Cache Optimization for Iterative Numerical Codes Aware of Hardware Prefetching

PARA 2004, 22 June 2004
Lyngby / Copenhagen, Denmark

Josef Weidendorfer, Carsten Trinitis

Technische Universität München,
Germany

Overview

- Introduction
  Scope, Motivation
- Simulating Hardware Prefetching
  Technique, First Results
- Block Prefetching
  Strategy, Benchmark Results
- Future Work

Introduction - Scope

- Project DiME
  - Research on Cache-Optimizations for Sequential PDE Solvers (Multigrid)
    - Transformations improving Locality
  - Approach (in developed library):
    - Use standard Techniques like Blocking/Padding
    - Best Parameters are searched at Compile Time for a given Platform

Introduction – Motivation

- Problem:
  Standard Techniques not as successful as expected on modern CPUs
- Caches are
  - not only used to exploit Locality
  - but also to buffer prefetched Data
    = Hide Latency of Main Memory Accesses by Overlapping with Computation

Introduction – Motivation (Cont'd)

- Prefetching can be done in
  - SW: Provide Hints for Cache Logic
    - About Address & About Type of Locality
  - HW: Prediction Algorithm for Future Accesses
    - Can go wrong (Reason for our Ineffectiveness ?)
- Idea for further Cache Optimizations
  - Complement HW by SW Prefetching
  - Guide HW Prefetch Algorithm not to go wrong
    E.g.: Abort Stream Prefetching in time

Introduction - How ?

- Tool hinting at bad HW Prefetch Behavior
  - Cache + HW Prefetch Simulation
  - Simple Stream Detection Algorithm Should be covered by most real HW Prefetchers
  - Goal: Allow for Insertion of SW Prefetching iff
    - Stream Prefetcher would go wrong
    - Data Not detectable by Stream Prefetcher
- Develop Prefetching Techniques
  - Promising: “Block Prefetching”

HW Prefetch Simulation

- Based on Execution Driven Simulation
  - Uses Runtime Instrumentation (Valgrind)
  - No Need for Binary Preparation
  - Cache and Prefetch Simulation on the Fly
- Our Current Prefetch Algorithm
  - Activated on L2 Misses (Arguable)
  - Detects Streams (16) after 3 Seq. Cache Line Loads
  - Stops at 4 kB Boundary (HW Prefetcher for physically tagged Cache has to stop at VM Page)

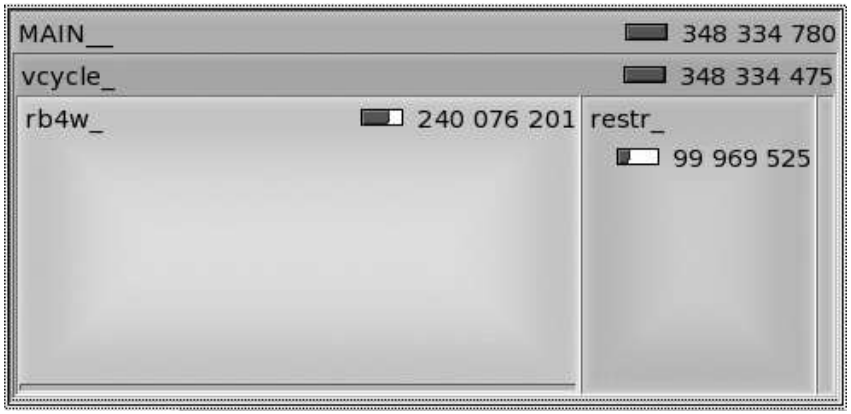
HW Prefetch Simulation (Cont'd)

- Application
  - Cache-Optimized regular 3D Multigrid on 129^3
- First Simulation Results (Real: Pentium-M)

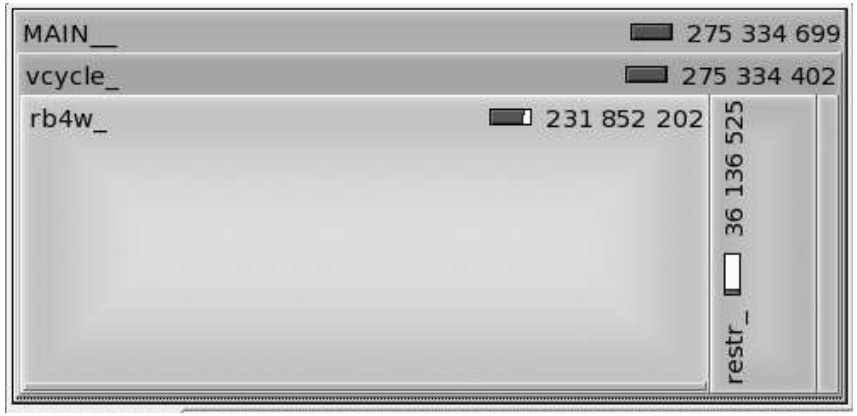
	Simulated			Real		
	L2 Misses		Pref.	L2 Lines In		Pref.
	Pf. Off	Pf. On	Requests	due to Misses	due to Pref.	Requests
Summary	361	277	110	241	130	373
RB4W	233	226	-	201	47	270
RESTR	108	37	-	28	76	92

