A Scheme of Predictor Based Stream Buffers

Bill Hodges, Guoqiang Pan, Lixin Su

Outline

- Background and motivation
- Project hypothesis
- Our scheme of predictor-based stream buffer
  - Predictors
  - Predictor table
  - “Stream buffer”
- Results and analysis
- Research topics left

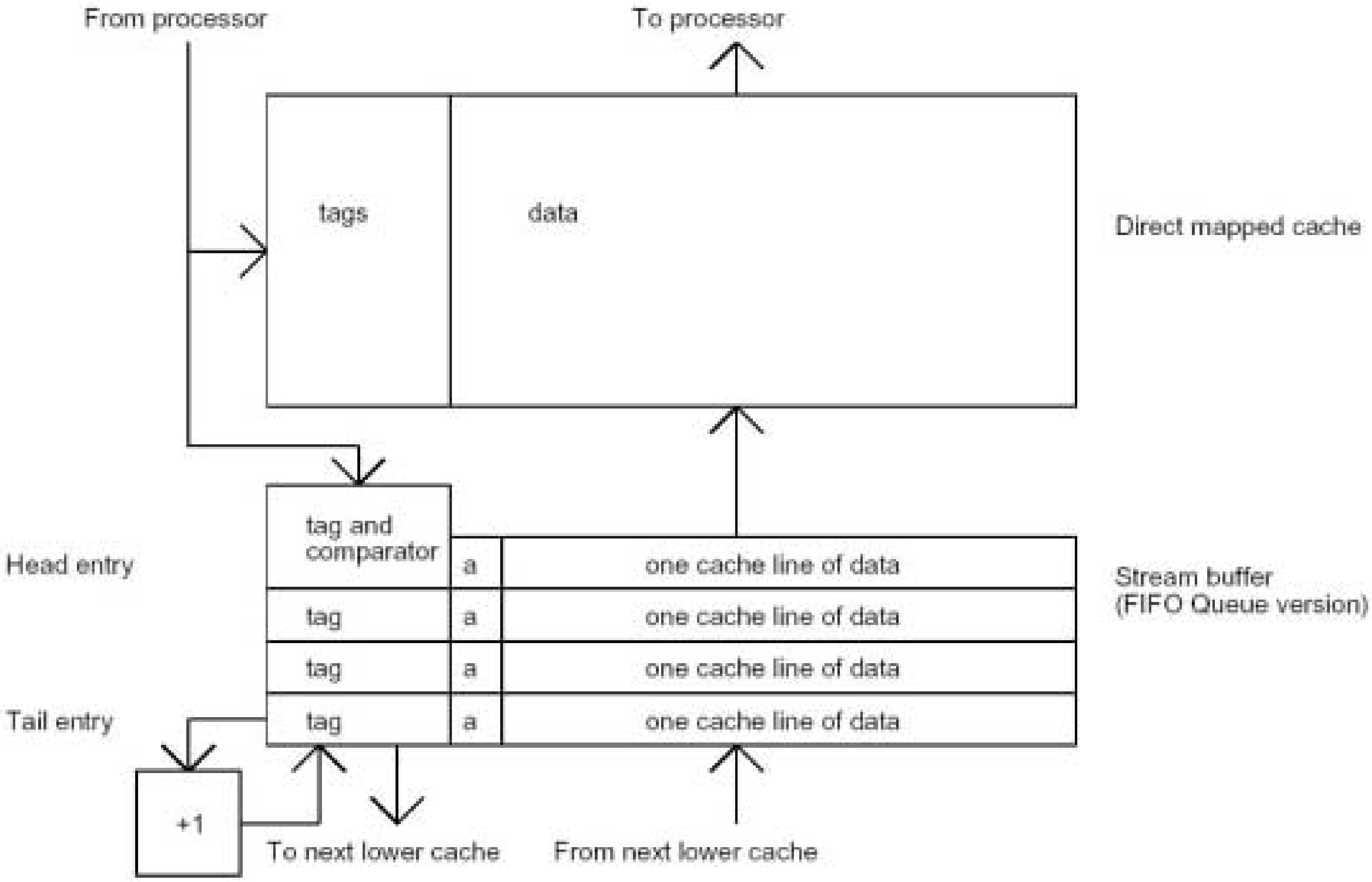
Contributors to Microprocessor Performance Degradation

- Branch misprediction
- Different hazards
- TLB misses
- Cache misses
  - Instruction cache misses
  - Data cache miss rate (vary from a few to tens of percent)

Techniques to Reduce Data Cache Misses In Superscalar

- Redesign RF-Cache-DRAM memory hierarchy
  - Addition of L2 and L3 caches, move L2 onto chip
  - Victim cache, Pseudo cache
  - Stream Buffer
- Aggressive instruction scheduling

How a Standard Stream Buffer Works



From N. Jouppi

Stream Buffer Research Activities

- Where to start prefetching?
  - Sequentially prefetching
  - Prefetching with a stride
  - Predictor based prefetching
- How to avoid frequent flushing of stream buffer
- How to prevent pollution of L1 data cache?
- Decoupled or coupled predictors?

