多核共享缓存下的编程优化和正确性
Program Behavior in Shared Cache: Performance and Correctness

丁晨 Chen Ding

美国纽约州私立罗切斯特大学

计算机科学系教授

Professor

University of Rochester

2014 DragonStar Course at University of Science and Technology of China

Lecture 1: Why Cache 缓存的存在

Necessity of memory hierarchy. Sec. 1.1, Ulrich Meyer, Peter Sanders, and Jop F. Sibeyn, editors. *Algorithms for Memory Hierarchies, Advanced Lectures*, volume 2625 of Lecture Notes in Computer Science. Springer, 2003.

Memory bandwidth bottleneck. Sec. 1, Chen Ding and Kennedy, "Improving Effective Bandwidth through Compiler Enhancement of Global Cache Reuse," *Journal of Parallel and Distributed Computing*, Volume 64, Issue 1, January 2004, Elsevier Press, pages 108--134 .

Performance metrics

footprint, reuse distance, fill time, miss ratio. Xiaoya Xiang, Chen Ding, Bin Bao and Hao Luo, "A Higher Order Theory of Cache Locality", in Proceedings of The Symposium on Architectural Support for Programming Languages and Operating Systems, March 2013.

AMAT and APC. Xian-He Sun and Dawei Wang. APC: a performance metric of memory systems. SIGMETRICS Performance Evaluation Review, 40(2):125–130, 2012.

Sec. 1--2, "Program Locality Analysis Using Reuse Distance ", Yutao Zhong, Xipeng Shen, and Chen Ding, ACM Transactions on Programming Languages and Systems, Volume 31, Number 6, August 2009, pages 1--39.

Cache hardware. Preface and Chap. 1, Rajeev Balasubramonian, Norman Jouppi, Naveen Muralimanohar, Multi-CoreCache Hierarchies, Synthesis Lecturess on Computer Architecture #17, Morgan Claypool Publishers, 2011.

Software defined cache -- memcached. Atikoglu et al. Workload Analysis of a Large Key-Value Store. SIGMETRICS 2012.

http://www.cs.rochester.edu/drupal/program-behavior-shared-cache-performance-and-correctness

Lecture 2: Footprint Theory of Locality 程序局部性的足迹理论

"多核程序交互理论及应用", 丁晨, 袁良, *计算机工程与科学*, Volume 36, Number 1, January 2014, pages 1--5.

"Performance Metrics and Models for Shared Cache (共享缓存性能的度量与分析)", Chen Ding (丁晨), Xiaoya Xiang (向晓娅), Bin Bao (包斌), Hao Luo (罗昊), Ying-Wei Luo (罗英伟), and Xiao-Lin Wang (汪小林), Journal of Computer Science and Technology, 2014,V29(4): 692-712.

Lecture 3: Locality Optimization 程序局部性优化的概述和举例

Five dimensiions of locality.

Sec. 6, "Program Locality Analysis Using Reuse Distance ", Yutao Zhong, Xipeng Shen, and Chen Ding, ACM Transactions on Programming Languages and Systems, Volume 31, Number 6, August 2009, pages 1--39.

"Performance Metrics and Models for Shared Cache (共享缓存性能的度量与分析)", Chen Ding (丁晨), Xiaoya Xiang (向晓娅), Bin Bao (包斌), Hao Luo (罗昊), Ying-Wei Luo (罗英伟), and Xiao-Lin Wang (汪小林), Journal of Computer Science and Technology, 2014,V29(4): 692-712.

(added 7/4) "On-the-Fly Elimination of Dynamic Irregularities for GPU Computing", Eddy Z. Zhang, Yunlian Jiang, Ziyu Guo, Kai Tian, Xipeng Shen, the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems, 2011.

(added 7/8) "Defensive Loop Tiling for Shared Cache", Bin Bao and Chen Ding, in Proceedings of on Code Generation and Optimization, February 2013. (slides, video)

Lecture 4: Fundamentals of Shared-Memory Synchronization 共享内存同步的基本原理

First 3 chapters and 8.2 in Michael L. Scott, *Shared-Memory Synchronization*, Synthesis Lecturess on Computer Architecture, Morgan Claypool Publishers, 2013.

(added 7/8) Safe (Hint based) Parallel Programming: "Software Behavior Oriented Parallelization", Chen Ding, Xipeng Shen, Kirk Kelsey, Chris Tice, Ruke Huang, and Chengliang Zhang, in Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation, San Diego CA, June 2007.

Lecture 5: Collaborative Cache Management and Optimization 软硬件协同管理和优化缓存

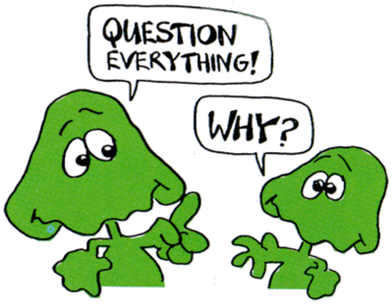
Xiaoming Gu (BS at USTC 2003, MS at ICT 2006, PhD at Rochester 2013), 谷晓铭(中科大计算机系本科2003，中科院计算所硕士2006，罗切斯特大学博士2013) Optimal Collaborative Caching: Theory and Applications. Ph.D. Dissertation, 2013.

(Guest lecture) Professor Song Jiang (韦恩州立大学江松教授，前中科大计算机系老师)

LIRS algorithm. Jiang et al. *IEEE Transactions on Computers*, 2005 and 2007. SIGMETRICS 2002.

Introduction to 大数据在互联网数据中心的管理和计算 (中科院深圳先进技术研究院龙星课程)

Please Ask Questions

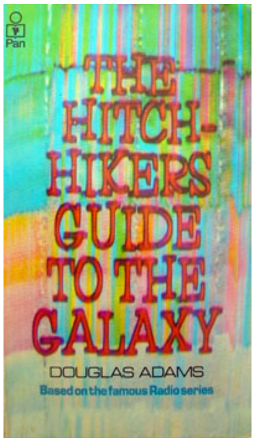


- To assert you as you
- To network
- To share
- To have a good time
- So I can do better than a lecture video

http://www.xtrasite.co.nz/monsta/gallery/early/question.gif

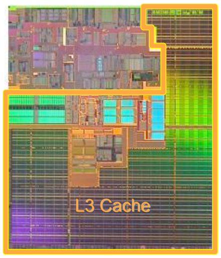
缓存必要性?

- "Nothing travels faster than the speed of light with the possible exception of bad news, which obeys its own special laws." Douglas Adams The Hitchhiker's Guide to the Galaxy


Pan
THE HITCH-HIKERS GUIDE TO THE GALAXY
DOUGLAS ADAMS
Based on the famous Radio series

Three problems:
latency/bandwidth and Matthew Hertz's beer
capacity and Trishul Chilimbi's cliff
sharing Chen's Platform

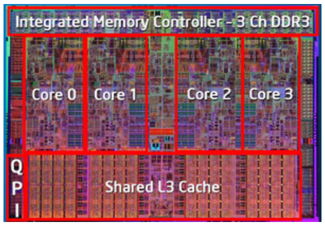
Cache System



- Cache
  - which most transistors are used for
  - where most memory accesses happen
  - managed by priority/usage
- Shared cache
  - available cache is variable
  - throughput/stability/fairness/QoS
  - sequential/parallel code

Madison Itanium 2, 2002
slide 3, Anant Agrawal, MIT 6.975 Fall 2007

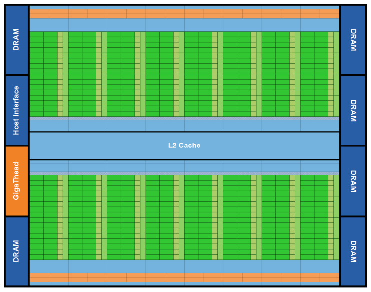
Hierarchical Cache
Parallel Access
Constant
Interaction



Whitepaper

NVIDIA's Next Generation
CUDA™ Compute Architecture:

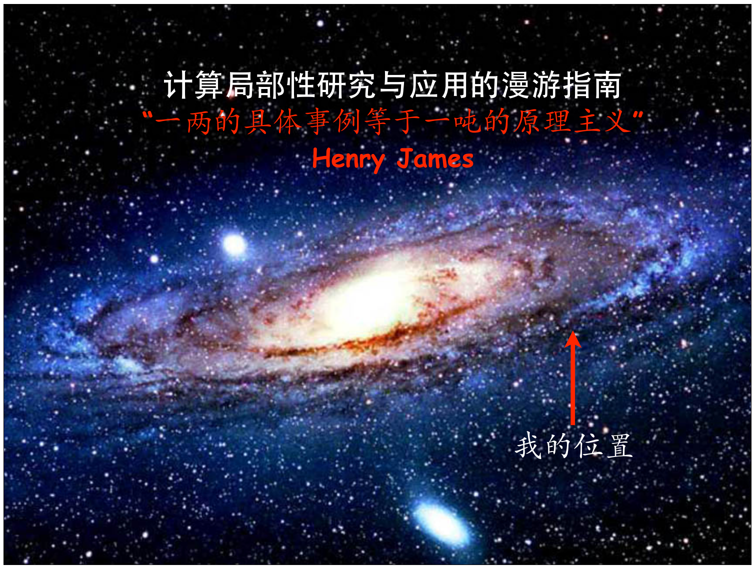
Fermi™



GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Special Function Units (SFUs) / SM	2	2	4
Warp schedulers (per SM)	1	1	2
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit

计算局部性研究与应用的漫游指南

"一两的具体事例等于一吨的原理主义"
Henry James



量化， 指标， 比例

- 直到你定义一个度量
  - 科学的起步
  - 工程的先决条件
- "没有测量就没有提高"
  - "You can't improve what you can't measure"
- 有了度量可以
  - 寻找关系
    - e.g. $e = mc^2$
  - 研究趋势
    - e.g. 宇宙膨胀
  - 比较优劣
    - e.g. 程序优化 （明天易青老师的讲座）

The Design of Parallel Memory Systems

Xian-He Sun
Illinois Institute of Technology
Chicago, Illinois
sun@iit.edu

Introduction

RESEARCH DRIVE

Computing Become Data Intensive

- Simulation, visualization, data mining, information retrieval, etc.

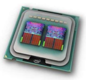
Data requirements for selected INCITE applications at ALCF

PI	Project	On-Line Data	Off-Line Data
Lamb, Don	FLASH: Buoyancy-Driven Turbulent Nuclear Burning	75TB	300TB
Fischer, Paul	Reactor Core Hydrodynamics	2TB	5TB
Dean, David	Computational Nuclear Structure	4TB	40TB
Baker, David	Computational Protein Structure	1TB	2TB
Worley, Patrick H.	Performance Evaluation and Analysis	1TB	1TB
Wolverton, Christopher	Kinetics and Thermodynamics of Metal and Complex Hydride Nanoparticles	5TB	100TB
Washington, Warren	Climate Science	10TB	345TB
Tsigelny, Igor	Parkinson's Disease	2.5TB	50TB
Tang, William	Plasma Microturbulence	2TB	10TB
Sugar, Robert	Lattice QCD	1TB	44TB
Siegel, Andrew	Thermal Striping in Sodium Cooled Reactors	4TB	8TB
Roux, Benoit	Gating Mechanisms of Membrane Proteins	10TB	10TB

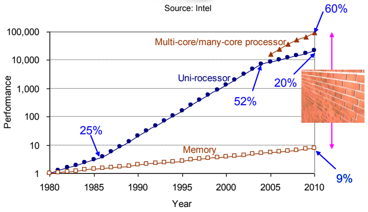
Source: R. Ross et. al., Argonne National Laboratory

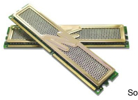
The Memory-wall Problem

- Processor performance increases rapidly
  - Uni-processor: ~52% until 2004, ~25% since then
  - New trend: multi-core/many-core architecture
    - Intel TeraFlops chip, 2007
  - Aggregate processor performance much higher
- Memory: ~9% per year
- Processor-memory speed gap keeps increasing



Source: Intel





Source: OCZ

Addressing the HPC Data Challenges

Trends indicate that the "data tsunami" and "memory-wall" will continue, waiting for miracle is not an answer

Big-Data problem is a HPC problem:

- Data access & Interface

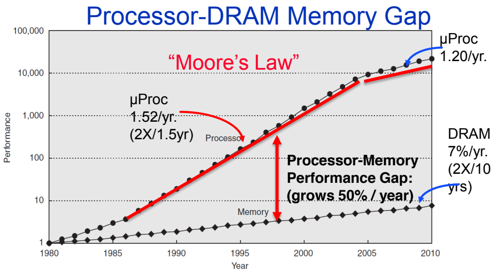
Need **rethinking** from the **data-centric** point-of-view in:

