Karlsruhe Institute of Technology

Improving Cloud Application Performance with Simulation-Guided CPU State Management

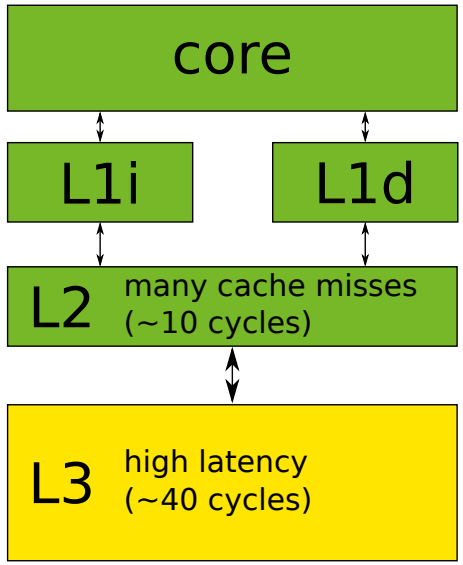
Mathias Gottschlag, Frank Bellosa | April 23, 2017

KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT) - OPERATING SYSTEMS GROUP

Motivation

- Scale-out/server workloads: Complex workloads, large working set

- Accesses miss L2 and hit slow L3
- Result:
  - Half of all cycles spent waiting for memory
    (Kanev et al.: Profiling a Warehouse-Scale Computer, ISCA'15)
  - Need prefetching to bring data closer to the core



- Only very simple hardware prefetchers available

$\Rightarrow$ **Need for efficient software prefetching**

Prefetching

Existing software solutions (e.g., Helper Threads, Call Graph Prefetching):

1. Developer profiles the application
2. Compiler inserts prefetching code
3. Application is deployed

Problems

- Addresses not known at compile time (address calculation overhead)
- Not known whether prefetching is beneficial
- Of limited use for the OS itself (workload not known in advance)

Prefetching

Existing software solutions:
(e.g., Speculative Precomputation, ISCA'01, Call Graph Prefetching, HPCA'01)

1. Developer profiles the application
2. Compiler inserts prefetching code
3. Application is deployed

Problem: Workload not known in advance

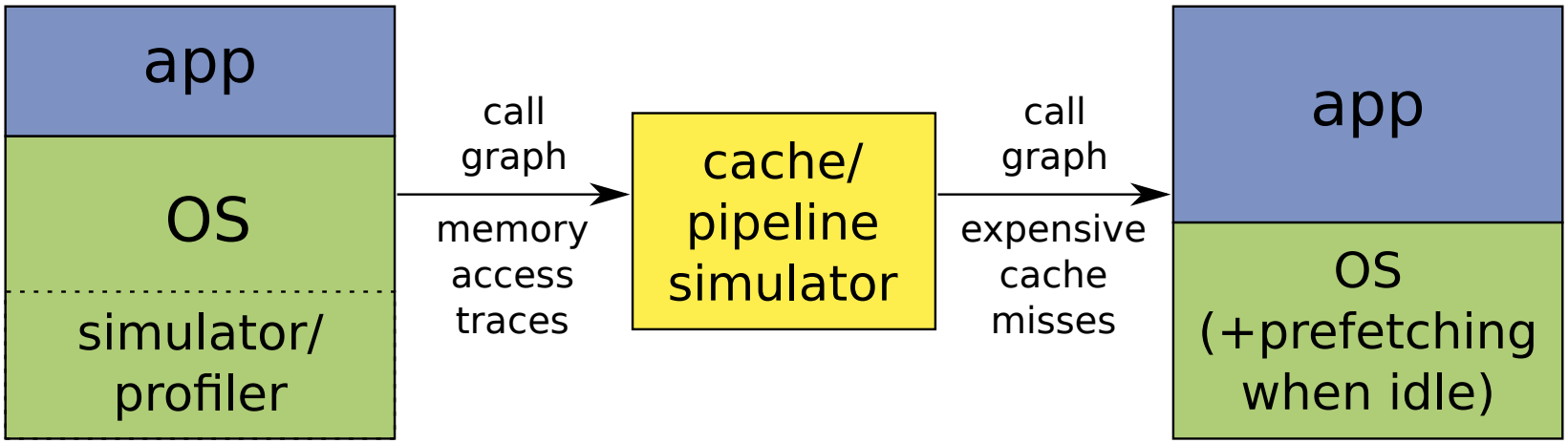
- Large working set: Many misses in the OS

