Speculative Multithreaded Processors

Guri Sohi and Amir Roth

Computer Sciences Department
University of Wisconsin-Madison

Outline

- Trends and their implications
- Workloads for future processors
- Program parallelization and speculative threads
  - speculative control-driven threads
  - speculative data-driven threads
- Sample applications and research issues
- Summary

Driving Factors

- Match upcoming technology trends
- Match upcoming software trends
- Match upcoming technology constraints
- Match upcoming design constraints
- Learn, and exploit, new program behaviors

Hardware/Design Trends

- Increasing wire delays
- Increasing memory latencies
- Deeper pipelines
- Design complexity
- Verification complexity
- Power issues

Implications of Trends

- Distributed microarchitectures
- Clustered superscalar, with multithreading
- Chip multiprocessor

Question: what to run on underlying microarchitecture?

Work for Distributed/Multithreaded Processor

- Independent programs
  - increase overall processing throughput
  - works well in server environment
- Independent threads of multithreaded application
  - increase overall throughput
  - compatible with software trends?
- Related threads
  - e.g., for reliability
- But what about speeding up single program execution?
  - single program speed will continue to be important
  - how to “parallelize” or “multithread” single program?

Program Parallelization

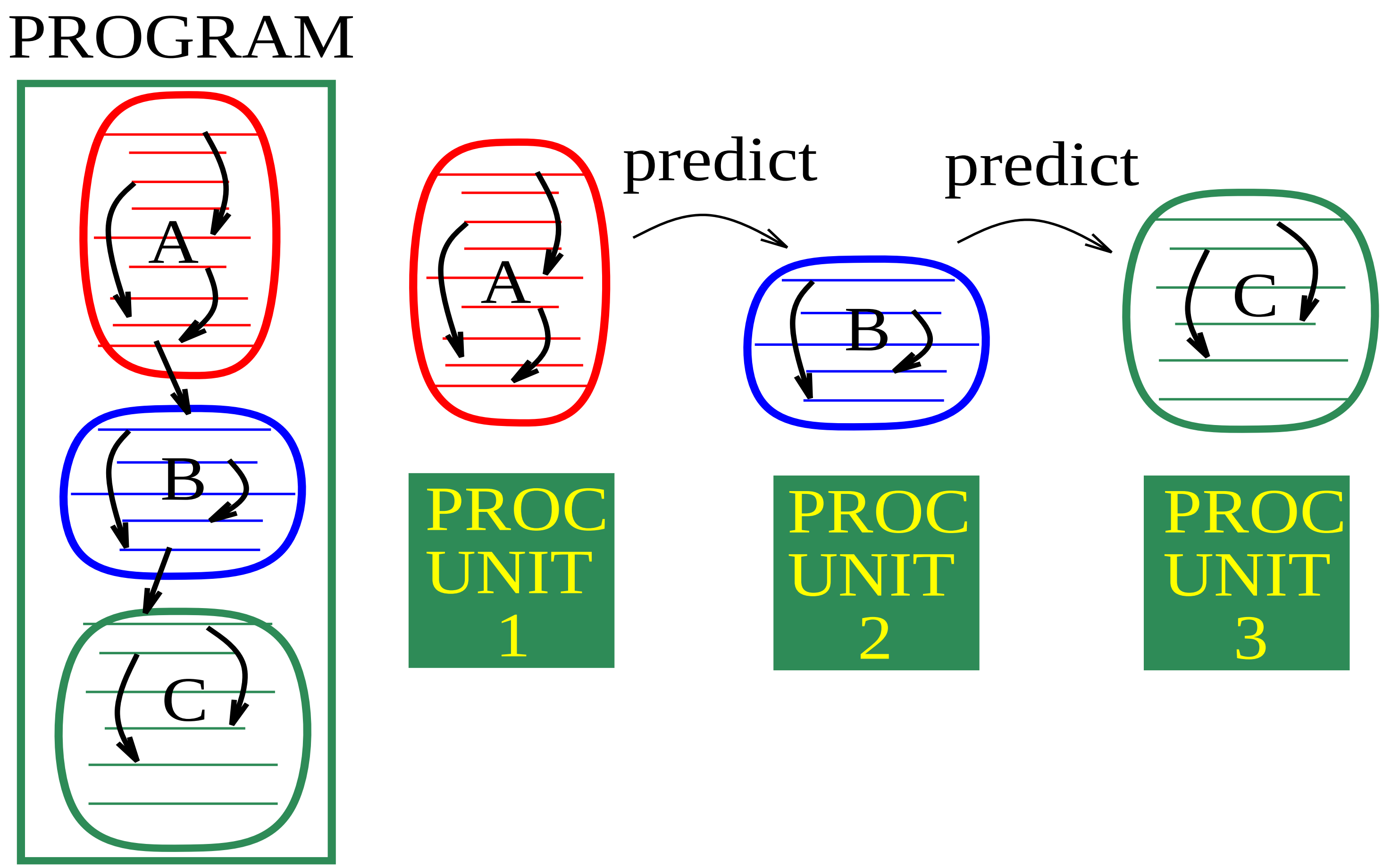
- What does it mean to parallelize?
  - how to divide program into multiple portions
- What constrains parallelization?
  - dependences (especially ambiguous)
- How to overcome constraints?
  - use speculation

