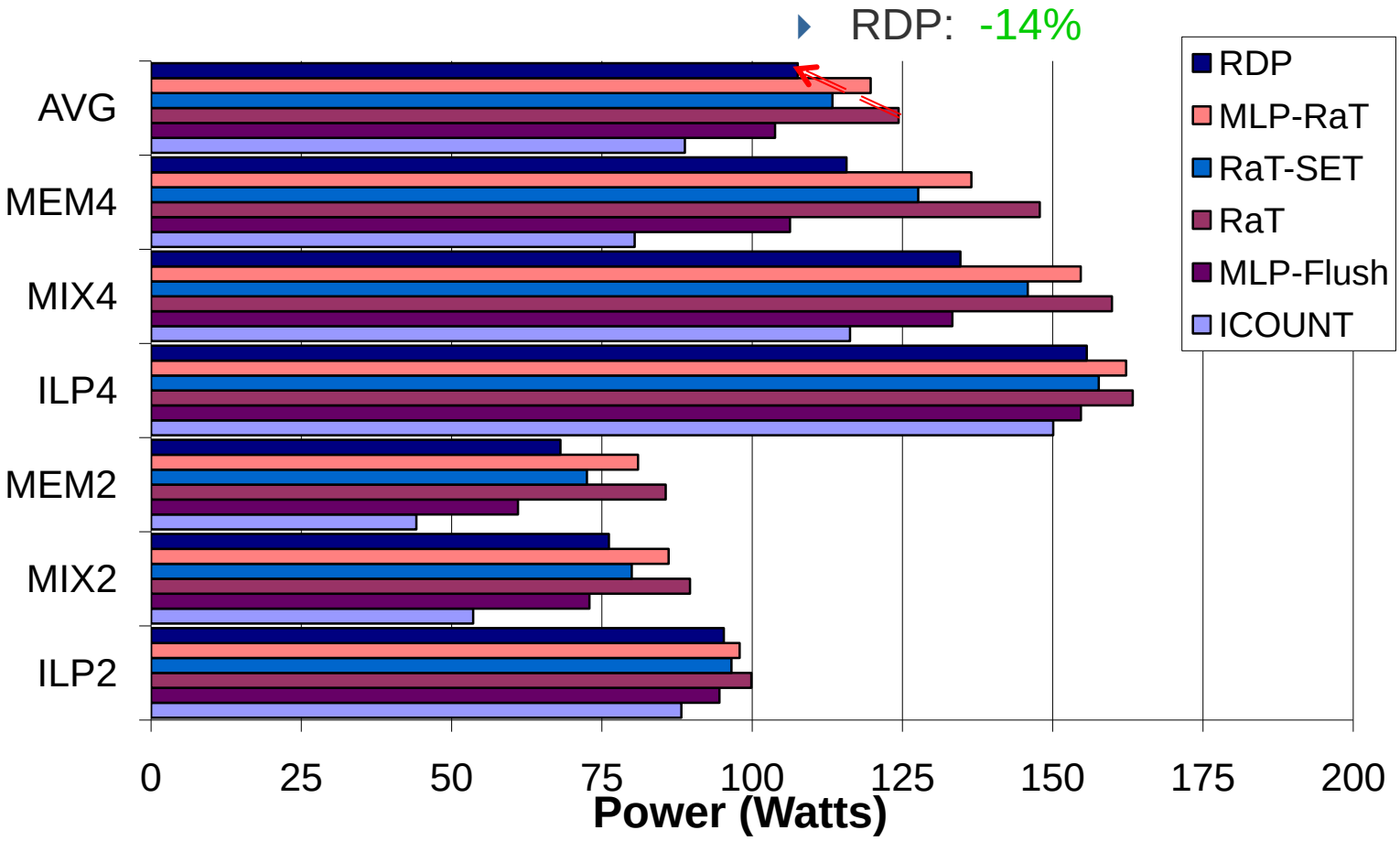
- RDP achieves the highest extra work reduction
  - come from eliminating both useless runahead threads and useless instructions out of runahead distance