- Small working set: No misses in the OS

$\Rightarrow$ Need to know whether accesses miss the cache

Approach

Problem: Cache misses depend on OS, workload, and microarchitecture



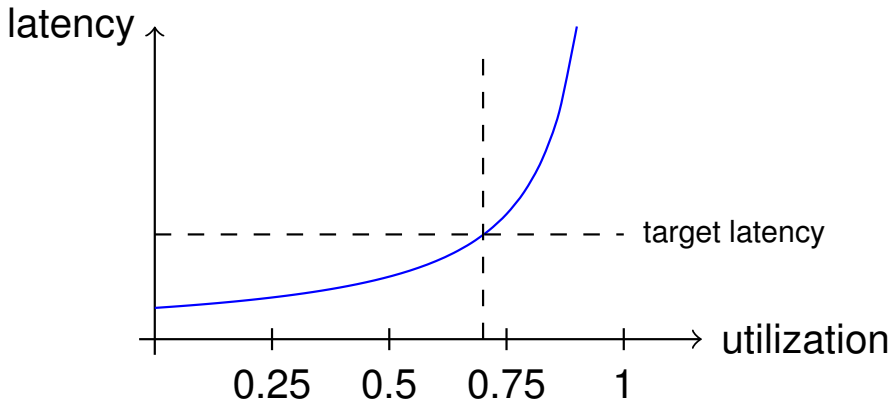
1. Temporary execution on simulator to record memory access patterns
2. Cache simulator to identify data worthwhile to be prefetched
3. Use resulting address list for prefetching

Challenge: Accurate models of existing hardware

Prefetching at Idle Time

Server systems: Usually not 100% utilized

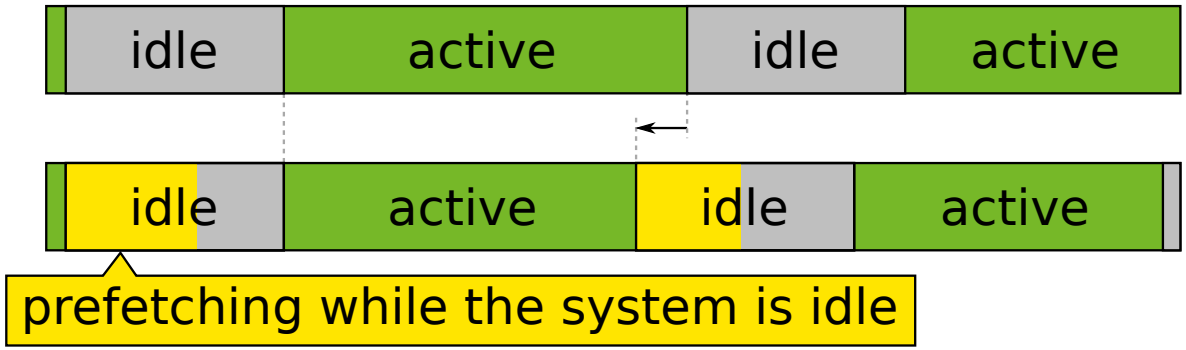
- Spare capacity for peak load
- Latency depends on utilization



Why not make use of the resulting idle time?

Prefetching at Idle Time

- Idea: Use idle time to prefetch next process
- Next process can often be predicted
  - One latency-critical service per core
  - Predictable network architectures
    (M. Gottschlag, F. Bellosa, Reducing Response Time with Preheated Caches, ROME'16)



- Experiments: Good results when prefetching from DRAM into L2 cache
- **Challenge: Interaction with aggressive hardware prefetchers**

Summary

- Problem: Compiler-based approaches cannot fully utilize runtime information
- Result: Missed opportunity for optimization

- Approach:
  - Runtime simulation-based profiling
    $\Rightarrow$ Collect precise cache miss information
  - Prefetching while the system is idle
    $\Rightarrow$ Hide prefetching cost