Program Parallelization -- Theme I

- **Traditional view: control-driven threads**
  - **divide work into multiple groups of instructions**
    - conservative assumptions about dependences constrain parallelization
  - **each group is specified using traditional control-driven (von Neumann) semantics**

- **A newer view: multiscalar**
  - **use dependence speculation to overcome constraints**
  - **commercial example: Sun MAJC**

Multiscalar: Speculative Control-Driven Threads



Program Parallelization -- Theme II

- **Another traditional view: dataflow**
  - divide work into (dependent) computations
  - each computation is represented in a data-driven manner
- **A newer view: speculative data-driven "threads"**
  - use speculation to facilitate thread creation

Motivation for Data-driven Threads

- program execution: processing of low-latency instructions, with pauses for high-latency events
- parallelizing low-latency instructions isn't crucial
- overlapping high-latency events is what matters!
- "threads" should create high-latency events early

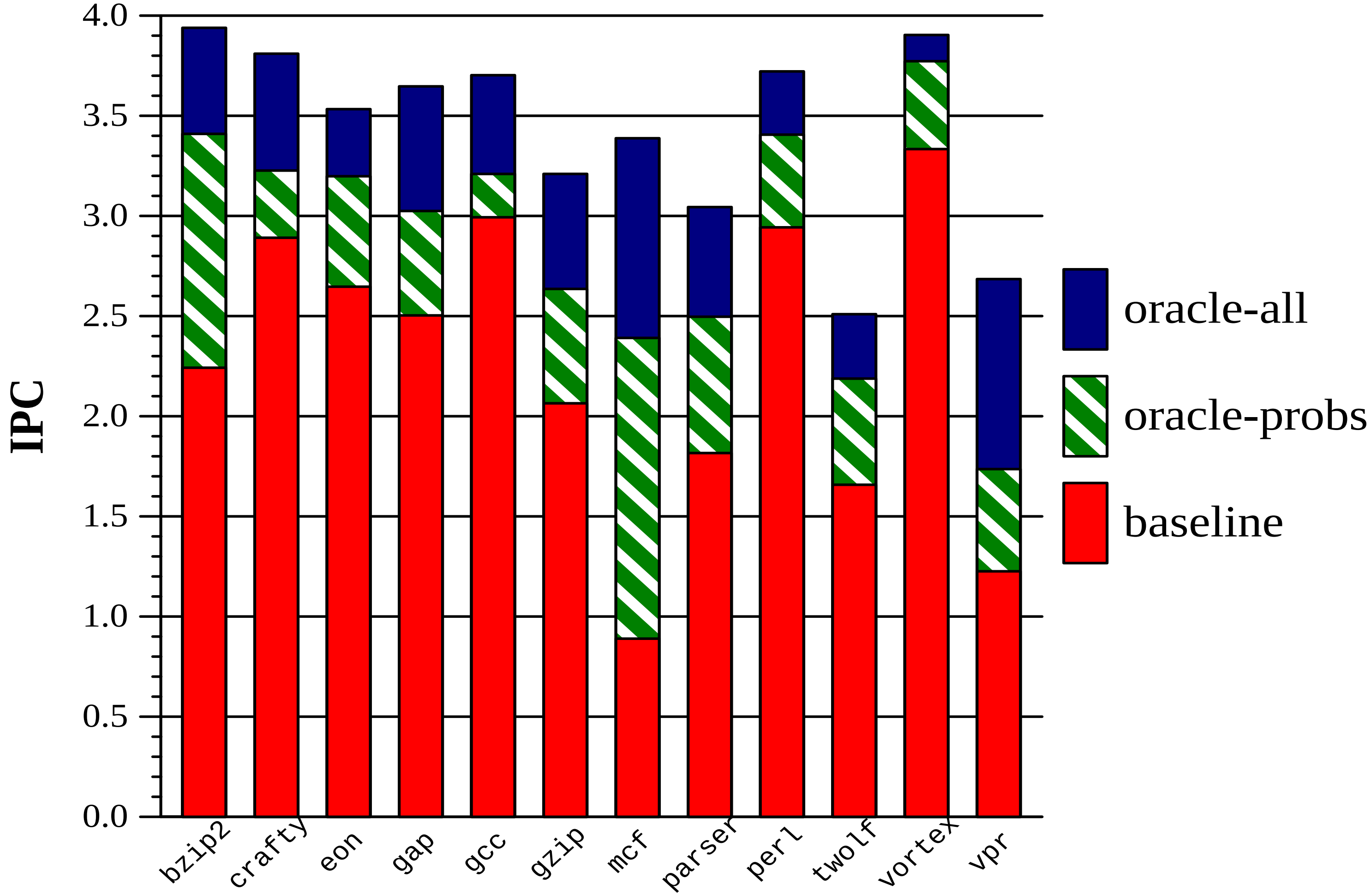
Speculative Data-Driven Threads

- Use dependence relationships to isolate thread(s) of code from main program thread
  - use speculation to facilitate creation
- Execute threads (speculatively) in parallel with “main program”
  - “assist” main thread via side-effects
  - don’t impact architectural correctness