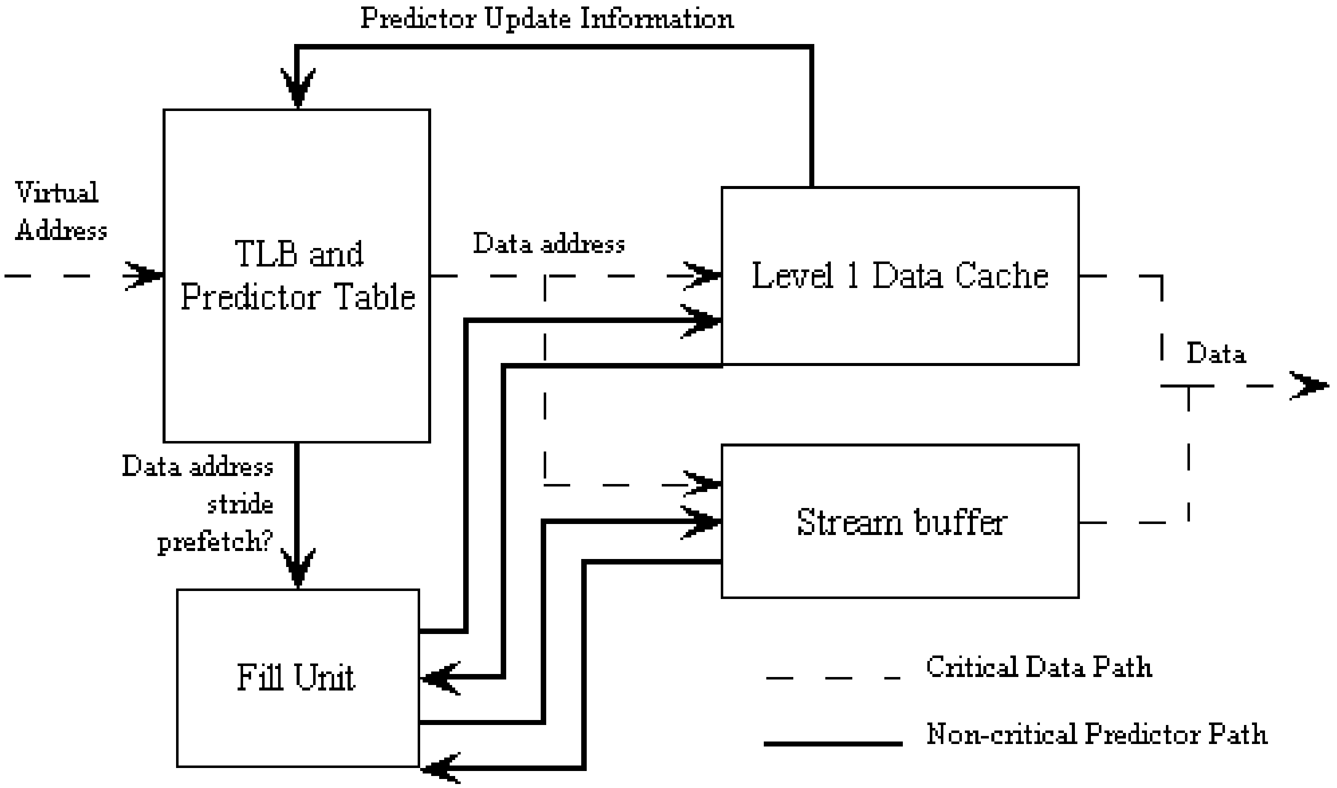
Project Hypothesis

- Memory accesses have patterns, like strides.
  - Localized data accesses have frequently used data with smaller strides.
  - Non-localized data accesses have infrequently used data with larger strides
  - Other access patterns may need to be identified.

Our Scheme of Predictor Based Stream Buffer

- Highlights
  - Decoupled predictors from stream buffer
  - Our stream buffer is a modified standard stream buffer.
  - A group of predictors with each predictor responsible for prediction falling within a “superline” (a consecutive range of addresses across several blocks)
  - Active predictors implemented in a table similar to TLB
  - Predictor makes prefetching decision and choose L1 data cache and stream buffer as prefetching target
  - No data stored in stream buffer is promoted to L1 data cache.