RDP : -44%

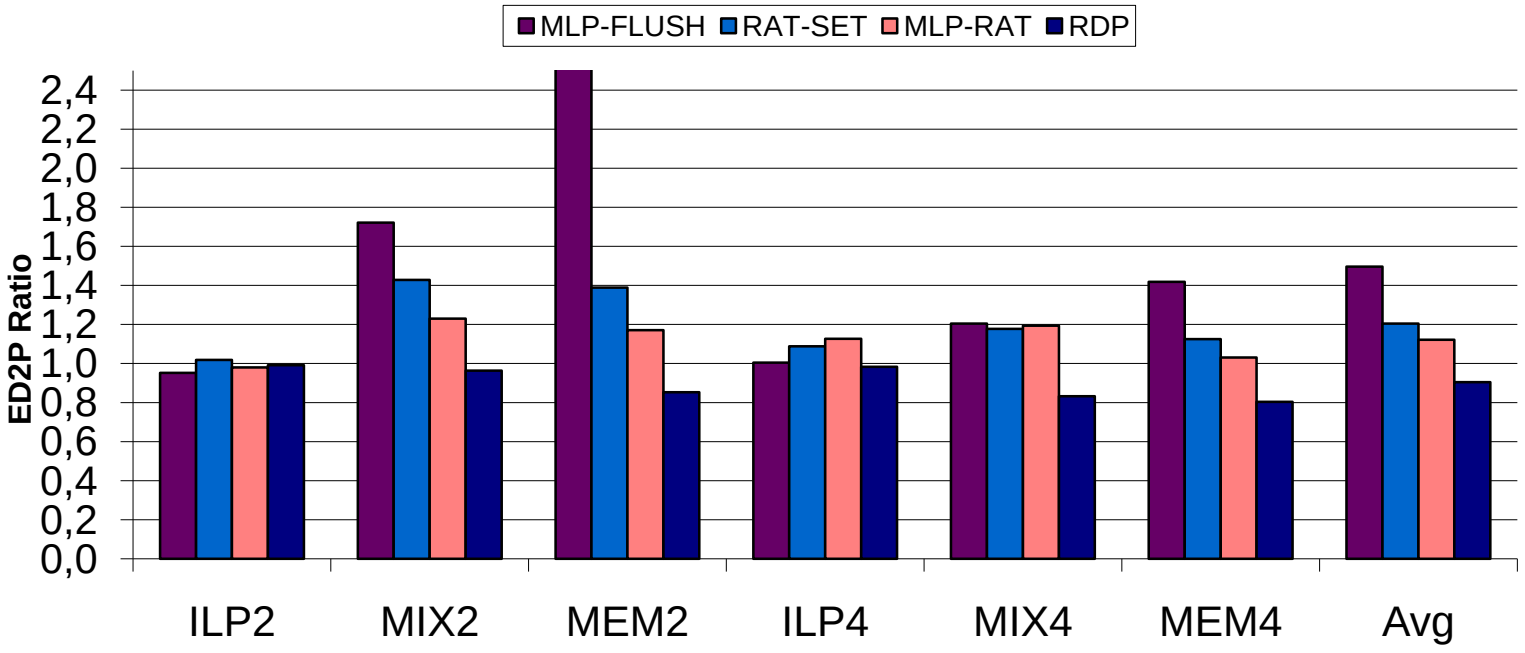
Power

- RDP effectively reduces the power consumption compared to RaT



Energy-efficiency

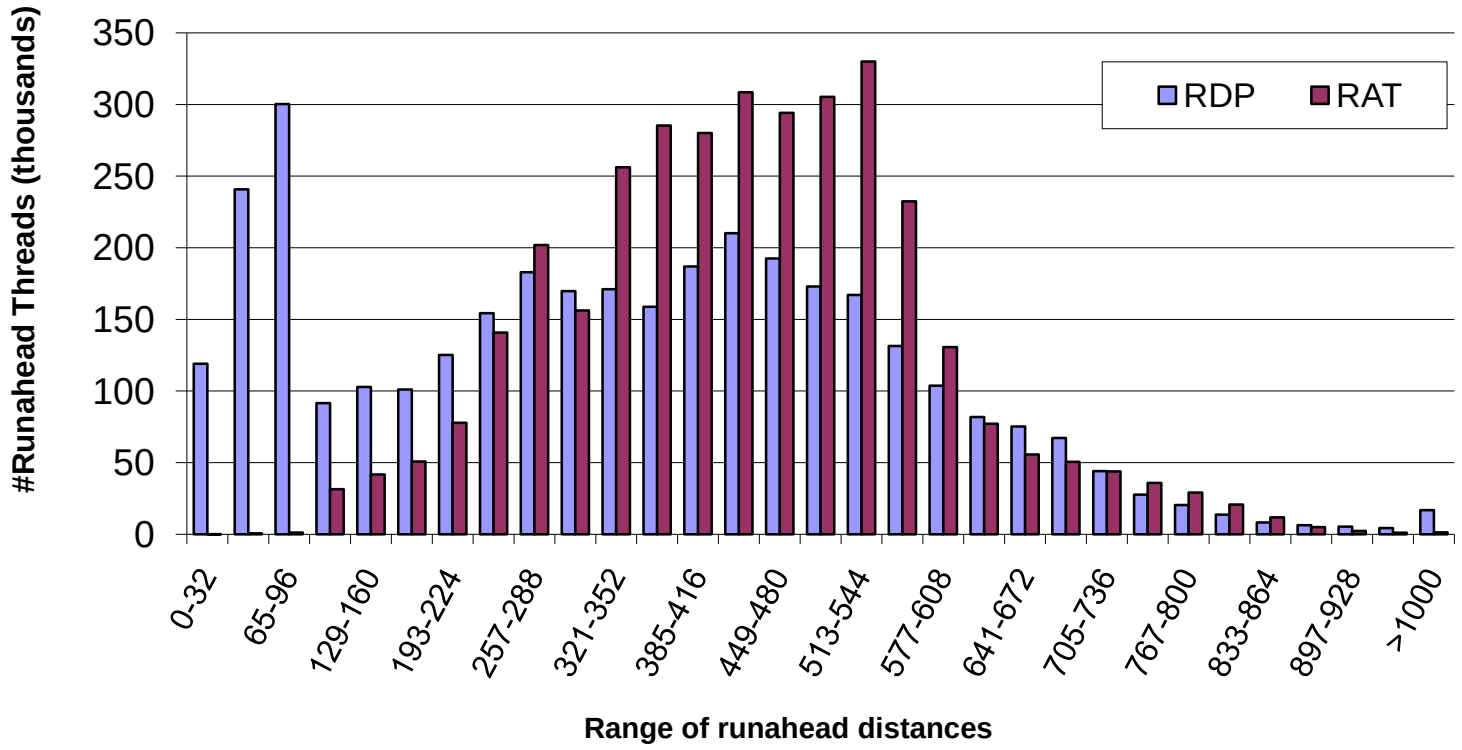
- Efficiency measured as energy-delay square (ED^2)
  - RDP presents the best ratio (10% better)
    - provide similar performance with RaT
    - cause less power consumption (-14% on average)



– *ED^2 normalized to the SMT+RaT*

Runahead threads behavior with RDP

- Controlled runahead threads distribution
  - RDP generates small runahead threads
  - but providing the same usefulness



Talk outline

- **Introduction**
  - RaT problem
  - Proposal
- **Efficient Runahead Threads**
  - Useful Runahead Distance
  - Runahead Distance Prediction (RDP)
- **Evaluation**
  - Framework
  - Results
- **Conclusions**

Conclusions

- RaT has a shortcoming
  - increase the extra work executed due to speculative instructions
  - efficiency
    - performance gain vs. extra energy consumption
- Our approach: ***efficient runahead threads***
  - Useful runahead distance prediction (RDP)
    - eliminate at a fine-grain the useless portion of runahead threads
    - reduces extra instructions while maintaining the performance benefits of runahead threads
- RDP shows significant performance and ED^2 advantage over state-of-the-art techniques

Thank you!