- **Understanding** the system, application, and algorithm relevant to data access
- **Optimizing** current systems
- **Developing** new system architectures
- Developing **integrated solutions**
  - Algorithm, programming model, system, architecture, co-design
  - In situ application-aware data access optimization

Understanding

MEMORY SYSTEM BEHAVIOR

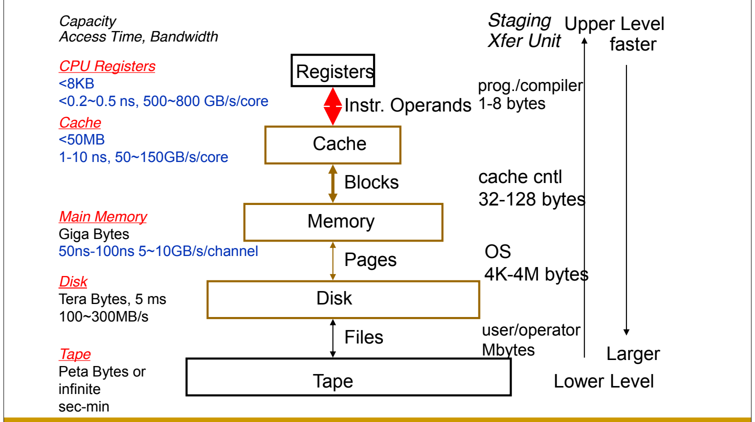
Memory System Performance



- 1980: no cache in micro-processor; 2010: 3-level cache on chip, 4-level cache off chip
- 1989 the first Intel processor with on-chip L1 cache was Intel 486, 8KB size
- 1995 the first Intel processor with on-chip L2 cache was Intel Pentium Pro, 256KB size
- 2003 the first Intel processor with on-chip L3 cache was Intel Itanium 2, 6MB size

Source: Computer Architecture A Quantitative Approach

Improve via Memory Hierarchy



Capacity
Access Time, Bandwidth

CPU Registers
<8KB
<0.2~0.5 ns, 500~800 GB/s/core

Cache
<50MB
1-10 ns, 50~150GB/s/core

Main Memory
Giga Bytes
50ns-100ns 5~10GB/s/channel

Disk
Tera Bytes, 5 ms
100~300MB/s

Tape
Peta Bytes or
infinite
sec-min

Registers

Instr. Operands

Cache

Blocks

Memory

Pages

Disk

Files

Tape

Staging
Xfer Unit

prog./compiler
1-8 bytes

cache cntl
32-128 bytes

OS
4K-4M bytes

user/operator
Mbytes

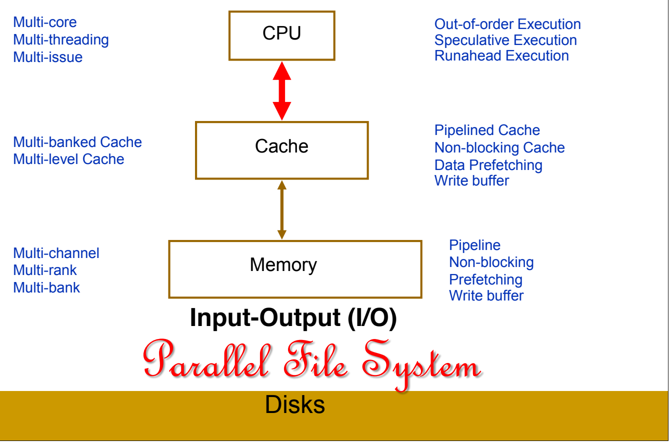
Upper Level
faster

Larger
Lower Level

Improve via Data Access Concurrence

- ❑ The complexity of CPU Design
  - ○ Out-of-order Execution
  - ○ Multithreading technology
  - ○ Speculation mechanisms
- ❑ The complexity of Memory Design
  - ○ Advanced Cache Technologies
  - ○ Allow tens or hundreds of cache accesses to overlap with each other
  - ○ Processor continue execution instructions under multiple cache misses

Solution: Memory Hierarchy & Parallelism



Multi-core
Multi-threading
Multi-issue

CPU

Out-of-order Execution
Speculative Execution
Runahead Execution

Multi-banked Cache
Multi-level Cache

Cache

Pipelined Cache
Non-blocking Cache
Data Prefetching
Write buffer

Multi-channel
Multi-rank
Multi-bank

Memory

Pipeline
Non-blocking
Prefetching
Write buffer

Input-Output (I/O)

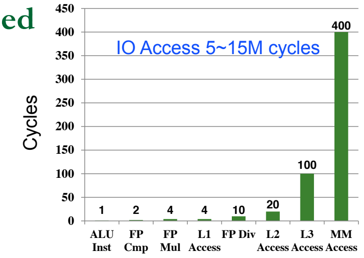
Parallel File System

Disks

Assumption of Current Solutions

- ❑ Memory Hierarchy: Locality
- ❑ Concurrence: Data access pattern
  - ○ Data stream

Extremely Unbalanced Operation Latency



Performances vary largely

Existing Memory Metrics

- ❑ Miss Rate(MR)
  - ○ {the number of miss memory accesses} over {the number of total memory accesses}
- ❑ Misses Per Kilo-Instructions(MPKI)
  - ○ {the number of miss memory accesses} over {the number of total committed Instructions × 1000}
- ❑ Average Miss Penalty(AMP)
  - ○ {the summary of single miss latency} over {the number of miss memory accesses}
- ❑ Average Memory Access Time (AMAT)
  - ○ AMAT = Hit time + MR×AMP
- ❑ Flaw of Existing Metrics
  - ○ Focus on a single component or
  - ○ A single memory access

Missing memory parallelism/concurrency

The Introduction of APC

- Access Per Cycle (APC)
  - APC = A/T
- APC is measured as the number of memory accesses per memory active cycle or Access Per Memory Active Cycle (APMAC)
- Benefits of APC (APMAC)
  - Separate memory evaluation from CPU evaluation
  - Each memory level has its own APC value
  - A better understanding of memory system as a whole, and at each layer
  - A better understanding of the match between computing capacity and memory system performance

X.-H. Sun and D. Wang, "APC: A Performance Metric of Memory Systems", ACM SIGMETRICS Performance Evaluation Review, Volume 40 , Issue 2, 2012.

APC Measurement

- The difficulty is measuring the total cycle T
  - Hundreds of memory accesses co-exist the memory system
- Measure T based on the ***overlapping mode***
  - When there are several memory accesses co-existing during the same clock cycle, *T* only increases by one
  - Measure the concurrence
  - Measure the concurrence at each level
- Hardware cost: one bit
- **Concurrence** and **Data-Centric** view

Exhausted Testing

- With different benchmarks, and with different configurations
- With advanced cache technologies
  - Non-block cache
  - Pipelined cache
  - Multi-port cache
  - Hardware prefetcher
- With single core or multicore
- **APC always has the highest CC values among all the memory metrics**

D. Wang, X.-H. Sun "Memory Access Cycle and the Measurement of Memory Systems", IEEE Transactions on Computers, (May-June) 2014

APC and C-AMAT Applications

- Provide a new way to measure and analyze the contribution of memory concurrence
- Provide new approaches to reduce memory access delay
- Reveal the importance of memory parallelism and its relation to data locality
- Provide a mean to study the matching between memory organization and microprocessor architecture,
- Provide a mean to study the matching between memory organization and a given application
- **Design and Co-Design of Parallel Memory Systems**

Understanding

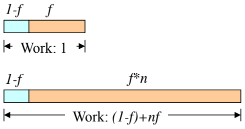
APPLICATION BEHAVIOR

Data Access is Application Dependent

- Conventional algorithm analysis
  - Floating point operation
- Data-centric algorithm analysis
  - Floating point operation
  - Memory requirement
  - Data reuse rate
  - Data access/movement pattern

The Memory-bounded Speedup

- Tacit assumption in Amdahl's law
  - The problem size is fixed
  - The speedup emphasizes time reduction
- Gustafson's Law, 1988
  - Fixed-time speedup model



$$Speedup_{fixed-time} = \frac{Sequential\ Time\ of\ Solving\ Scaled\ Workload}{Parallel\ Time\ of\ Solving\ Scaled\ Workload}$$
$$= (1-f) + nf$$

- Sun and Ni's law, 1990
  - Memory-bounded speedup model

$$Speedup_{memory-bounded} = \frac{Sequential\ Time\ of\ Solving\ Scaled\ Workload}{Parallel\ Time\ of\ Solving\ Scaled\ Workload}$$
$$= \frac{(1-f) + fG(n)}{(1-f) + fG(n)/n}$$

X.-H. Sun, and L. Ni, "Another View of Parallel Speedup," *Proc. of IEEE Supercomputing'90*, NY, NY, Nov.12--Nov.16, 1990.

Contribution of Memory-bounded Speedup

- Data-centric thinking
- Where the memory-bound function **W = G(M)** provide
  - W, the work in floating point operation
  - M, the memory requirement
  - G, the data reuse rate
  - Enough for memory hierarchy, but not concurrence
- Need to find the data access/movement patterns for data access concurrency

- Dense Linear Algebra, M memory, $M^{3/2}$ work
- FFT, M memory, O(M *log*(M)) work
- G(pM) > pW, can lead to large increase in execution time
  - (ex) 10K x 10K matrix factorization: 800MB, 1 hr in uniprocessor with 1024 processors, 320K x 320K matrix, 32 hrs

Data Access Signature: Patterns and Notation

- Comprehensive access pattern classification
- Implemented for I/O and memory access (MPI datatype)

Spatial Patterns

❑Contiguous
❑Non-contiguous
▪Fixed strided
▪2d-strided
▪Negative strided
▪Random strided
▪*k*d-strided
❑Combination of contiguous and non-contiguous patterns

Request size

❑ Fixed
❑ Variable

❑ Small
❑ Medium
❑ Large

Temporal Intervals

❑Fixed
❑Random

Repetition

❑Single occurrence
❑Repeating

I/O Operation

❑Read only
❑Write only
❑Read/write

S. Byna, X.-H. Sun, et. al, "Parallel I/O Prefetching Using MPI File Caching and I/O Signatures," SC'08

I/O Trace Signature

- Description of a sequence of I/O accesses in a pattern
- Form: *{I/O operation, init position, dimension, ([{offset pattern}, {request size pattern}, {pattern of number of repetitions}], [...]), # of repetitions}*

Pattern Signature

- provides a simple description that explains the nature of a pattern
- Form: *{I/O operation, <Spatial pattern, Dimension>, <Repetitive behavior>, <Request size>, <Temporal Intervals>}*

I/O Pattern Detection

- Developed a pattern detection tool
- Five pattern detectors for finding patterns among initial positions, offsets, request sizes, temporality, and repetitions
- Outputs I/O Signature that can be used for prefetching, data layout, and data reorganization

IOSIG: An I/O Characterization Tool

Website: www.cs.iit.edu/~scs/iosig/

Goal: To provide a better understanding of parallel I/O accesses and information to be used for optimization techniques.

