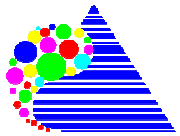
Architecture and Compiler Support for Speculative Precomputation

Presented by:

Steve Shih-wei Liao (Intel MRL)

Dean Tullsen (UCSD)

Donald Yeung (UMCP)



A tutorial held in conjunction with the 12th International Conference on Parallel Architectures and Compilation Techniques

Tutorial Information

- Schedule

8:30 - 8:45	Welcome and Introduction
8:45 - 9:45	Architecture Support
9:45 -10:00	Compiler Support
10:00 -10:30	Break
10:30 -11:00	Compiler Support cont. d
11:00 -12:00	Industry Perspective

- Web: http://www.ece.umd.edu/~yeung/tutorial

- Contact:

Donald Yeung
1327 A. V. Williams
College Park, MD 20742
yeung@eng.umd.edu
(301) 405-3649

Performance Degrading Events

- Long-latency events degrade performance
- Architectural level
  - Memory access

    e.g. instructions and data
  - Instruction execution

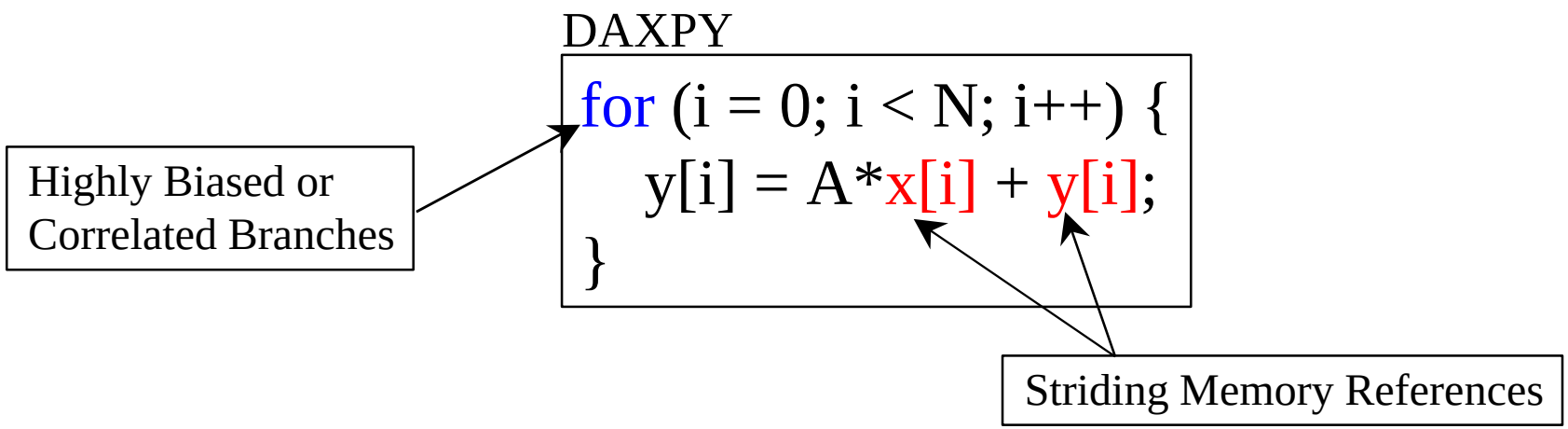
    e.g. branches
- Application or OS level
  - I/O operations