Efficient Runahead Threads

Tanausú Ramírez

Alex Pajuelo

Oliverio J. Santana


Onur Mutlu

Mateo Valero

The Nineteenth International Conference on

Parallel Architectures and Compilation Techniques (PACT)

11-15 September, Vienna, Austria

Backup Slices

PACT 2010

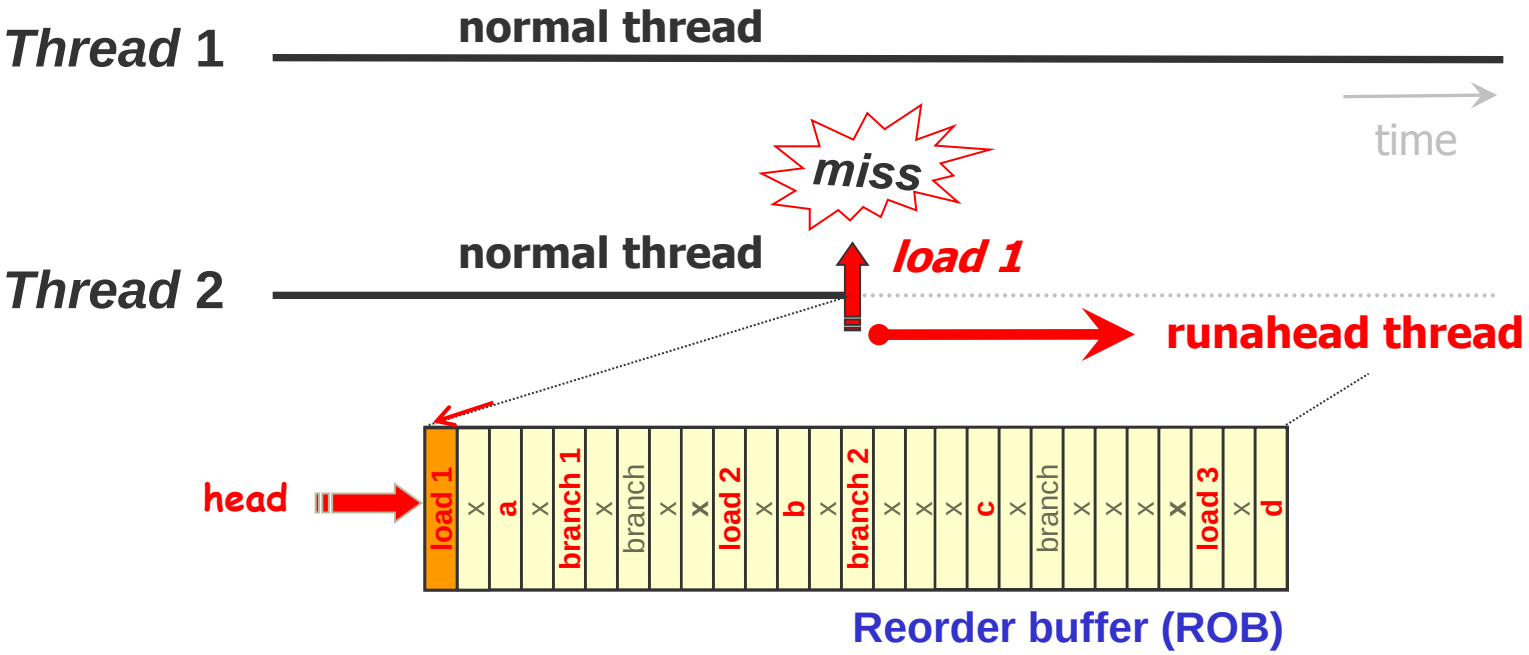
Talk

Runahead Threads (RaT)