Two steps:

Trace collection

- Collects parallel I/O calls of an application
- Does not require any code modification

Trace analysis

- Analyzes the collected information to give a clear understanding of I/O behavior of the application
- Handles trace files in any text-based format

Y. Yin, et.al, "Boosting Application-Specific Parallel I/O Optimization Using IOSIG", in Proc. of IEEE/ACM CCGrid, 2012.

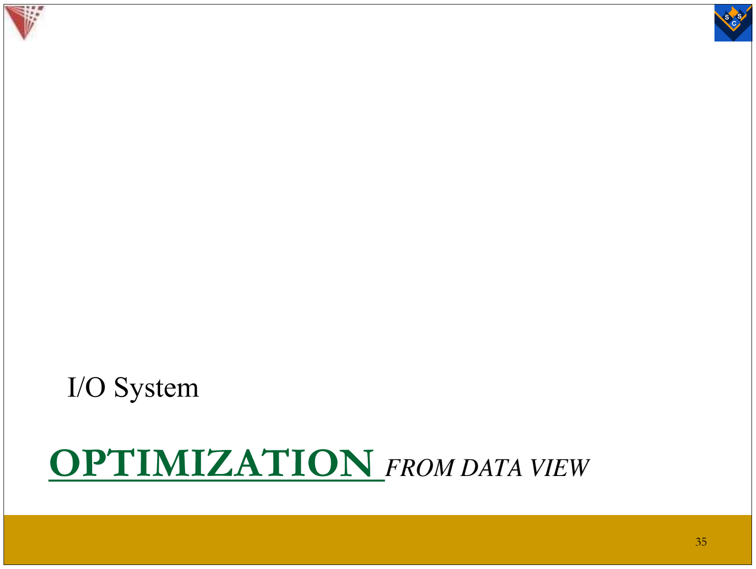
More about IOSIG

- Additional contributions
  - Data access pattern categories
  - Local and global I/O signatures
- Applications

Prefetching	**SC'08**	S. Byna, Y. Chen, X.-H. Sun, R. Thakur, and W. Gropp *Parallel I/O Prefetching Using MPI File Caching and I/O Signatures*
Data Layout	**HPDC11**	H. Song, Y. Yin, Y. Chen, X.-H. Sun, *A Cost-intelligent Application-specific Data layout Scheme for Parallel File Systems*
Data Coordination	**SC11**	H. Song, Y. Yin, X.-H. Sun, et. al, *Server-Side I/O Coordination for Parallel File Systems*
Data Organization	**PDSW11**	J.He, H. Song, X.-H. Sun, Y. Yin, and R. Thakur Pattern-aware File Reorganization in MPI-IO
Data Replication	**IPDPS13**	Y. Yin, J. Li, J. He, X.-H. Sun, and R. Thakur, *Pattern-Direct and Layout-Aware Replication Scheme for Parallel I/O Systems*

Data Compression

- Website: www.cs.iit.edu/~scs/iosig/

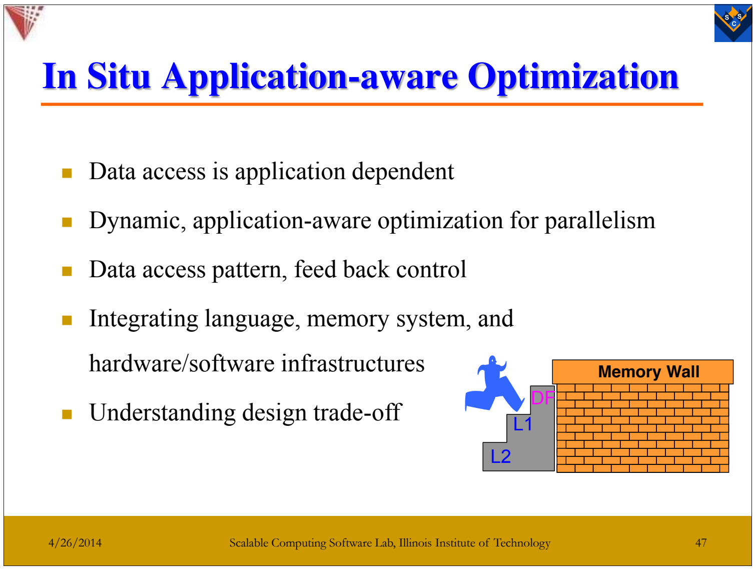


I/O System

OPTIMIZATION *FROM DATA VIEW*

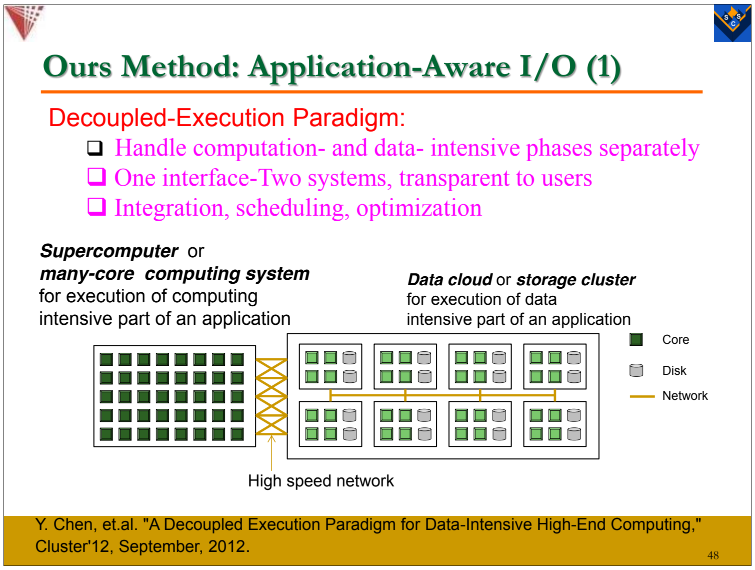
Developing

INTEGRATED SYSTEM DESIGN



In Situ Application-aware Optimization

- Data access is application dependent
- Dynamic, application-aware optimization for parallelism
- Data access pattern, feed back control
- Integrating language, memory system, and hardware/software infrastructures
- Understanding design trade-off



Ours Method: Application-Aware I/O (1)

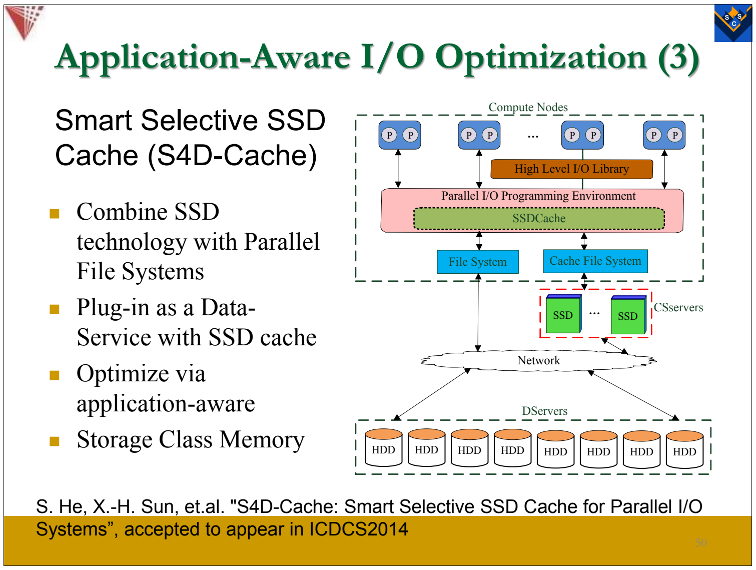
Decoupled-Execution Paradigm:

- ❑ Handle computation- and data- intensive phases separately
- ❑ One interface-Two systems, transparent to users
- ❑ Integration, scheduling, optimization

Supercomputer or ***many-core computing system*** for execution of computing intensive part of an application

Data cloud or ***storage cluster*** for execution of data intensive part of an application

Y. Chen, et.al. "A Decoupled Execution Paradigm for Data-Intensive High-End Computing," Cluster'12, September, 2012.

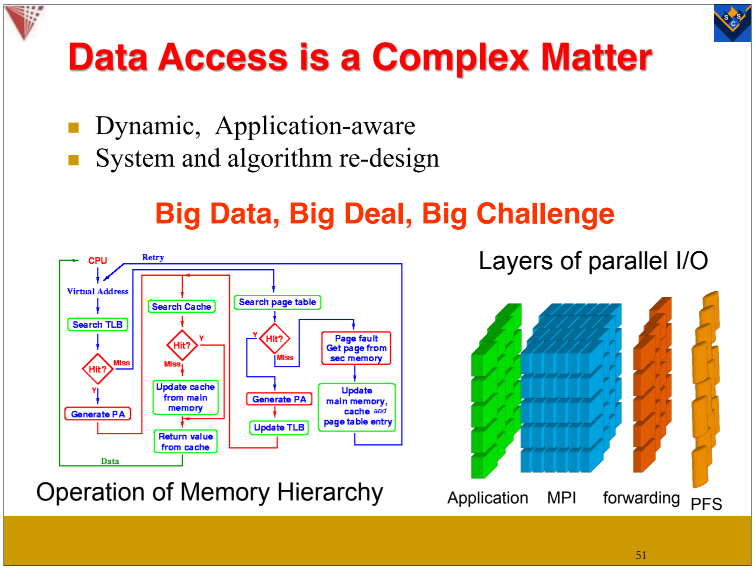


Application-Aware I/O Optimization (3)

Smart Selective SSD Cache (S4D-Cache)

- Combine SSD technology with Parallel File Systems
- Plug-in as a Data-Service with SSD cache
- Optimize via application-aware
- Storage Class Memory

S. He, X.-H. Sun, et.al. "S4D-Cache: Smart Selective SSD Cache for Parallel I/O Systems", accepted to appear in ICDCS2014



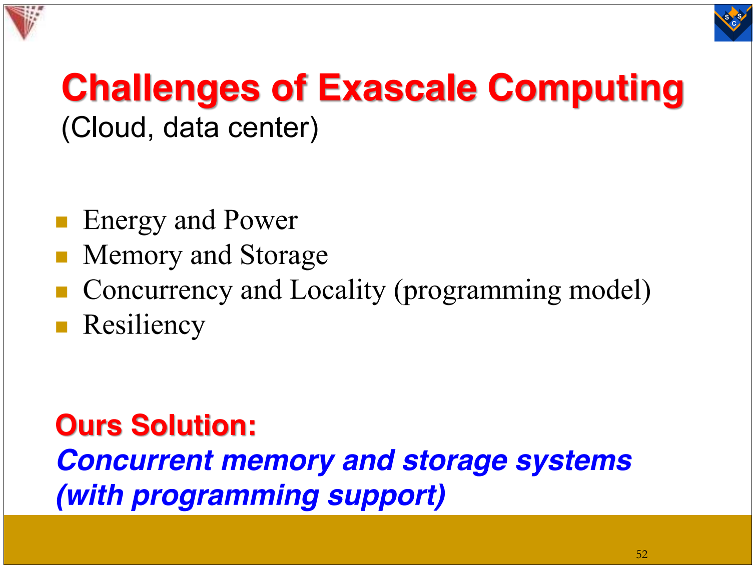
Data Access is a Complex Matter

- Dynamic, Application-aware
- System and algorithm re-design

Big Data, Big Deal, Big Challenge

Operation of Memory Hierarchy

Layers of parallel I/O



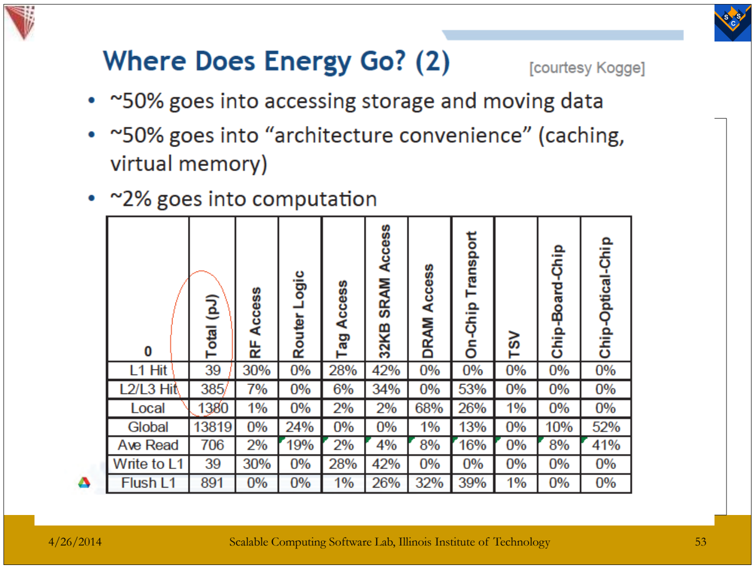
Challenges of Exascale Computing

(Cloud, data center)

- Energy and Power
- Memory and Storage
- Concurrency and Locality (programming model)
- Resiliency

Ours Solution:

Concurrent memory and storage systems (with programming support)

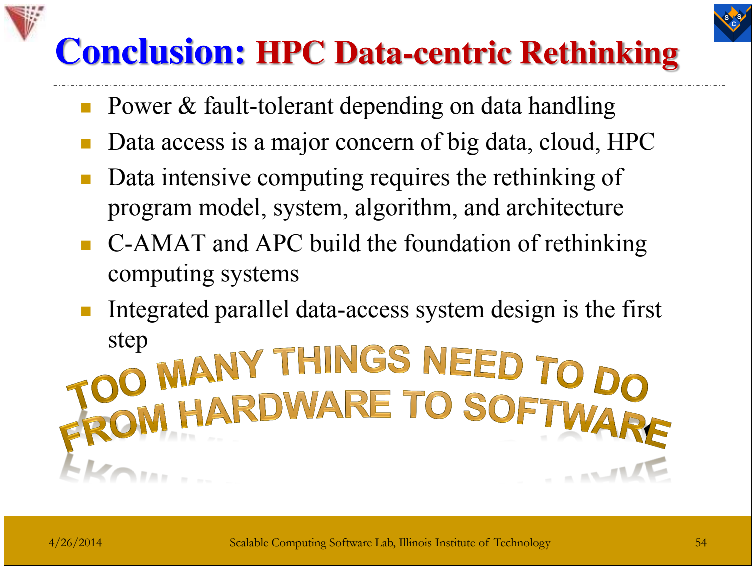


Where Does Energy Go? (2)

