Boosting Single-thread Performance in Multi-Core Systems through Fine-Grain Multi-Threading

Intel Labs Barcelona

Carlos Madriles, Pedro López, Josep M. Codina, Enric Gibert, Fernando Latorre, Alejandro Martínez, Raúl Martínez and Antonio González

ISCA’09
June 24th
Why Is Single Thread Performance Important?

- Exploiting TLP is more power-efficient than exploiting ILP
  - The trend is to increase the number of cores integrated into the same die
  - Power and thermal constraints advocate for simpler cores
- However, single thread performance matters
  - Sequential and lightly threaded apps
  - Sequential regions limit scalability of parallelism (Amdahl’s law)

Schemes to boost single-thread performance on CMPs are crucial
Main Research Efforts

• Main research efforts to exploit both ILP and TLP for boosting single-thread performance on CMPs
  – Adaptative and cluster microarchitectures
  – Combine or fuse cores
    – Parallelism constrained by the instruction window
  – Speculative multithreading (SpMT) techniques
    – Traditional schemes constrain the exploitable parallelism

New techniques to exploit parallelism are needed
Our Contribution: Anaphase

• Novel speculative multithreading technique to boost single-thread applications on CMPs

• Effectively exploits ILP, TLP, and MLP
  – Thread decomposition performed at instruction granularity
    – Adapts to available parallelism in the application
    – Minimizes inter-thread dependences
    – Improves workload balance
    – Increases the amount of exploitable MLP
  – Hw / Sw hybrid scheme
    – Low hardware complexity and powerful thread decomposition

• Cost-effective hardware support on top of a conventional CMP
  – A novel uncore module supports anaphase threads execution
  – Very few changes on the cores
Outline

• The Anaphase Scheme
  – Threading Model
  – Decomposition Algorithm

• The Anaphase Hardware Support
  – Architecture Overview
  – Architectural State Management

• Evaluation

• Conclusions
Anaphase Scheme Overview

Compiler

Serial Program

Anaphase Parallelizer

Parallellized Program

CMP with Anaphase Hw

L3

L2

TILE

C0

C1

C2

C3

Sw

Hw

It behaves like a fused core

Select hot regions
Fine-grain Thread Decomposition

Serial Program

Parallellized Program

Compiler

Anaphase Parallelizer

TILE

C0

C1

C2

C3

Sw

Hw

It behaves like a fused core
Anaphase Scheme Overview

- Main distinguishing features:
  1. Threading model
  2. Fine-grain thread decomposition
  3. Hardware support

It behaves like a fused core
Threading Model: Key Features

Original Region CFG

Thread 1

Thread 2

Threads may be composed of non-consecutive instructions
Threading Model: Key Features

Uses a heuristic to decide how to handle inter-thread dependences:
• Ignore → Hw recovers in case of error
• Explicit communication
• Pre-compute (p-slice)
Threading Model: Key Features

Original Region CFG

Thread 1

Thread 2

Replicate those instructions needed to generate the consumed value
Threading Model: Key Features

Original Region CFG

A1
A2
A3 (Br)

B1
B2
B3
B4

C1
C2 (Br)

D1
D2
D3 (Br)

Thread 1

A1’
A2
A3 (Br)

B2
B3

C1
C2 (Br)

D1
D2
D3 (Br)

Thread 2

A1
A3 (Br)

B1
B4

D3 (Br)

Threads include all required control instructions
- Some branches may need to be replicated
- Keep Hw simple (no changes in front-end)
Decomposition Algorithm [1]

- Multi-level graph partitioning
  - Coarsening
    - First partition of the region DDG
    - Based on: critical path, independence, workload balance, and delinquent loads
  - Refinement
    - Evolution of the traditional K-L algorithm
    - Refines first partition based on a model that estimates execution time

Outline

• The Anaphase Scheme
  – Threading Model
  – Decomposition Algorithm

• The Anaphase Hardware Support
  – Architecture Overview
  – Architectural State Management

• Evaluation

• Conclusions
About 7% increase in area on an Intel® Core™ 2 Duo
Anaphase Execution

Tile in single-core mode

Sequential

0 1 2 3 4 5 6 7

Core 0

IC DC

Core 1

IC DC

ICMC

L2
Anaphase Execution

Tile in cooperative mode

Thread 1: 0, 2, 4, 6
Thread 2: 1, 3, 5, 7

Local retire in thread order

Core 0: IC, DC
Core 1: IC, DC

ICMC

L2

TILE
Anaphase Execution

Tile in cooperative mode

Thread 1
0
2
4
6

Thread 2
1
3
5
7

Global commit in program order

Core 0
IC
DC

Core 1
IC
DC

ICMC
2
1

L2

TILE

ICMC
2
1
Anaphase Execution

Tile in cooperative mode

Thread 1
0 2 4 6

Thread 2
1 3 5 7
Anaphase Execution

Tile in cooperative mode

Thread 1: 0, 2, 4, 6
Thread 2: 1, 3, 5, 7

Core 0
IC
DC

Core 1
IC
DC

L2

ICMC

Inter-core memory violation!
Anaphase Execution

Tile in single-core mode

Sequential

0
1
2
3
4
5
6
7

TCLE

L2

ICMC

Core 0

IC
DC

Core 1

IC
DC
Architectural State Management

• Memory
  – Speculative state kept private in L1 and L2 caches
  – Merge performed in program order in the L2 cache
  – Detects inter-core memory violations at chunk granularity
  – Supports frequent memory checkpoints and very fast squash and commit operations

• Registers
  – Speculative state kept in core PRF
  – Register architectural state merged in the ICMC in program order
  – Supports frequent register checkpoints
    – Reduces pressure on the PRF
    – It allows a core to make forward progress
Outline

• The Anaphase Scheme
  – Threading Model
  – Decomposition Algorithm

• The Anaphase Hardware Support
  – Architecture Overview
  – Architectural State Management

• Evaluation

• Conclusions
Experimental Framework

• Compiler
  – Anaphase partitioning implemented on top of the Intel® Compiler (icc)

• Simulator
  – Cycle accurate x86 CMP (1 tile with 2 OoO cores modeled)
  – Two different core configurations evaluated
    – Intel® Core™ 2 like (Medium core)
    – Half-sized main structures (Tiny core)
  – Core Fusion[1] proposal implemented for comparison

• Benchmarks: Spec2006
  – Profiling: PinPoint 20M traces using train input set
  – Evaluation: 100M traces using ref input set

Performance Evaluation

AVG SpeedUp: Anaphase=41% CoreFusion=28%

MLP exploited by Anaphase

Results shown for Tiny core configuration
Dynamic Instruction Breakdown

High coverage → less 10% serial code
Low overhead → 22% replicated instrs and explicit comms
Few explicit comms → only 3%
High accuracy → less 1% squashed instrs
Conclusions

• Anaphase is a Sw / Hw technique that leverages multiple cores to boost single-thread code

• It exploits ILP, TLP, and MLP with high accuracy and low overheads

• Low hardware overhead (7%)

• Important benefits: 41% average speedup for Spec2006
Thank you!