- Multithreaded approach of Runahead paradigm
  - speculative thread → runahead thread

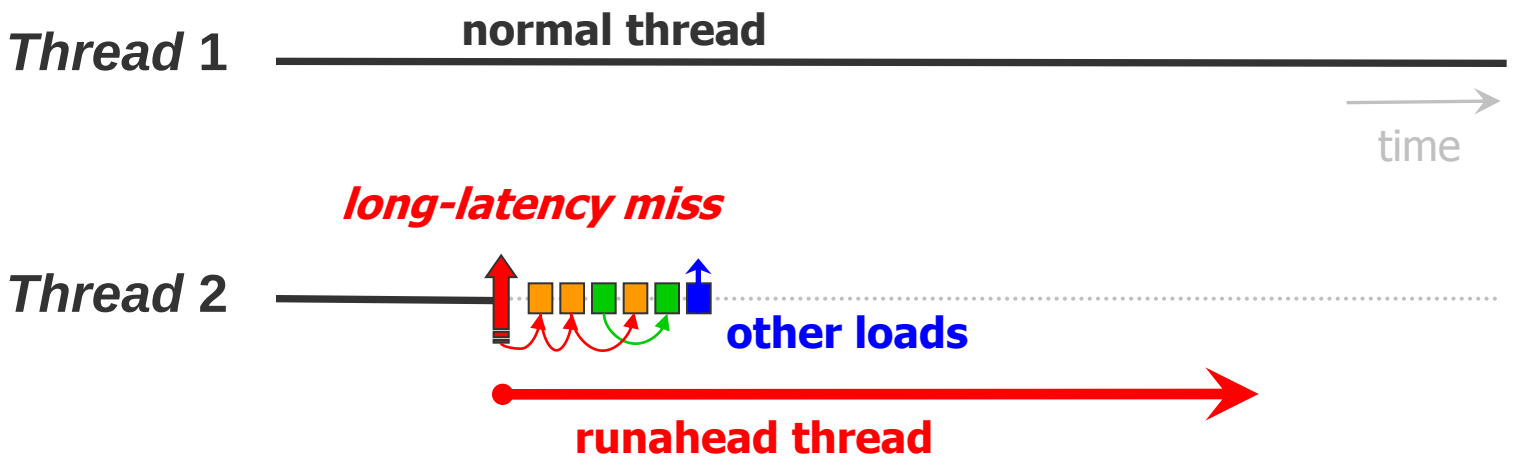
  - a runahead thread executes speculative instructions while a long-latency load is resolving
    - follow the program path instead of being stalled
    - independent execution of long-latency misses