HW Prefetch Simulation (Cont'd)

- Treemap Visualization for sim. L2 Misses
  (Rectangle Area linear to inclusive Cost)



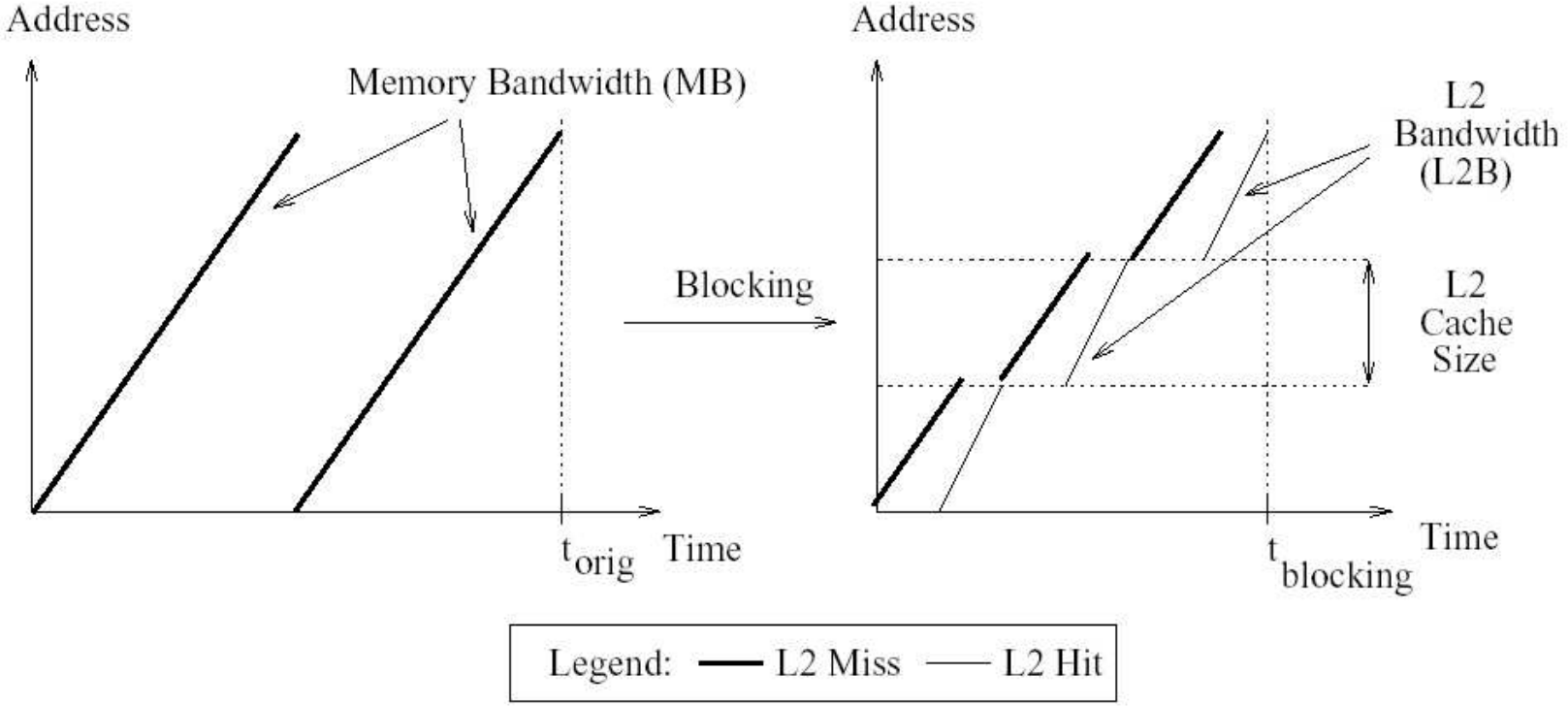
Without Prefetching



With Prefetching

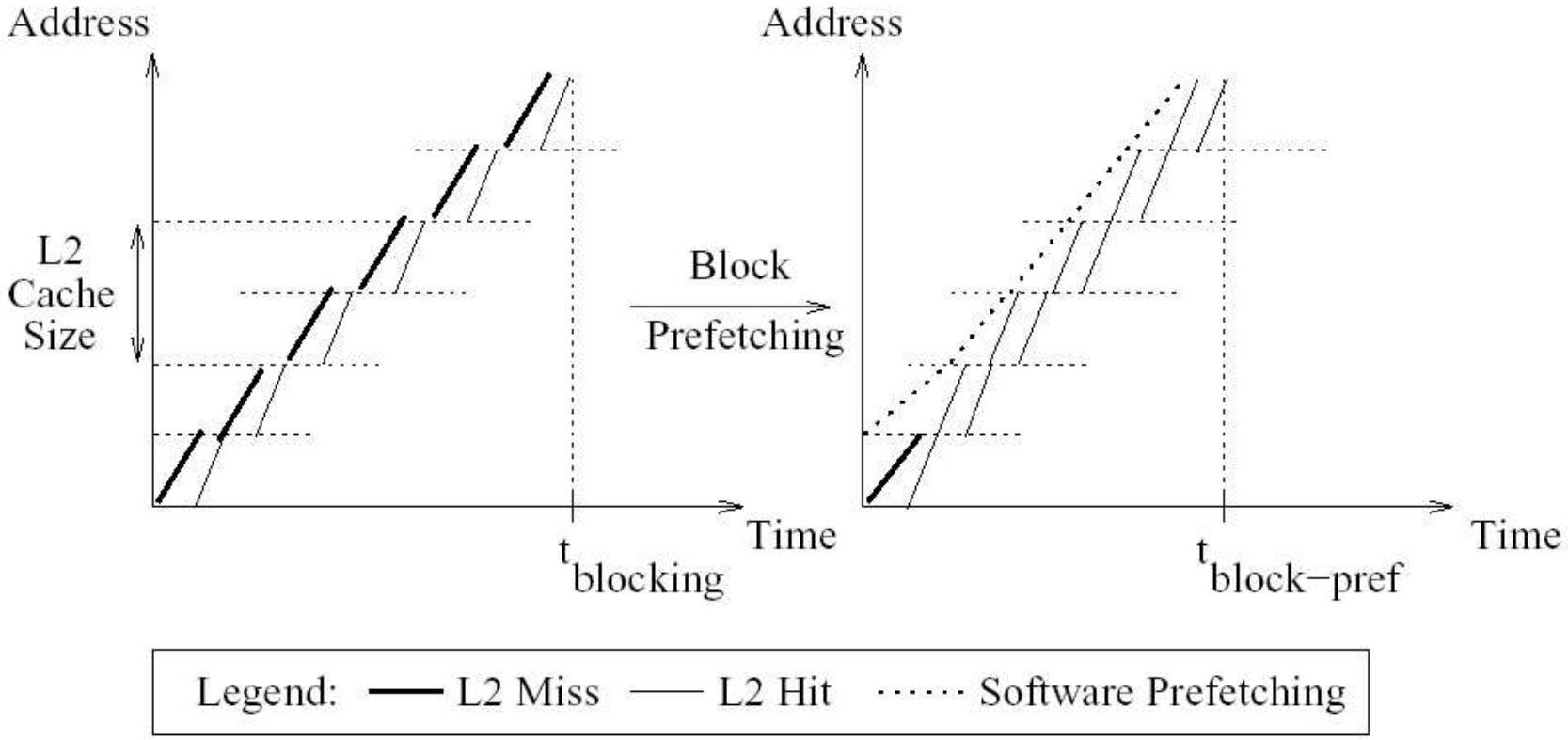
Block Prefetching - Strategy

- Twofolded 1D Blocking Transformation



Block Prefetching – Strategy (Cont'd)

- Half block Size / Add Software Prefetching



Block Prefetching - Results

32% Improvement

N	Without Block Pf. 1	2	3	10	With Block Pf. 1	2	3	10
1 Flop/Load								
Runtime [s]	5.52	3.92	3.45	2.73	4.89	2.67	2.50	2.44
MFlop/s	194	274	311	393	220	402	430	440
Netto [GB/s]	1.48	1.04	0.79	0.30	1.68	1.53	1.09	0.34
Lines In b/o Misses [MEv.]	4.2	2.1	1.4	0.4	134	67	45	13
Lines In b/o Pref. [MEv.]	130	65	44	13	0.45	0.54	0.27	0.15
2 Flops/Load								
Runtime [s]	7.20	5.24	4.81	3.23	6.86	4.29	3.60	2.84
MFlop/s	298	410	446	665	313	501	597	756
Netto [GB/s]	1.14	0.78	0.57	0.25	1.19	0.95	0.76	0.29

Measured on Pentium-M 1.4 GHz

Future Work

- Simulation
  - Other Prefetch Algorithms

Thanks for Listening!
Any Questions?

  - Combine with Padding Parameter Search (Conflicts Possible !)