[courtesy Kogge]

- ~50% goes into accessing storage and moving data
- ~50% goes into “architecture convenience” (caching, virtual memory)
- ~2% goes into computation

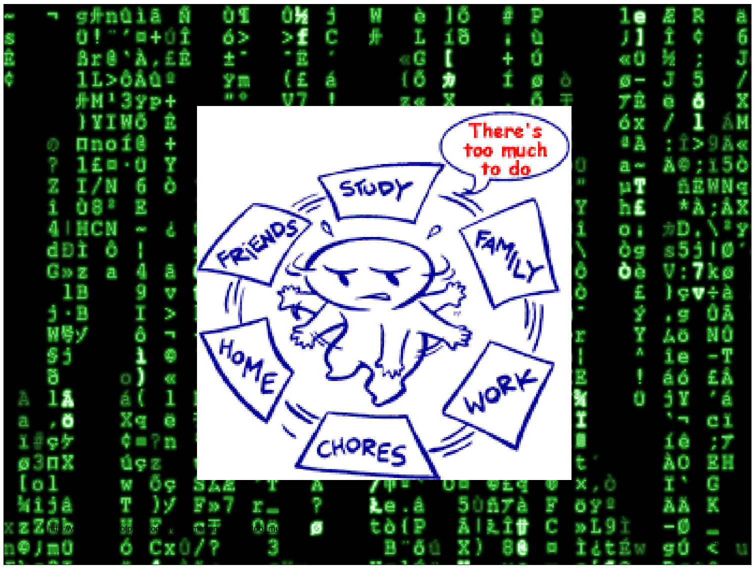
0	Total (pJ)	RF Access	Router Logic	Tag Access	32KB SRAM Access	DRAM Access	On-Chip Transport	TSV	Chip-Board-Chip	Chip-Optical-Chip
L1 Hit	39	30%	0%	28%	42%	0%	0%	0%	0%	0%
L2/L3 Hit	385	7%	0%	6%	34%	0%	53%	0%	0%	0%
Local	1380	1%	0%	2%	2%	68%	26%	1%	0%	0%
Global	13819	0%	24%	0%	0%	1%	13%	0%	10%	52%
Ave Read	706	2%	19%	2%	4%	8%	16%	0%	8%	41%
Write to L1	39	30%	0%	28%	42%	0%	0%	0%	0%	0%
Flush L1	891	0%	0%	1%	26%	32%	39%	1%	0%	0%

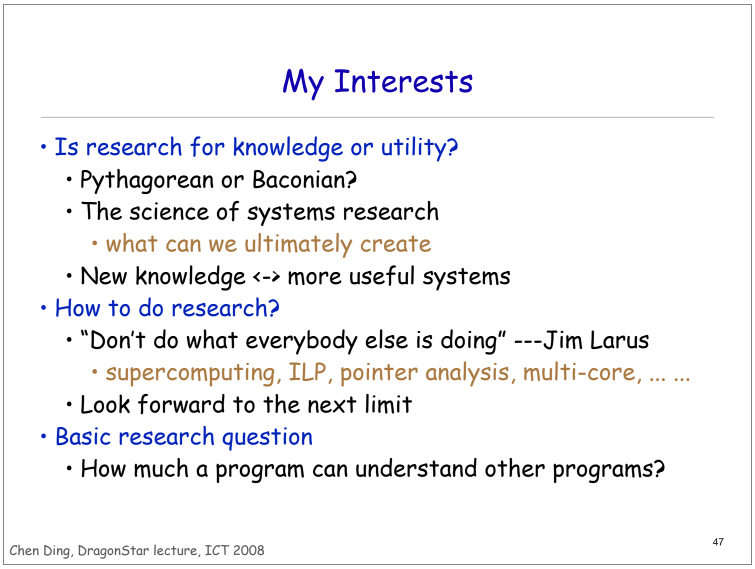


Conclusion: HPC Data-centric Rethinking

- Power & fault-tolerant depending on data handling
- Data access is a major concern of big data, cloud, HPC
- Data intensive computing requires the rethinking of program model, system, algorithm, and architecture
- C-AMAT and APC build the foundation of rethinking computing systems
- Integrated parallel data-access system design is the first step

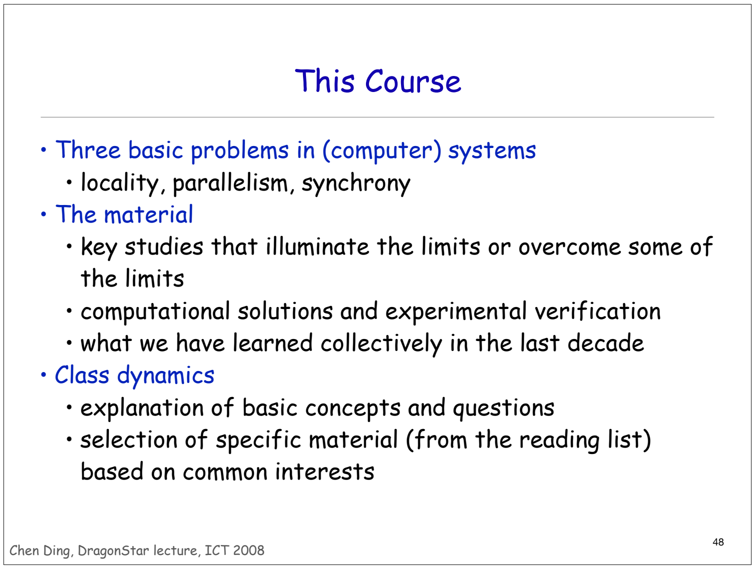
TOO MANY THINGS NEED TO DO FROM HARDWARE TO SOFTWARE





My Interests

- Is research for knowledge or utility?
  - Pythagorean or Baconian?
  - The science of systems research
    - what can we ultimately create
  - New knowledge <-> more useful systems
- How to do research?
  - “Don’t do what everybody else is doing” ---Jim Larus
    - supercomputing, ILP, pointer analysis, multi-core, ...
  - Look forward to the next limit
- Basic research question
  - How much a program can understand other programs?

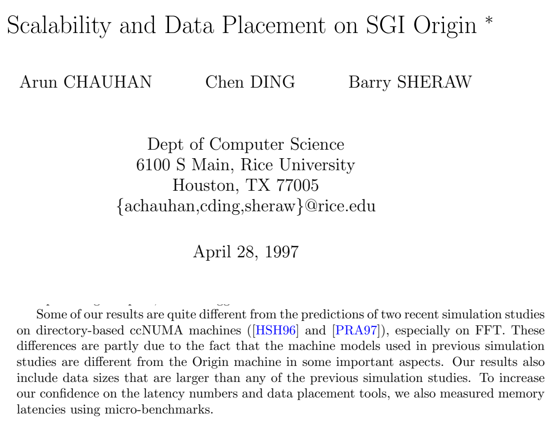


This Course

- Three basic problems in (computer) systems
  - locality, parallelism, synchrony
- The material
  - key studies that illuminate the limits or overcome some of the limits
  - computational solutions and experimental verification
  - what we have learned collectively in the last decade
- Class dynamics
  - explanation of basic concepts and questions
  - selection of specific material (from the reading list) based on common interests

1994: Instruction-level Parallelism

- Studying under Philip Sweany and Steven Carr at MTU
- Hiding the latency of operations and branches
  - most operations have predictable latency
  - except for ... ...
- Memory accesses
  - papers assumed L1 miss and L2 hit
- What about L2 misses?
  - the latency can be over a hundred cycles
  - ILP may not matter
  - no discussion in ILP papers
  - no one really knows the general answer until this decade



Scalability and Data Placement on SGI Origin *

Arun CHAUHAN Chen DING Barry SHERAW

Dept of Computer Science
6100 S Main, Rice University
Houston, TX 77005
{achauhan,cding,sheraw}@rice.edu

April 28, 1997

Some of our results are quite different from the predictions of two recent simulation studies on directory-based ccNUMA machines ([HSH96] and [PRA97]), especially on FFT. These differences are partly due to the fact that the machine models used in previous simulation studies are different from the Origin machine in some important aspects. Our results also include data sizes that are larger than any of the previous simulation studies. To increase our confidence on the latency numbers and data placement tools, we also measured memory latencies using micro-benchmarks.

2002: Mark Wegman

- Compiler legend, co-invented many classic techniques
  - compression, universal hashing, global value numbering, constant propagation, congruence, and static single assignement
- First ACM Workshop on Memory System Performance and Correctness (MSPC) in 2002
  - recurring comment during the PC meeting
    - "The first load takes a long time, the next 10 do not matter!"
  - Performance depends on not instruction count and not instruction type but when and how often there is a miss

The Memory Problem

- The journey of an idea

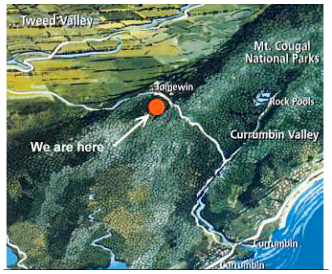
  1997 summer, I told Ken the problem of memory bandwidth
  1998(?) John Hennessy "single-node bandwidth" is fundamental problem
  1998 (?) John McCalpin "It is the bandwidth, stupid"
  1999 summer, Burton Smith at LCPC at UCSD
  2000, my dissertation done, talk w/ Crawford of Intel and Carter of Utah
  2001, my visits at Intel Itanium compiler group and Lawrence Livermore
  2002, Intel used RAMBUS in Pentium 4
  2002, Earth Simulator became world's fastest computer
  2002, Utah work won ICS best student paper award
  2003/4, US invested in high-end computing
  2003 PACT, global loop fusion by Intel
  2005 ICS, array regrouping by IBM
  2005 ICS, data packing used by Lawrence Livermore
  2003--2007, a new understanding of locality emerged
  2007, DARPA MIT multi-core workshop listed off-chip bw as #1 problem

Program Behavior Research

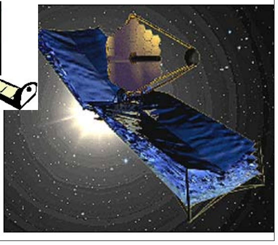
Introduction

1. The "memory problem"

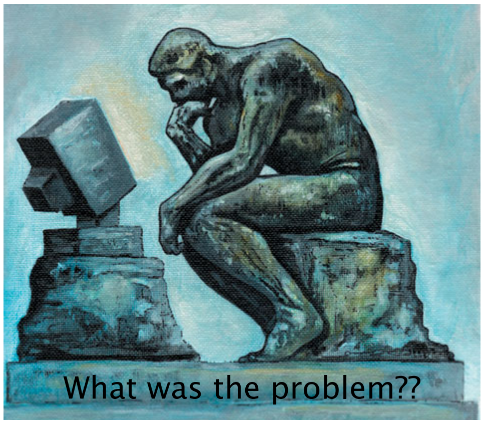
2. What is locality?



3. Measuring locality

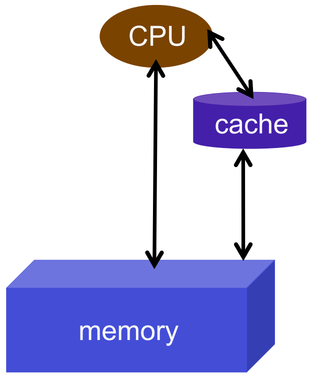


The Memory Problem



http://www.birminghamfreepress.com/commercial/illustrations/Thinker.jpg

Memory Performance



Problem

high memory latency

Improvement 1

fast cache

Improvement 2 & 3

Data prefetching

Multi-threading

Is there enough bandwidth?

