Blasting Through The Front-End Bottleneck With Shotgun

Rakesh Kumar, Boris Grot, Vijay Nagarajan
The Front-End Problem

Traditional and emerging server applications

- Deep SW stacks, complex functionality
- Huge instruction working set size
  - Multiple megabytes
  - Growing 25% per year at Google [Kanev ISCA’15]

Frequent L1-I and BTB Misses

- Working sets don’t fit in latency-critical components
Front-End Problem #1: L1-I Misses

L1-I

Last Level Cache (LLC)

Core

Miss

30-40 cycles
Front-End Problem #2: BTB Misses

Execution Cycles

Wrong Path Detected
Overcoming the Front-End Bottleneck: What are the Options?

Bigger or multi-level caches, BTB
- High access latency hurts performance

Prefetching
- No impact on L1-I & BTB access latency and area
Outline

Motivation

Existing front-end prefetching approaches
- Temporal Stream: High storage 😞
- BTB-directed: Low Performance 😞

Shotgun: Low Storage and High Performance 😊

Summary
Temporal Stream Prefetching

Principle: Record and Replay

Confluence [MICRO’15]

Prohibitive metadata storage costs
BTB-directed Prefetching

Idea: Instead of recording, construct the control flow

Control flow construction for prefetching without metadata cost
BTB-directed Prefetching

No prefetching under a BTB miss: A common case in server workloads with large instruction footprints

Prefetching stalls on BTB miss

Boomerang [HPCA’17]
Our Goal

Temporal Stream Prefetchers
- Performance: ☺️
- Storage: 😞

BTB-directed Prefetchers
- Performance: ☺️
- Storage: ☺️
**Problem:** Conventional BTB cannot accommodate the branch working set of server workloads

- BTB misses stall prefetching

**Objective:** Improve BTB control flow coverage

**Approach:** Rethink BTB organization for **prefetching** leveraging S/W behavior
Understanding Control Flow Behavior

Global Control Flow
- Control flow between distinct code regions (e.g. functions).
- Comprised of unconditional branches
  - calls, returns, traps,…

Local Control Flow
- Inside a code region
- Comprised of conditional branches
Global Control Flow Insight

Small working set size

Global control flow fits in a practical-sized BTB
Local Control Flow Insight

High spatial locality within a code region

Local control flow affords a compact spatial representation
Mapping Control Flow to a BTB

Idea: Control flow footprint can be represented as

- Global control flow: unconditional branches
- Spatial encoding (footprint) of local control flow around each unconditional branch target
From Idea to Microarchitecture

Unconditional branches + target region footprints enable high-coverage L1-I prefetching
From Idea to Microarchitecture

BTB

BTB for conditional branches (C-BTB)

Miss

L1-I

C-BTB: Proactively filled with conditional branches of active regions.

Last Level Cache (LLC)
Shotgun: A Specialized BTB Organization for Control Flow Delivery

**BTB for unconditional branches (U-BTB)**
- Bulk of BTB storage budget goes to U-BTB (unconditional branches + spatial footprints)
- Small size: nearly 20x less storage than U-BTB

**BTB for conditional branches (C-BTB)**

**Return Instruction Buffer (RIB)**
- Storage optimization for return instructions: minimal metadata → avoid placement in U-BTB
Evaluation Methodology

- 16-core CMP, 8MB LLC
- L1-I: 32KB
- BTB: 2K-entry
  - Equivalent storage budget for Shotgun
- Workloads: Enterprise and scale-out (databases, web search, media streaming, web serving)
- Evaluated prefetchers:
  - Temporal stream prefetcher: Confluence [MICRO’15]
  - BTB-directed prefetcher: Boomerang [HPCA’17]
  - Shotgun
Performance Comparison

- Confluence
- Boomerang
- Shotgun
- Nutch
- Streaming
- Apache
- Zeus
- Oracle
- DB2
- Gmean
Shotgun Summary

• Front-end bottleneck critical in servers
  – Prior work: trades off between storage and performance

• Control flow behavior-guided BTB design
  – Global control flow fits in a practical-sized BTB
  – Uses the BTB to map the instruction working set using control flow behavior insights
    • Enables highly effective front-end prefetching
    • Erases the performance gap between metadata-rich and metadata-free front-end prefetchers

High performance core front-end without costly metadata