Starting a runahead thread



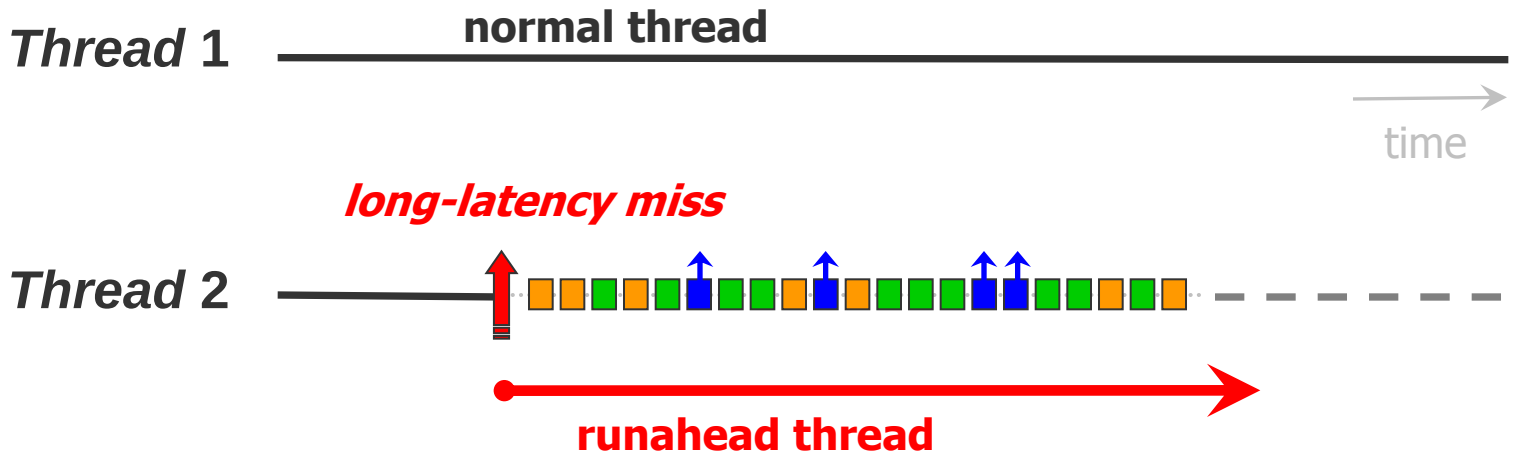
- When a thread is stalled by L2 cache miss
  - turn the thread into *runahead thread*
    - checkpoint architectural state
    - retire the miss
    - enter *runahead mode* (context switch is not needed)

Executing a runahead thread



- Instructions are speculatively executed
  - L2-miss dependent instructions are marked invalid and dropped
    - status are tracked through the registers (INV bits)
  - Valid instructions are executed
    - loads are issue to memory

Executing a runahead thread



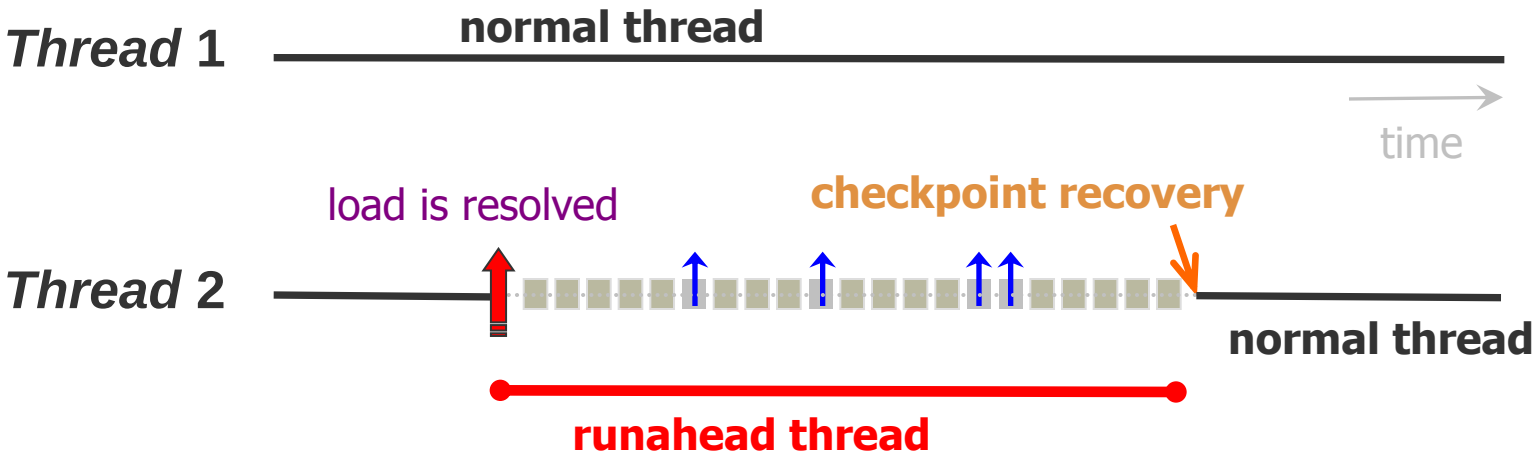
INValid instructions → don't use resources