Bandwidth Bottleneck

- Hardware trends
  - CPU speed improved 6400 times in 20 years
  - Memory bandwidth improved 139 times
- Software trends
  - large data sets
  - dynamic content and computation
  - modularized programming
- "Moore's gap"
  - data supply cannot keep up with CPU speed

Performance Model

- Balance
  - Callahan, Cocke, and Kennedy. JPDC 1988.
  - Ding and Kennedy. JPDC 2004.
- Machine balance
  - max words per cycle divided by max flops per cycle
- Program balance
  - # words accessed divided by # flops executed
  - total loads/stores divided by total floating-point ops
- Consequences
  - MB = PB → full utilization
  - MB > PB → memory idle
  - MB < PB → CPU idle

Program and Machine Balance
[Callahan, Cocke, and Kennedy, JPDC 1988]
[Ding and Kennedy, IPDPS 2000 and JPDC 2004]

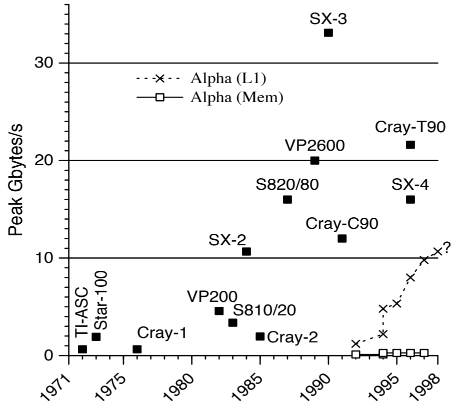
program/ machine	Program/machine balance		
	L1-Reg	L2-L1	Mem-L2
Convolution	6.4	5.1	5.2
Dmxpy	8.3	8.3	8.4
Mmjki (-O2)	24.0	8.2	5.9
Mmjki (-O3)	8.1	1.0	0.04
FFT	8.3	3.0	2.7
SP	10.8	6.4	4.9
Sweep3D	15.0	9.1	7.8
Origin2000	4.0	4.0	0.8

Memory-Bandwidth Bottleneck

- Ratios of demand to supply

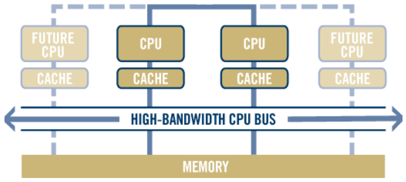
Applications	Ratio: demand/supply		
	Reg BW	Cache BW	Mem BW
Convolution	1.6	1.3	6.5
Dmxpy	2.1	2.1	10.5
Mmjki (-O2)	6.0	2.1	7.4
FFT	2.1	0.8	3.4
SP	2.7	1.6	6.1
Sweep3D	3.8	2.3	9.8

- Memory bandwidth is least sufficient
- Maximal CPU utilization: 10% to 33%
- **The imbalance is getting worse**
- **Software solution: Better caching**



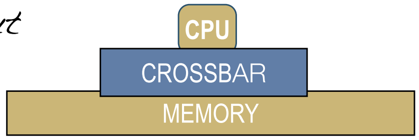
Steve Wallach's Fall 1999 Seminar

Instead of



http://qdn.qnx.com/images/articles/

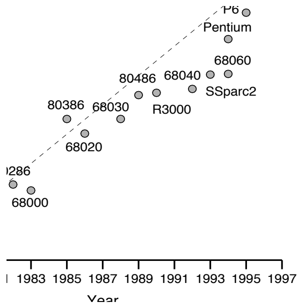
How about



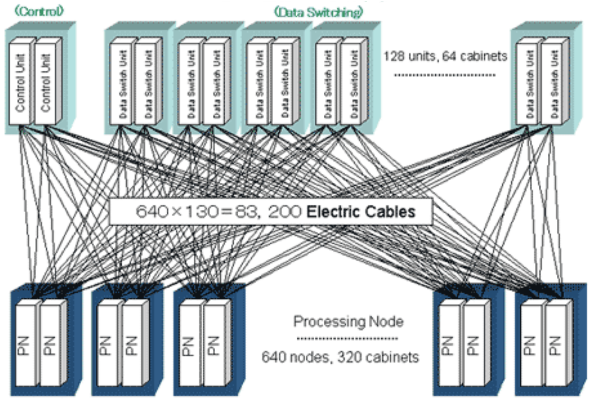
Memory Bandwidth Limitations of Future Microprocessors

Doug Burger, James R. Goodman, and Alain Kägi

Computer Sciences Department
University of Wisconsin-Madison

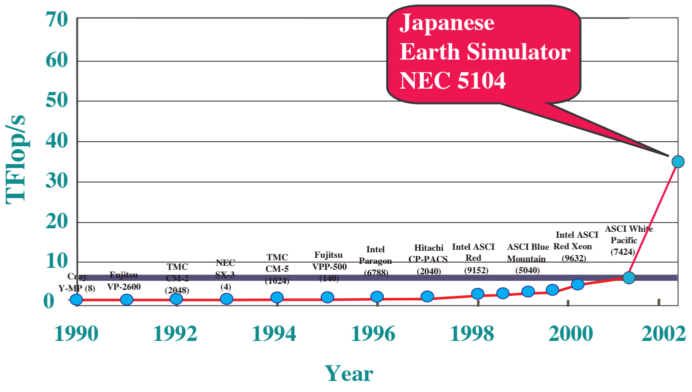


Earth Simulator 2002



http://www.es.jamstec.go.jp/esc/eng/Hardware/images/IN_02_b.gif

Fastest Computer Over Time



In 1980 a computation that took 1 full year to complete can today be done in ~ 5.4 seconds!

Jack Dongarra
University of Tennessee

IBM BlueGene/L 2005

Traditional Processors

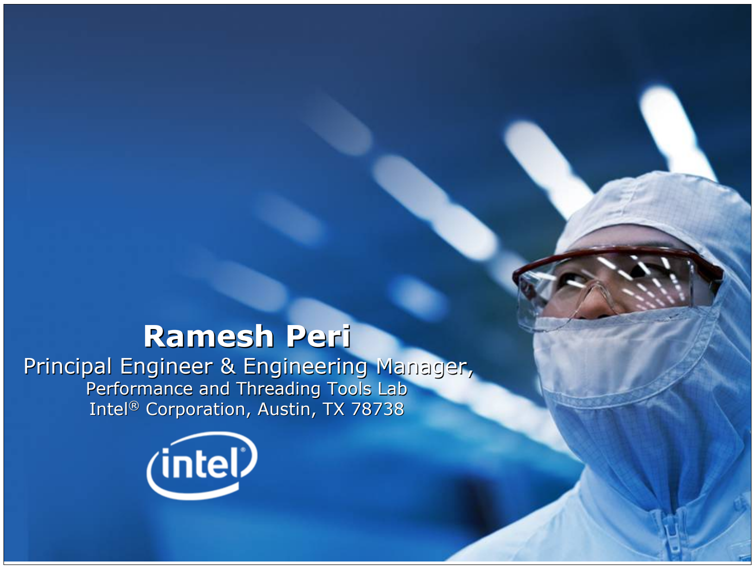
- NEC SX-4
  - vector processor, June 1997, 2 Gflops, 7.4GB/s bandwidth, 3.7 bytes per flop
- Alpha Server
  - May 1999, 600MHz CPU, 2MB cache, 932 Mflops, 50MB/s bandwidth, 0.48 byte per flop
- Pentium 4
  - Sept. 2003, 2.4GHz, 4.8 Gflops, 1.58GB/s bandwidth, 0.33 byte per flop
- Opteron
  - Dec. 2003, 2.2GHz, 4 Gflops, 1.1GB/s bandwidth, 0.28 byte per flop

Chip Multi-processors

- 4-way Power 4
  - May 2003, 1.7GHz, 6MB partitioned L2, 27.2 Gflops, 6GB/s, 0.22 byte per flop
- Intel Core2 Quad
  - April 2007, 2.4GHz, 154 Gflops, 5.3 GB/s bandwidth, 0.07 byte per flop
- 64-core Tilera
  - Dec. 2007, 750MHz, 25GB/s peak memory bandwidth, 3.8 TB on-chip bandwidth

Current Expert Opinion

http://www.westminstervillage.co.uk/images/crystal_ball2.jpg



Ramesh Peri

Principal Engineer & Engineering Manager,
Performance and Threading Tools Lab
Intel® Corporation, Austin, TX 78738

Multi-Core Processors – Are they Here Yet ?

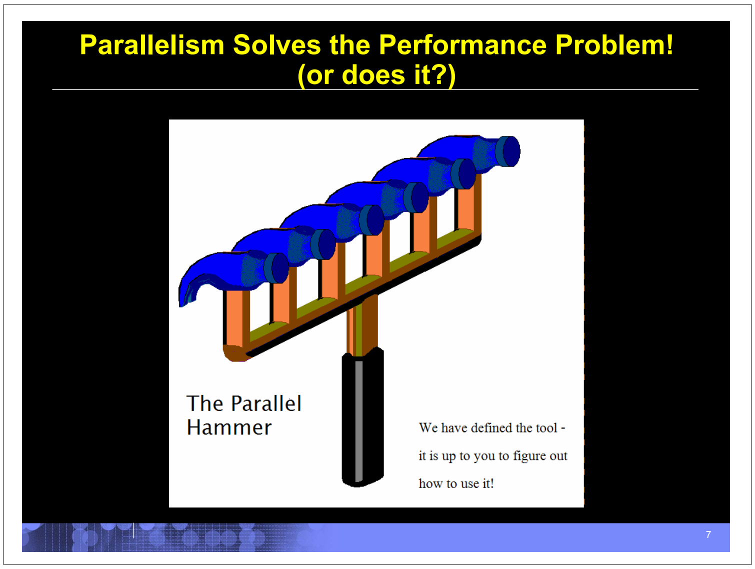
- My shopping basket at Fry's electronics on BlackFriday

Item	Cost
Motherboard+Intel® Quad core 2.4Ghz	200
4GB Memory	70
0.5TB Disk	80
Case	10
Graphics	10
CD/DVD	10
Total	**380**

Languages and Compilers for Multicore Computing Systems

FRAN ALLEN
allen@watson.ibm.com

Workshop Keynote
IIT Kanpur, India
December 13, 2007

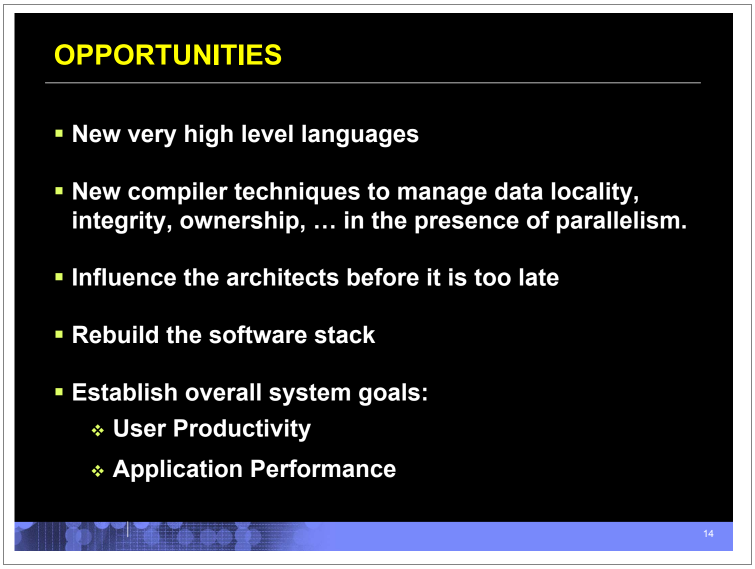


Parallelism Solves the Performance Problem! (or does it?)

The Parallel Hammer

We have defined the tool - it is up to you to figure out how to use it!