    e.g. file system

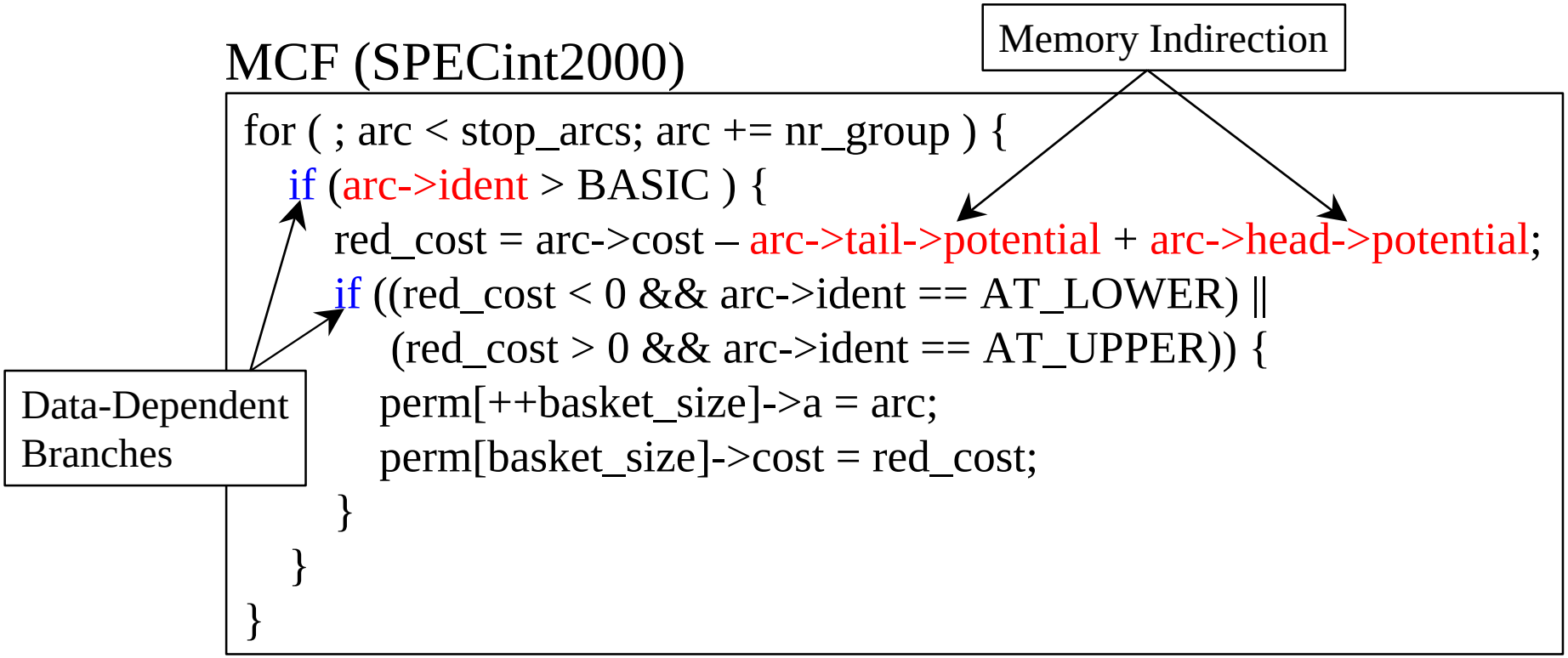
Existing Techniques

- Latency reduction (caches)
- Latency tolerance (prefetching)
- Speculation (branch prediction)

⇨ Effective for programs that exhibit locality, regularity, and/or predictability:



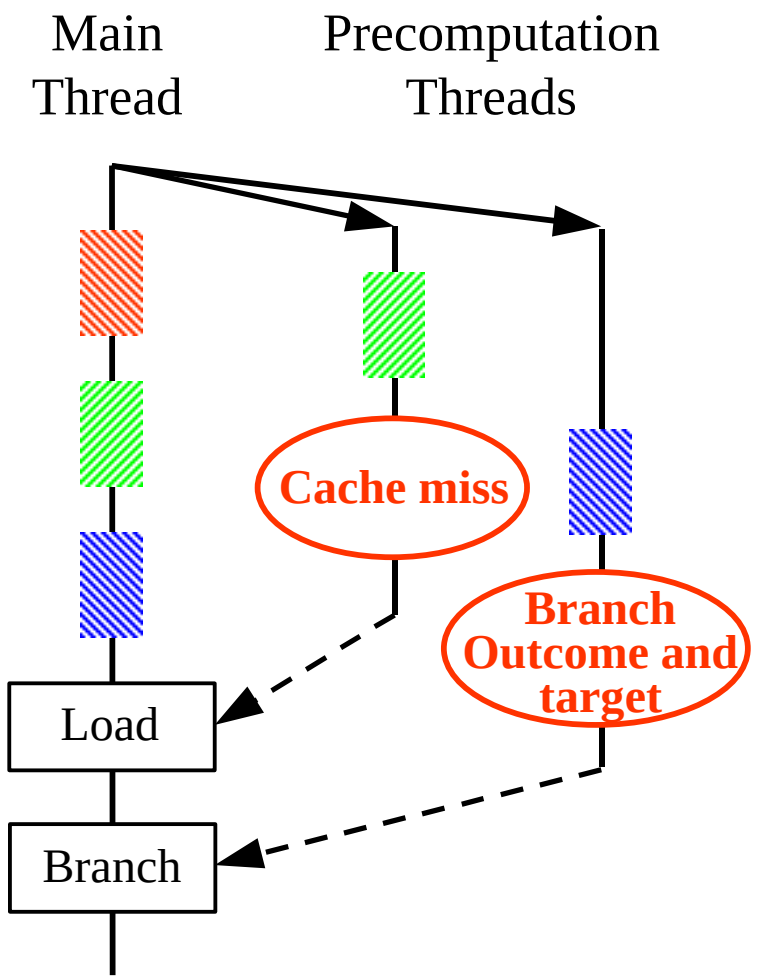
Many Programs Exhibit Complex Behavior



- Conventional techniques are less effective for:
  - Pointer chasing memory references
  - Uncorrelated data-dependent branches
  - Complex control and data flow

Speculative Precomputation

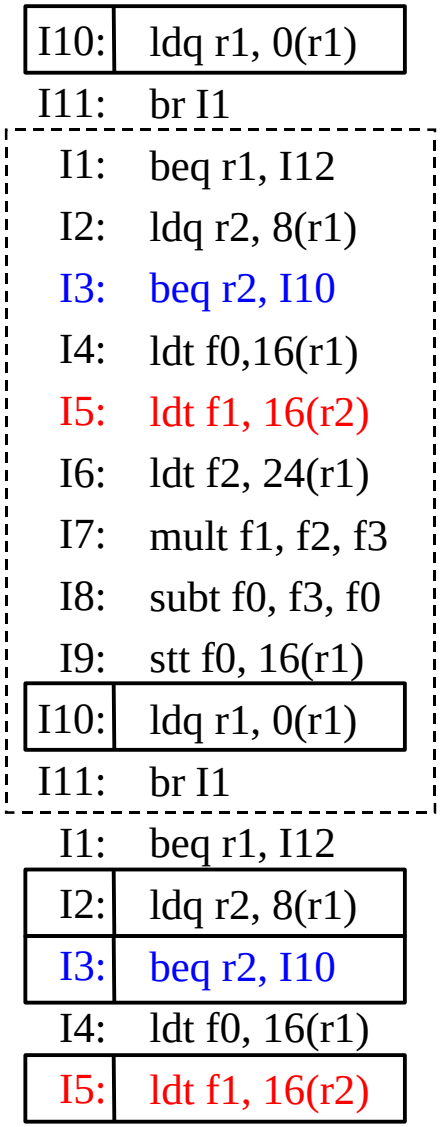
- Perform redundant execution in **precomputation threads** to trigger long-latency events on behalf of the main thread
- Key: precomputation threads must get ahead of the main thread



Precomputation Slices (p-slices)

```
for (node = list; node; node = node->next)
   if (node->neighbor != NULL)
      node->value -=
            node->neighbor->value * node->coeff
```

- **Backward slice** from each long-latency instruction
- Stop when slice advantage is sufficient
- Stopping point is the **trigger** for the slice



Compared to Existing Techniques

- Precomputation threads are:
  - highly accurate
  - purely speculative
  - independent of main thread

- Exploits multithreading for single-thread performance