Valid instructions → free resources faster

Long-latency loads → generate prefetches

- A runahead thread use and release faster shared resources to other threads

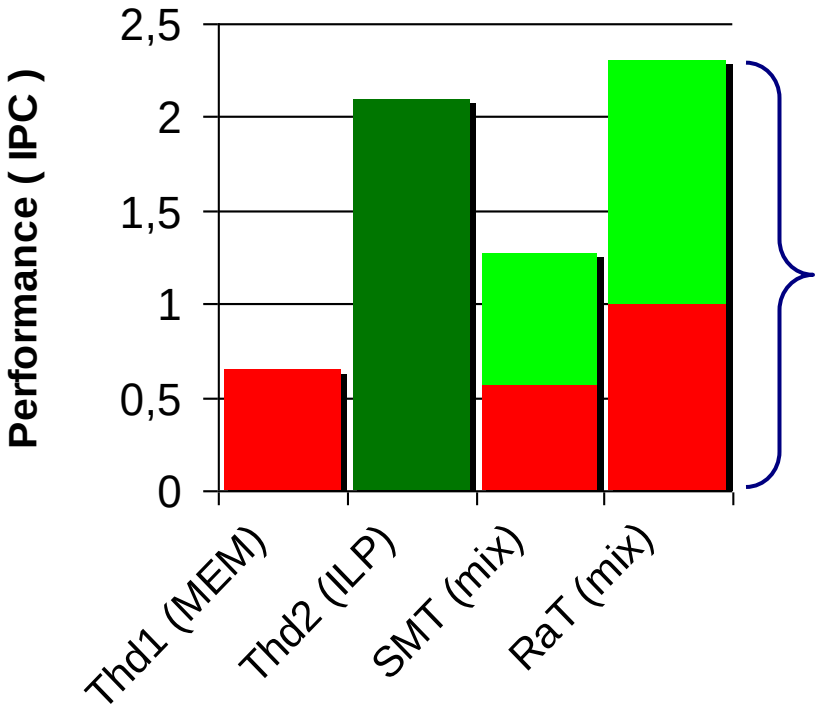
Finishing a runahead thread



- Leave runahead when L2 miss resolves
  - flush pipeline,
  - restore state from the checkpoint and,
  - resume to normal execution

RaT benefit results

Performance of 2-thread mix of a memory-bound and ILP-bound



- each thread is faster without disturbing the other threads
- Overall SMT processor performance is improved

- **single-thread performance is improved**
- **resource clogging is avoided**

Evaluation Framework

- **Methodology**
  - FAME
    - ensure multithreaded state continuously
    - threads fairly represented in the final measurements

- **Metrics**
  - performance
    - throughput
      - sum of individual IPCs
    - harmonic mean (hmean)
      - performance/fairness balance

$$\text{IPC Throughput} = \sum_{i=1}^{N} IPC_i$$

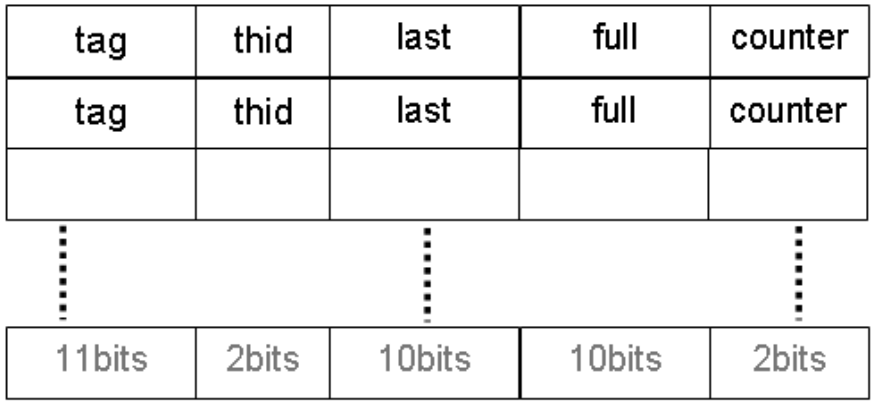
$$\text{Hmean} = \frac{N}{\sum_{i=1}^{N} \frac{IPC_{i,alone}}{IPC_{i,multithreaded}}}$$

