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

thread 1  normal thread

thread 2  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

![Diagram showing memory resource usage]
- Orange: don’t use resources
- Green: free resources faster
- Blue: prefetches exploit MLP

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

Impact on performance-energy efficiency

More instructions → more energy

RaT increase ~50% power consumption
Proposal

- Improving RaT efficiency
  - avoid inefficient cases: no prefetching $\rightarrow$ useless instructions
  - only useful executions that maximize the performance
    - control the number of *useless* instructions executed by runahead threads

- Our goal $\rightarrow$ *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

long-latency miss

Prefetch (MLP)

Useless instructions

load is resolved

runahead thread

balance between prefetching and useless instructions
Useful Runahead Distance

By limiting the runahead execution of a thread, we generate **efficient runahead threads** that

- avoid unnecessary 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

---

<table>
<thead>
<tr>
<th>Runahead Distance Information Table (RDIT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

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

→ 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

1. decide to start the runahead thread?
   - if last < activation threshold → do not start runahead thread

2. 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

<table>
<thead>
<tr>
<th>Processor core</th>
<th>Memory subsystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor depth</td>
<td>64 KB, 4-way, pipelined</td>
</tr>
<tr>
<td>Processor width</td>
<td>64 KB, 4-way, 3 cyc latency</td>
</tr>
<tr>
<td>Reorder buffer size</td>
<td>1 MB, 8-way, 20 cyc latency</td>
</tr>
<tr>
<td>INT/FP registers</td>
<td>Caches line size</td>
</tr>
<tr>
<td>INT / FP /LS issue queues</td>
<td>64 bytes</td>
</tr>
<tr>
<td>INT / FP /LdSt unit</td>
<td>Main memory latency</td>
</tr>
<tr>
<td>Hardware prefetcher</td>
<td>300 cycles minimum</td>
</tr>
</tbody>
</table>
Evaluation

- Performance and energy-efficiency analysis
  - throughput, hmean of thread weighted speedups
  - extra instructions, power, ED²

- 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

Efficient Runahead Threads

Tanausú Ramírez (UPC)
RDP achieves the highest extra work reduction come from eliminating both useless runahead threads and useless instructions out of runahead distance.

![Bar chart showing speculative instruction ratio w.r.t. RaT for different workloads. The chart indicates that RDP achieves a significant reduction in extra work, specifically -44%.](image-url)
RDP effectively reduces the power consumption compared to RaT

RDP: -14%
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)

![Bar chart showing ED$^2$ ratio for different workloads: ILP2, MIX2, MEM2, ILP4, MIX4, MEM4, and Avg. The bars are labeled MLP-FLUSH, RAT-SET, MLP-RAT, and RDP. The chart indicates that RDP has the highest ED$^2$ ratio, followed by MLP-FLUSH, RAT-SET, and MLP-RAT, with Avg showing a slight variation across workloads.](image-url)

- ED$^2$ normalized to the SMT+RaT

Efficient Runahead Threads
Runahead threads behavior with RDP

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

![Range of runahead distances](chart.png)
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

Thread 1

Thread 2

- 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

Thread 1

Thread 2

Efficient Runahead Threads

Tanausú Ramírez (UPC)

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

- INValid instructions: don’t use resources
- Valid instructions: free resources faster
- Long-latency loads: generate prefetches
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

- 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
  - energy efficiency
    - energy-delay² (ED²)
      - ratio of performance and energy consumption

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

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

<table>
<thead>
<tr>
<th>tag</th>
<th>thid</th>
<th>last</th>
<th>full</th>
<th>counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>thid</td>
<td>last</td>
<td>full</td>
<td>counter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11bits</td>
<td>2bits</td>
<td>10bits</td>
<td>10bits</td>
<td>2bits</td>
</tr>
</tbody>
</table>

Efficient Runahead Threads  
Tanausú Ramírez (UPC)
DDT analysis

- Distance Difference threshold study

![Graph showing the relationship between DDT and performance and extra work. The graph lines indicate that as DDT increases, performance decreases and extra work increases.](image-url)
Distribution of RDP decisions

- breakdown of runahead threads based on the RDP decisions

![Graph showing the distribution of RDP decisions for various threads and benchmarks across ILP2, MIX2, and MEM2.]