Application: Cache Misses and Branch Mispredicts

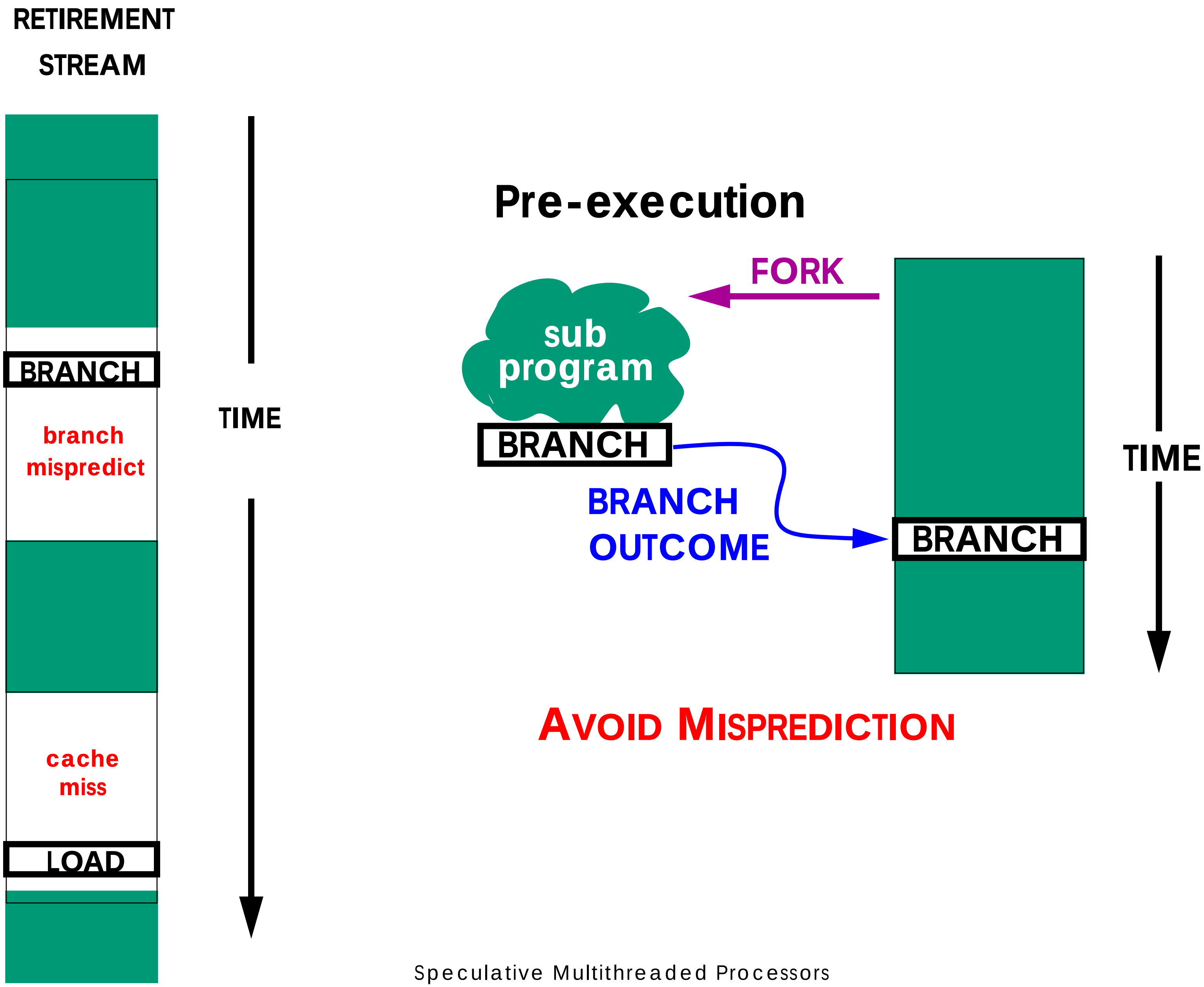
Spec2000 Benchmark	Memory: # inst	Memory: % dyn. memops	Memory: % miss	Control: # inst	Control: % dyn. branch	Control: %misp
bzip2	24	3	63	62	26	77
crafty	35	2	54	51	7	30
eon	Insufficient misses			24	13	71
gap	66	1	28	123	10	65
gcc	122	4	5	122	6	34
gzip	15	21	75	46	14	55
mcf	42	35	69	32	20	71
parser	70	4	42	80	10	38
perl	74	1	26	43	7	61
twolf	116	7	60	87	39	73
vortex	71	1	22	83	1	41
vpr (route)	55	13	67	72	16	75

Performance Leverage



Perfecting a small set of instructions provides significant performance much of that of a perfect branch predictor and data cache

Using Speculative Data-driven Threads



Sample Performance Results

VPR (ROUTE)

- 200M instruction sample (starting at 14.1B on 20B run)
- 100M instruction warm-up for caches/predictors

32% SPEEDUP: 16% FROM PRE-FETCHING, 16% FROM BRANCHES

	Cache Misses (primary L1)			Branch Mispredictions		
	number	rate	/1000 inst	number	rate	/1000 inst
base	2,850,000	3.3%	14.3	1,400,000	7.3%	7.0
w/slices	1,340,000	1.6%	6.7	420,000	2.2%	2.1

- The full latency of the misses and mispredictions is not always hidden

Sample Applications

- Cache prefetching/management
- Computing branch outcomes
- TLB prefetching/management
- I/O prefetching
- Multiprocessor communication management

Some Research Issues

- How to divide control-driven program into data-driven threads?
- When to divide program?
- How to represent data-driven threads?
- Managing mixed thread workloads

Summary

- Hardware and design trends will lead to distributed/multithreaded processors
- Many options for running different thread types on underlying microarchitecture
- Overcome constraints to traditional "parallelization" techniques with speculation
    - speculative control-driven threads
    - speculative data-driven threads
- Most of the research still needs to be done