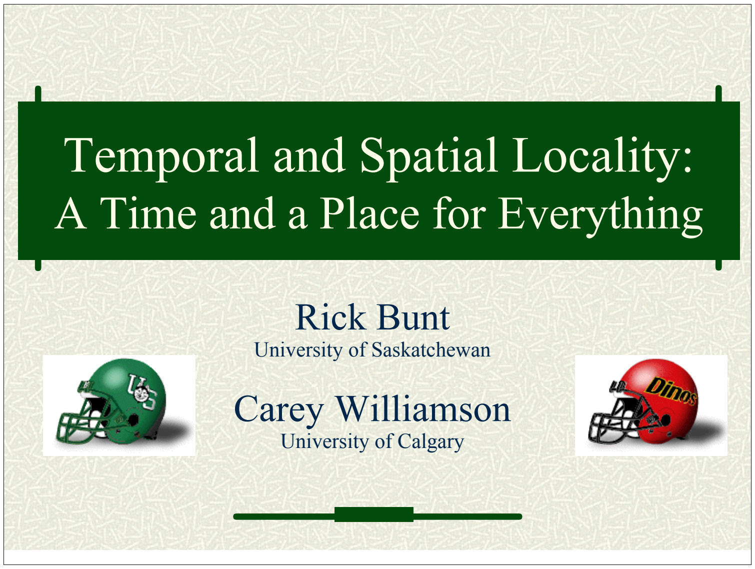
7



OPPORTUNITIES

- New very high level languages
- New compiler techniques to manage data locality, integrity, ownership, … in the presence of parallelism.
- Influence the architects before it is too late
- Rebuild the software stack
- Establish overall system goals:
  - User Productivity
  - Application Performance

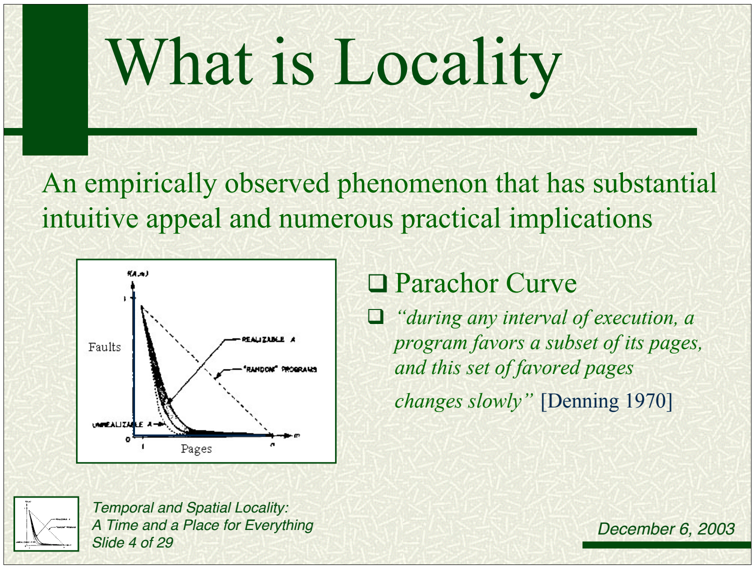
14



Temporal and Spatial Locality: A Time and a Place for Everything

Rick Bunt
University of Saskatchewan

Carey Williamson
University of Calgary



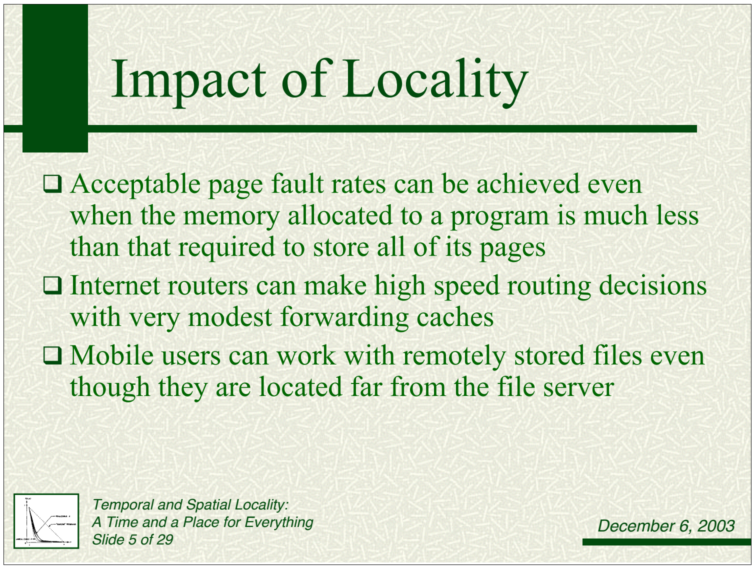
What is Locality

An empirically observed phenomenon that has substantial intuitive appeal and numerous practical implications

- ❑ Parachor Curve
- ❑ *"during any interval of execution, a program favors a subset of its pages, and this set of favored pages changes slowly"* [Denning 1970]

Temporal and Spatial Locality:
A Time and a Place for Everything
Slide 4 of 29

December 6, 2003

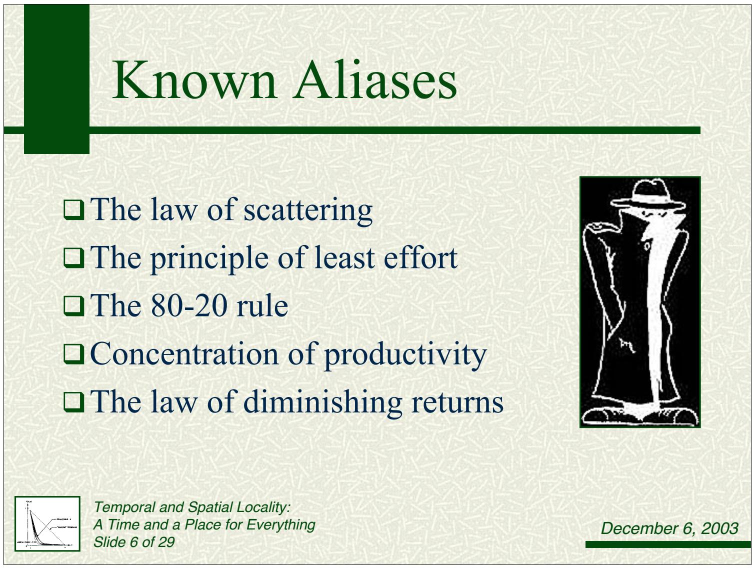


Impact of Locality

- ❑ Acceptable page fault rates can be achieved even when the memory allocated to a program is much less than that required to store all of its pages
- ❑ Internet routers can make high speed routing decisions with very modest forwarding caches
- ❑ Mobile users can work with remotely stored files even though they are located far from the file server

Temporal and Spatial Locality:
A Time and a Place for Everything
Slide 5 of 29

December 6, 2003



Known Aliases

- ❑ The law of scattering
- ❑ The principle of least effort
- ❑ The 80-20 rule
- ❑ Concentration of productivity
- ❑ The law of diminishing returns

Temporal and Spatial Locality:
A Time and a Place for Everything
Slide 6 of 29

December 6, 2003

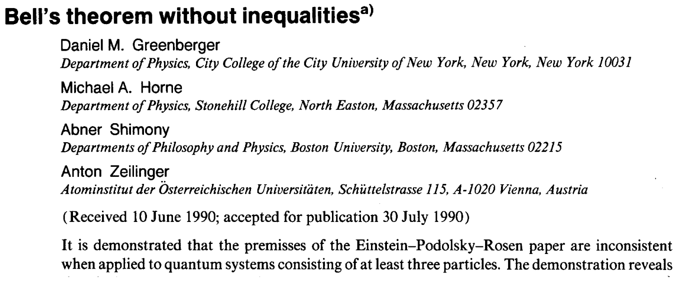
Locality Through the Ages

- ❑ Bradford's Law of Scattering [1934]
- ❑ Zipf's Principle of Least Effort [1949]
- ❑ Many applications before we *discovered* it
  - population distribution, distribution of wealth, distribution of biological species, article distribution in journals, and word usage in natural language.
  - has been used to plan the location of libraries and other facilities, to model the popularity of television programs, and to order search keys in hashing tables

The Underlying Concept

- ❑ There is a very large population of items, many more than we can manage
- ❑ There is a small core of relevant items on which we can productively focus our attention
- ❑ This core will continue to be relevant long enough to justify our attention

Locality: Innate or Emergent?

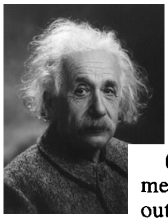


Bell's theorem without inequalities[a)]

Daniel M. Greenberger
Department of Physics, City College of the City University of New York, New York, New York 10031
Michael A. Horne
Department of Physics, Stonehill College, North Easton, Massachusetts 02357
Abner Shimony
Departments of Philosophy and Physics, Boston University, Boston, Massachusetts 02215
Anton Zeilinger
Atominstitut der Österreichischen Universitäten, Schüttelstrasse 115, A-1020 Vienna, Austria

(Received 10 June 1990; accepted for publication 30 July 1990)

It is demonstrated that the premisses of the Einstein–Podolsky–Rosen paper are inconsistent when applied to quantum systems consisting of at least three particles. The demonstration reveals



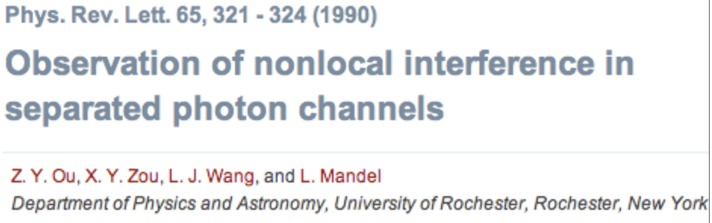
(i) *Perfect correlation*: If the spins of particles 1 and 2 are measured along the same direction, then with certainty the outcomes will be found to be opposite.

(ii) *Locality*: "Since at the time of measurement the two systems no longer interact, no real change can take place in the second system in consequence of anything that may be done to the first system."

(iii) *Reality*: "If, without in any way disturbing a system, we can predict with certainty (i.e., with probability equal to unity) the value of a physical quantity, then there exists an element of physical reality corresponding to this physical quantity."

(iv) *Completeness*:"Every element of the physical reality must have a counterpart in the [complete] physical theory."

Leonard Mandel



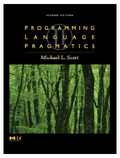
Phys. Rev. Lett. 65, 321 - 324 (1990)

Observation of nonlocal interference in separated photon channels

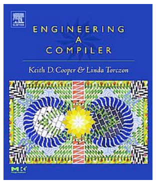
Z. Y. Ou, X. Y. Zou, L. J. Wang, and L. Mandel
Department of Physics and Astronomy, University of Rochester, Rochester, New York

How to Analyze Locality?

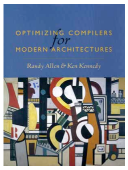
Programming and Program Analysis



- Language design & implementation [**Scott** *Programming Language Pragmatics*]
  - naming, types, control and data abstractions, imperative, functional, logical, parallel, ...



- Program analysis and optimization [**Cooper&Torczon** *Engineering a Compiler*]
  - invariance in (cyclic) graphs



- Dependence and parallelization [**Allen&Kennedy** *Optimizing Compilers for Modern Architectures*]
  - (re)ordering constraints
  - reorganization of loop and data spaces

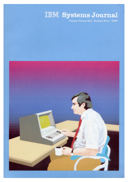
Program Analysis Methods

- Compilers
  - effective for scalars
  - for loop nests with linear index expressions
  - not for branches, recursion, indirect data access
- Profiling
  - accurate for one input
  - not for other inputs
- Run-time analysis
  - needed for input-dependent patterns
  - costly for detailed analysis and large-scale transformation

Programs and Program Behavior

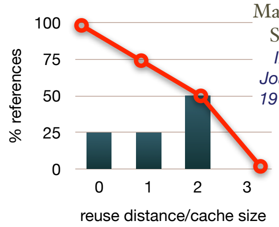
- Software trends
  - data intensive
    - dynamic and input dependent
  - parameterized code
    - templates, polymorphism
  - outside code
    - library, VM, VMM, OS, network, hardware
- Program behavior
  - a long sequence of operations
  - large-scale, compound effects
- Behavior-based analysis
  - identify composite patterns through off-line training or online monitoring

Reuse Distance



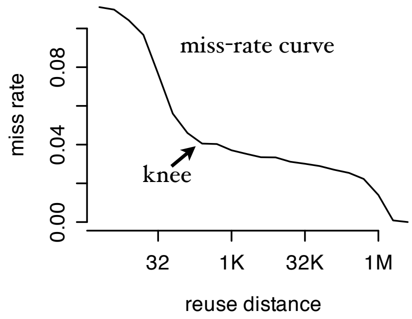
- Reuse distance of an access to data d
  - the volume of data between this and the previous access to d
- Reuse signature of an execution
  - the distribution of all finite reuse distances
  - gives the miss rate of fully associative cache of all sizes

a b c a a c $\overset{2}{b}$



Mattson, Gecsei, Slutz, Traiger
IBM Systems Journal, vol. 9(2), 1970, pp. 78-117

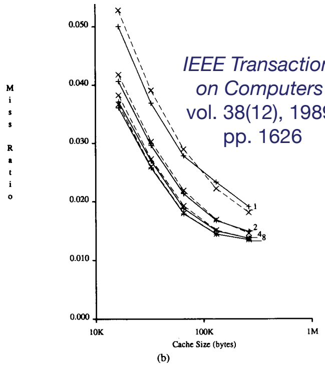
Working Set



Peter J. Denning

IEEE Transactions on Software Engineering, vol. 6(1), 1980, pp. 66

Cache Miss Rate

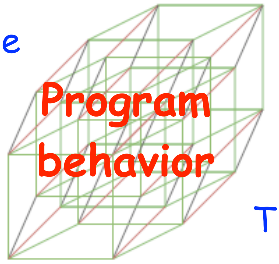


IEEE Transactions on Computers, vol. 38(12), 1989, pp. 1626

Fig. 11. Predicted (dashed) and actual (solid) miss ratios for trace "mul2" with caches of associativity 1, 2, 4, and 8. (a) Smaller caches. (b) Larger caches.