  - energy efficiency
    - energy-delay^2 (ED^2)
      - ratio of performance and energy consumption

RDIT

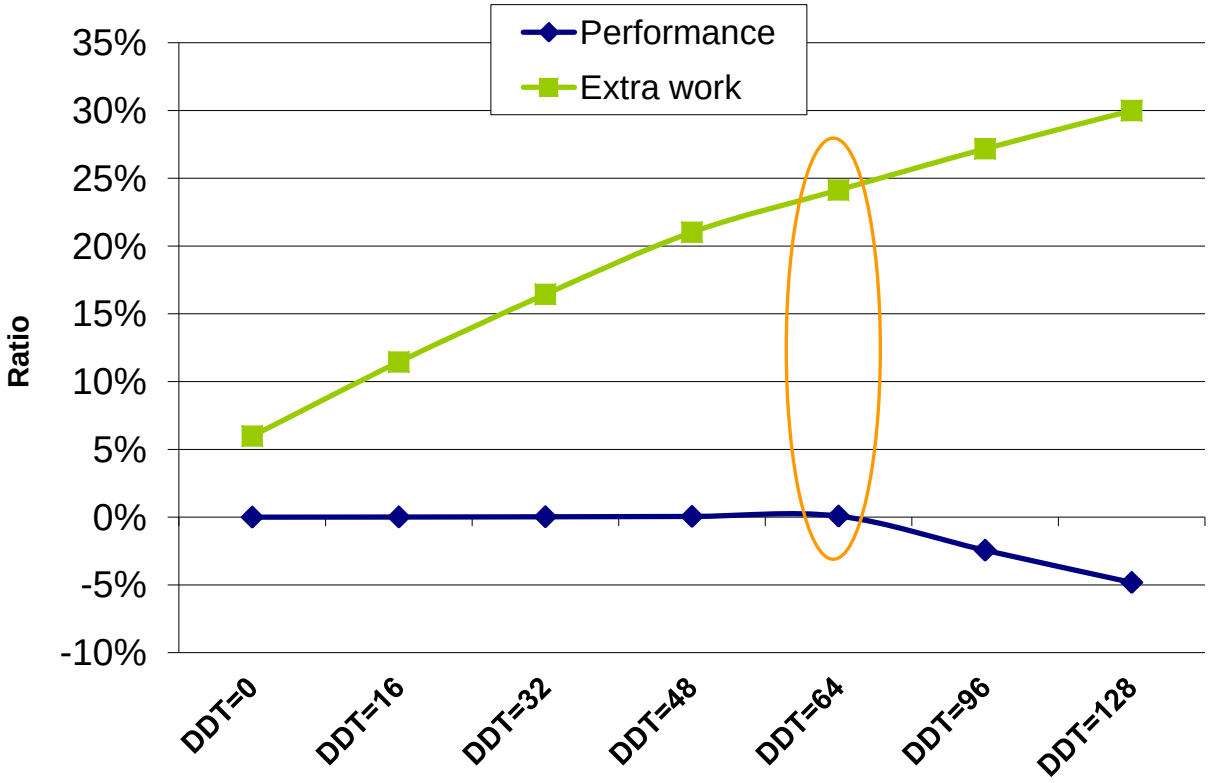
- RDIT related data
  - 4-way table of 1024 entries
  - size ~ 4.5KB
  - indexed by (PC) XORed (2-bits GHR)
  - power 1.09 Watts based on CACTI

Runahead Distance Information Table (RDIT)



DDT analysis

- Distance Difference threshold study



Distribution of RDP decisions

- breakdown of runahead threads based on the RDP decisions