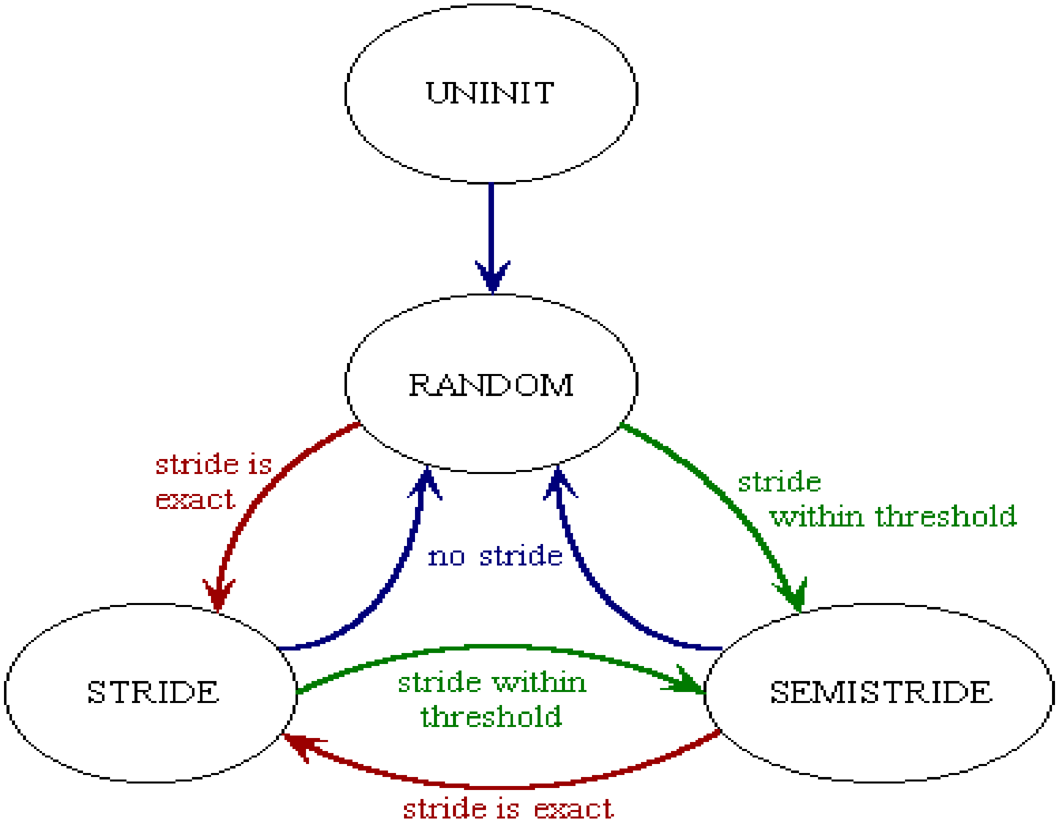
Block Diagram



Predictors

- Decoupled predictors based upon data addresses
- Predictor states
  - Non-predictable states: UNINIT and RANDOM
  - Predictable states: STRIDE and SEMISTRIDE
  - Open to new states for new access patterns
- Look up and update a predictor on each memory access
- Predicts to fetch into stream buffer for long strides and into L1 data cache for short strides

Predictor State Transition Diagram



Predictor Table

- Virtual address indexed table
- A structure similar to TLB or even may be implemented inside TLB
- It hosts a number of active predictors.

Our “Stream Buffer”

- A small fully associative buffer
- Replacement algorithms
  - Track the latest accessed line
  - Kick out the next line from the latest accessed one
- Stream buffer is orthogonal with L1 data access

Performance Comparison (IPC)

Benchmark	4k L1	4k L1 with SB (12)	8k L1
vpr	0.9326	0.9337	0.9341
mcf	1.0839	1.1163	1.1150
ammp	0.7477	0.7614	0.7495

Prefetch Distribution

BM	stride≤8		8<stride<32		stride≥32	
	Req. PF	PF Done	Req. PF	PF Done	Req. PF	PF Done
vpr	4,196,590	49,124	1,956,374	142,838	4,920,951	255,372
mcf	16,460,868	7,530,754	9,239,329	41,531	2,504,918	1,756,205
ammp	9,248,365	28,294	438,057	2,718	3,339,133	2,589,741

Cache and Stream Buffer Performance

Benchmark	Dec L1 Misses	Stream Hits	SB Prefetches	L1 Prefetches
vpr	39,976	190,001	398,210	49,124
mcf	546,259	538,237	3,234,279	7,530,754
ammp	16,584	461,067	2,615,366	28,294

Data Analysis

- Distributions of strides exist in application but show different characteristics in different applications.
- Gained performance is modest but comparable to the increase of the L1 cache size.
- Performance increase correlates to prefetch prediction effciency and prefetch efficiency.

Research Topics Left

- Implementation of better predictors like Markov predictors
- Implementation of PC-coupled predictors and compare with current performance.
- Redesign the structure of stream buffer and see performance changes
- Search for more data access patterns and add new predictor states

Conclusions

- There exist localized and non-localized data accesses.
- Localized data tend to have smaller strides.
- The distribution of stride sizes depend upon applications.
- Predictor-based stream buffer can increase performance.