Allan J. Smith

- Predicted miss rate [Smith 1976, Hill&Smith 1989]
  - direct-mapped
  - set-associative
  - all sizes



Computation

Code

Program behavior

Data

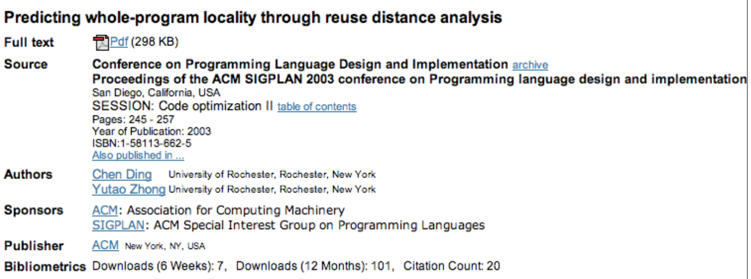
Input

Time

Environment

Measuring Reuse Distance

http://www.skynightly.com/images/telescopes-ancient-chinese-bg.jpg



Predicting whole-program locality through reuse distance analysis

Full text Pdf (298 KB)

Source **Conference on Programming Language Design and Implementation** archive
Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation
San Diego, California, USA
SESSION: Code optimization II table of contents
Pages: 245 - 257
Year of Publication: 2003
ISBN:1-58113-662-5
Also published in ...

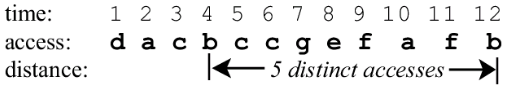
Authors Chen Ding University of Rochester, Rochester, New York
Yutao Zhong University of Rochester, Rochester, New York

Sponsors ACM: Association for Computing Machinery
SIGPLAN: ACM Special Interest Group on Programming Languages

Publisher ACM New York, NY, USA

Bibliometrics Downloads (6 Weeks): 7, Downloads (12 Months): 101, Citation Count: 20

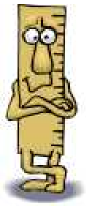
Measuring Reuse Distance



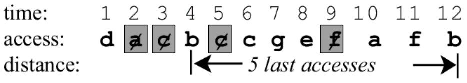
time: 1 2 3 4 5 6 7 8 9 10 11 12
access: d a c b c c g e f a f b
distance: ⟵ 5 distinct accesses ⟶

(a) an example access sequence
the reuse distance between two b's is 5

- Naive counting, O(N) time per access, O(N) space
  - N is the number of memory accesses
  - M is the number of distinct data elements
- Too costly
  - N is up to 120 billion, M 25 million



Precise Methods



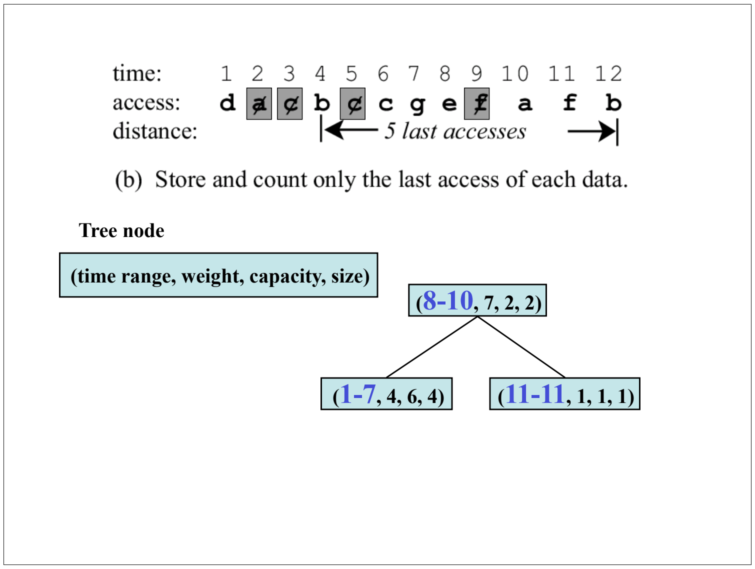
time: 1 2 3 4 5 6 7 8 9 10 11 12
access: d ~~a~~ ~~c~~ b ~~c~~ c g e ~~f~~ a f b
distance: ⟵ 5 last accesses ⟶

(b) Store and count only the last access of each data.

- stack algorithm [Mattson+ IBM 70]
  - O(M) time per access, O(M) space
- vector tree [Bennett&Kruskal IBM 75]
  - O(log N) time per access, O(N) space
- search tree [Olken LBL 81, Sugumar&Abraham UM 93]
  - O(log M) time per access, O(M) space
- space cost remains a major problem

Approximation

- Basic idea
  - measure only the first few digits of a long distance
  - use non-unit size tree nodes
    - tree size = M / average node size
  - bound the error by tree node size
- Guaranteed relative accuracy
  - a <= measured/actual_distance <= 1
    - e.g. a = 99%
  - logarithmic space cost
- Hashtable cost
  - space problem solved by Bennett-Kruskal in 1975
  - not considered in the discussion



time: 1 2 3 4 5 6 7 8 9 10 11 12
access: d ~~a~~ ~~c~~ b ~~c~~ c g e ~~f~~ a f b
distance: |← 5 last accesses →|

(b) Store and count only the last access of each data.

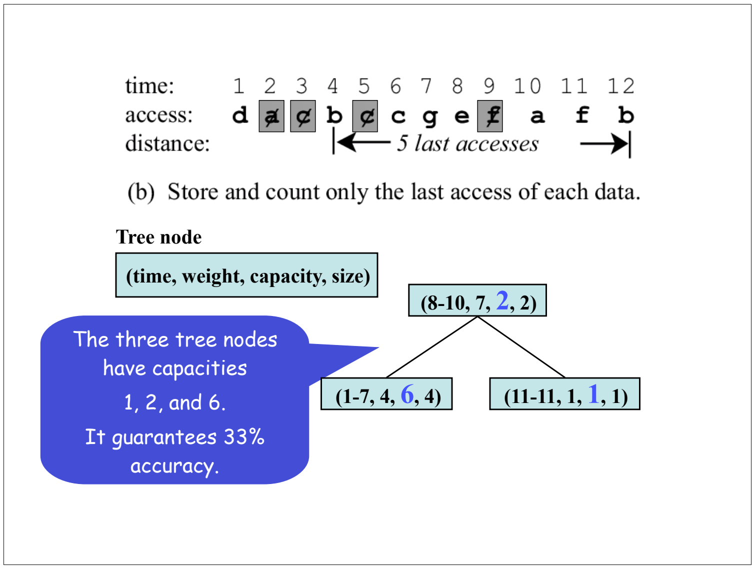
Tree node

(time range, weight, capacity, size)

(8-10, 7, 2, 2)

(1-7, 4, 6, 4)

(11-11, 1, 1, 1)



time: 1 2 3 4 5 6 7 8 9 10 11 12
access: d ~~a~~ ~~c~~ b ~~c~~ c g e ~~f~~ a f b
distance: |← 5 last accesses →|

(b) Store and count only the last access of each data.

Tree node

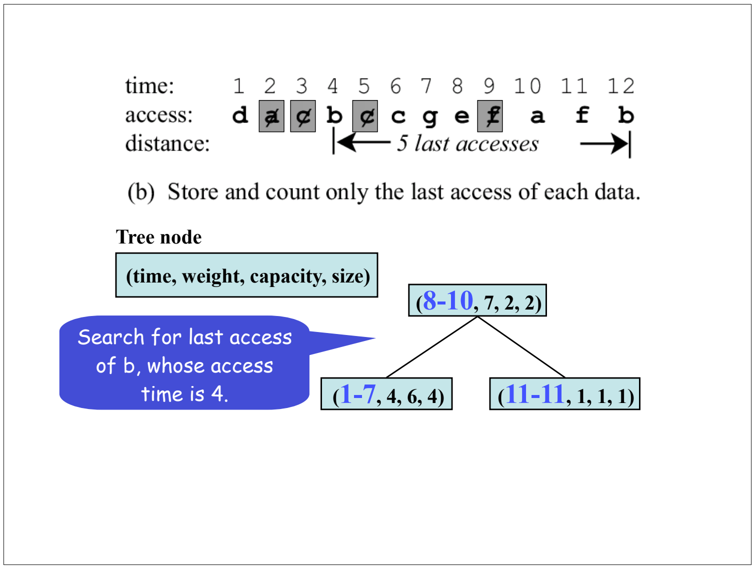
(time, weight, capacity, size)

(8-10, 7, 2, 2)

(1-7, 4, 6, 4)

(11-11, 1, 1, 1)

The three tree nodes have capacities 1, 2, and 6.
It guarantees 33% accuracy.



time: 1 2 3 4 5 6 7 8 9 10 11 12
access: d ~~a~~ ~~c~~ b ~~c~~ c g e ~~f~~ a f b
distance: |← 5 last accesses →|

(b) Store and count only the last access of each data.

Tree node

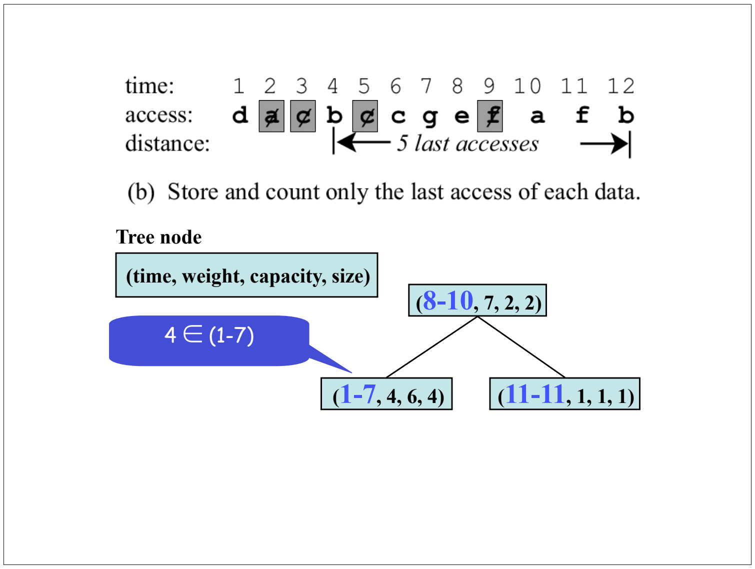
(time, weight, capacity, size)

(8-10, 7, 2, 2)

(1-7, 4, 6, 4)

(11-11, 1, 1, 1)

Search for last access of b, whose access time is 4.



time: 1 2 3 4 5 6 7 8 9 10 11 12
access: d ~~a~~ ~~c~~ b ~~c~~ c g e ~~f~~ a f b
distance: |← 5 last accesses →|

(b) Store and count only the last access of each data.

Tree node

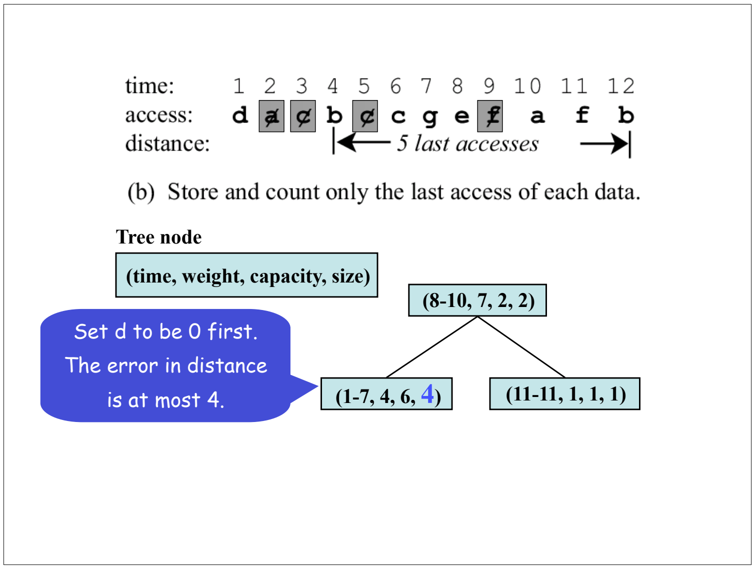
(time, weight, capacity, size)

(8-10, 7, 2, 2)

(1-7, 4, 6, 4)

(11-11, 1, 1, 1)

4 ∈ (1-7)



time: 1 2 3 4 5 6 7 8 9 10 11 12
access: d ~~a~~ ~~c~~ b ~~c~~ c g e ~~f~~ a f b
distance: |← 5 last accesses →|

(b) Store and count only the last access of each data.