- Challenges:
  - Modelling existing caches and prefetchers
  - Interaction between software and hardware prefetchers

References

- Ferdman, M. et al.: Clearing the Clouds: A Study of Emerging Scale-Out Workloads on Modern Hardware
- Kanev, S. et al.: Profiling a Warehouse-Scale Computer
- Lee, J., Kim, H., Vuduc, R.: When Prefetching Works, When It Doesn't, and Why
- Collins, J. D. et al.: Speculative Precomputation: Long-Range Prefetching of Delinquent Loads
- Annavaram, M., Patel, J. M., Davidson, E. S.: Call Graph Prefetching for Database Applications.
- Gottschlag, M., Bellosa, F.: Reducing Response Time with Preheated Caches

Scope

- Target:
  - Long-running
  - Latency-critical
  - Networked
- Out of scope: Runtime ASLR
- Need to figure out interaction with garbage collectors
  - Copying collectors problematic
  - Most long-lived objects expected to be at fixed location

Evaluation

Baseline: Existing software prefetching

- Software prefetching benefits from runtime information?
- Augment existing approaches with prefetching during idle time?

Targeted workloads:

- Server and scale-out applications
- CloudSuite, TPC-C, ...

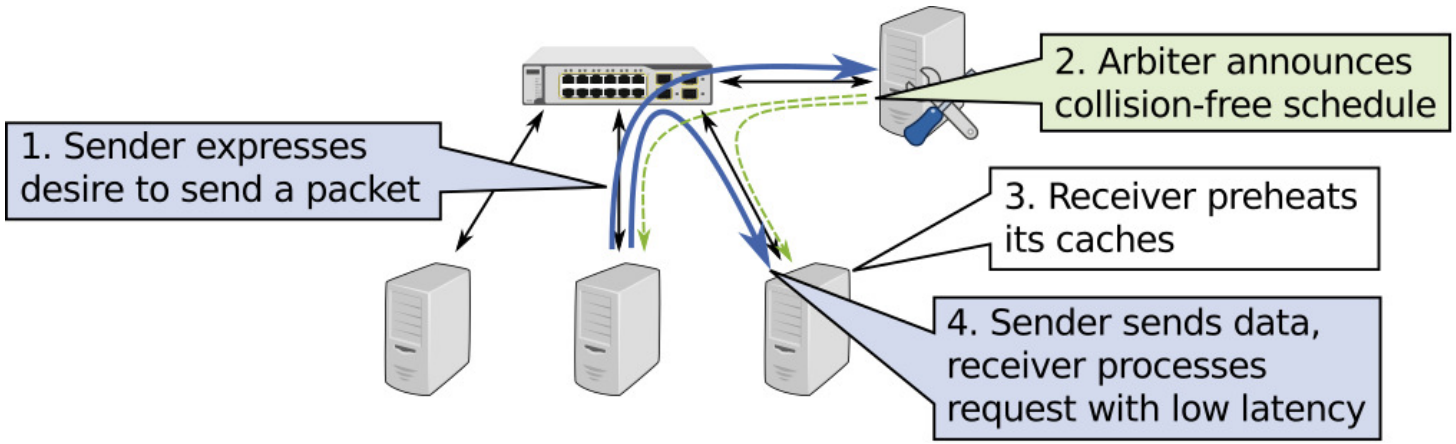
Current State

- Network packet announcements based on Fastpass
  - Announcements $\sim 50 \mu s$ in advance
  - Next process can be predicted
- Early prototype:
  - Runtime profiling with Intel PEBS
  - Prefetching does not yet yield expected results
- Problems identified:
  - Conflicts with hardware prefetchers
  - PEBS traces incomplete
- Next step: Determine benefit in controlled environment
  $\Rightarrow$ Simulator

Predictable Network Architectures

- Problem: Predict next incoming network request
- Basis: Deterministic low-latency networks with central arbiter
  (example: Fastpass, SIGCOMM'14)
- Central arbiter has global view of the network

➜ Arbiter can announce future incoming network packets



(M. Gottschlag, F. Bellosa, Reducing Response Time with Preheated Caches, ROME'16)