Tree node

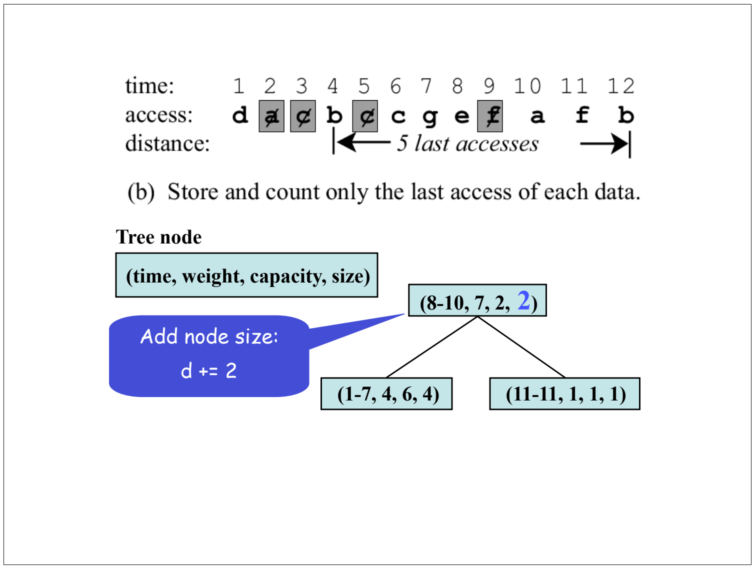
(time, weight, capacity, size)

(8-10, 7, 2, 2)

(1-7, 4, 6, 4)

(11-11, 1, 1, 1)

Set d to be 0 first.
The error in distance is at most 4.



time: 1 2 3 4 5 6 7 8 9 10 11 12
access: d ~~a~~ ~~c~~ b ~~c~~ c g e ~~f~~ a f b
distance: |← 5 last accesses →|

(b) Store and count only the last access of each data.

Tree node

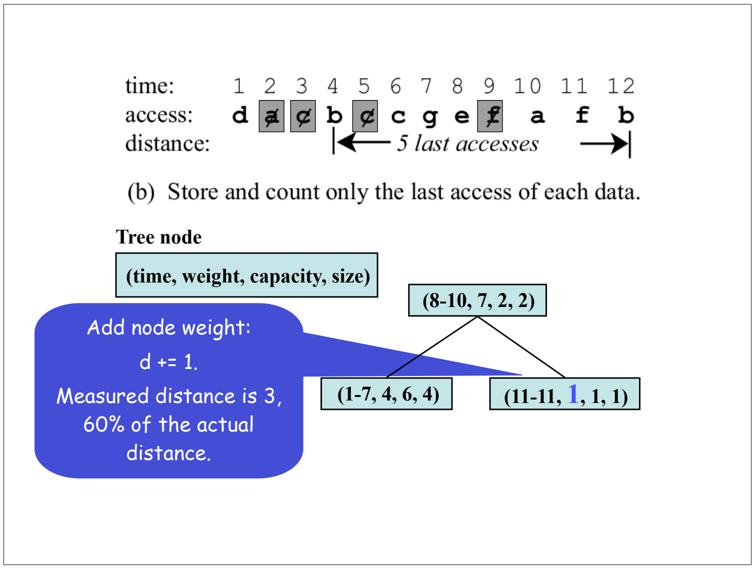
(time, weight, capacity, size)

(8-10, 7, 2, 2)

(1-7, 4, 6, 4)

(11-11, 1, 1, 1)

Add node size:
d += 2



time: 1 2 3 4 5 6 7 8 9 10 11 12
access: d ~~a~~ ~~c~~ b ~~c~~ c g e ~~f~~ a f b
distance: |← *5 last accesses* →|

(b) Store and count only the last access of each data.

Tree node

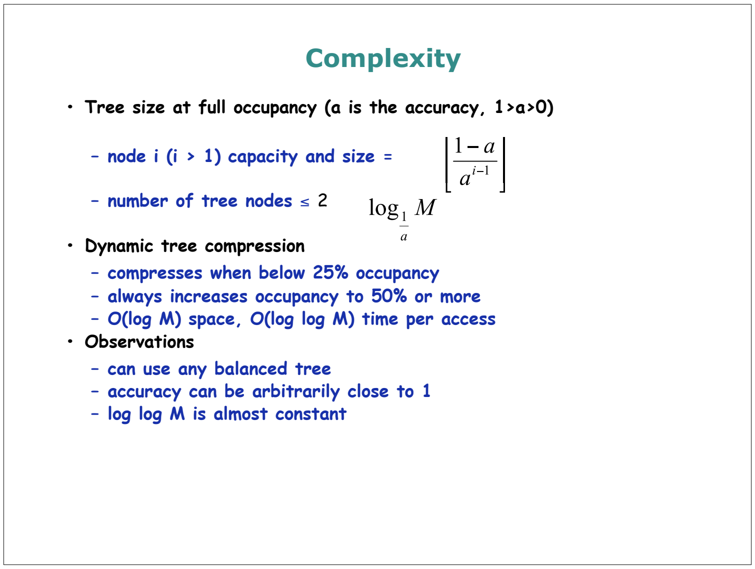
(time, weight, capacity, size)

(8-10, 7, 2, 2)

(1-7, 4, 6, 4)

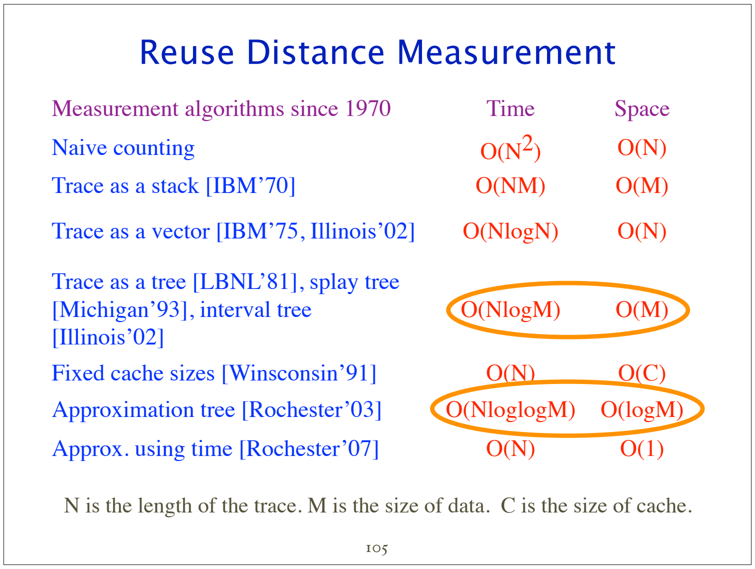
(11-11, 1, 1, 1)

Add node weight:
d += 1.
Measured distance is 3, 60% of the actual distance.



Complexity

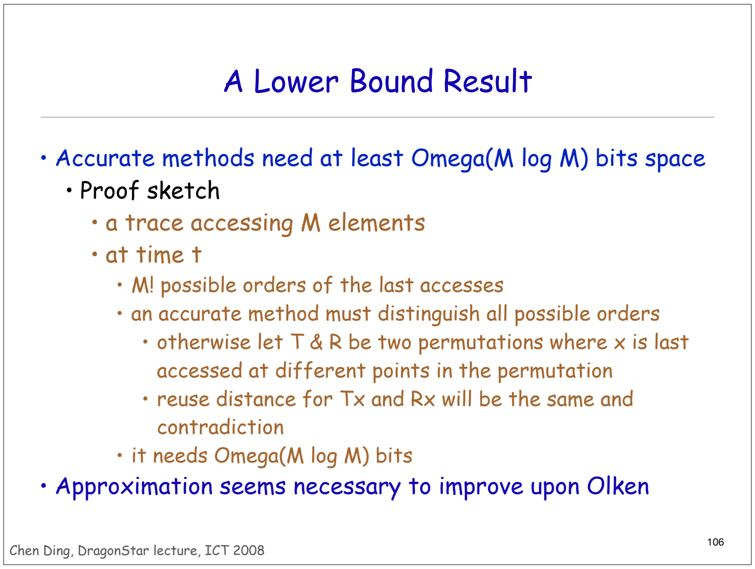
- Tree size at full occupancy (a is the accuracy, 1>a>0)
  - node i (i > 1) capacity and size = $\left\lfloor \frac{1-a}{a^{i-1}} \right\rfloor$
  - number of tree nodes ≤ $2 \log_{\frac{1}{a}} M$
- Dynamic tree compression
  - compresses when below 25% occupancy
  - always increases occupancy to 50% or more
  - O(log M) space, O(log log M) time per access
- Observations
  - can use any balanced tree
  - accuracy can be arbitrarily close to 1
  - log log M is almost constant



Reuse Distance Measurement

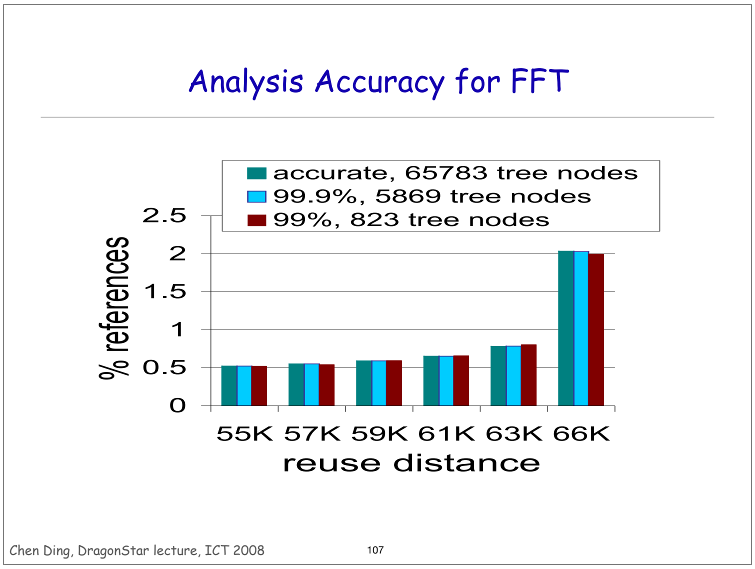
Measurement algorithms since 1970	Time	Space
Naive counting	$O(N^2)$	O(N)
Trace as a stack [IBM'70]	O(NM)	O(M)
Trace as a vector [IBM'75, Illinois'02]	O(NlogN)	O(N)
Trace as a tree [LBNL'81], splay tree [Michigan'93], interval tree [Illinois'02]	O(NlogM)	O(M)
Fixed cache sizes [Winsconsin'91]	O(N)	O(C)
Approximation tree [Rochester'03]	O(NloglogM)	O(logM)
Approx. using time [Rochester'07]	O(N)	O(1)

N is the length of the trace. M is the size of data. C is the size of cache.

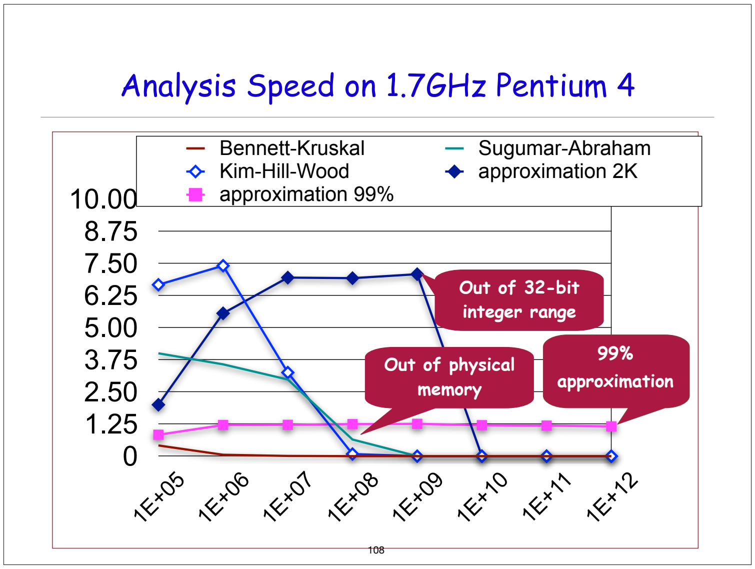


A Lower Bound Result

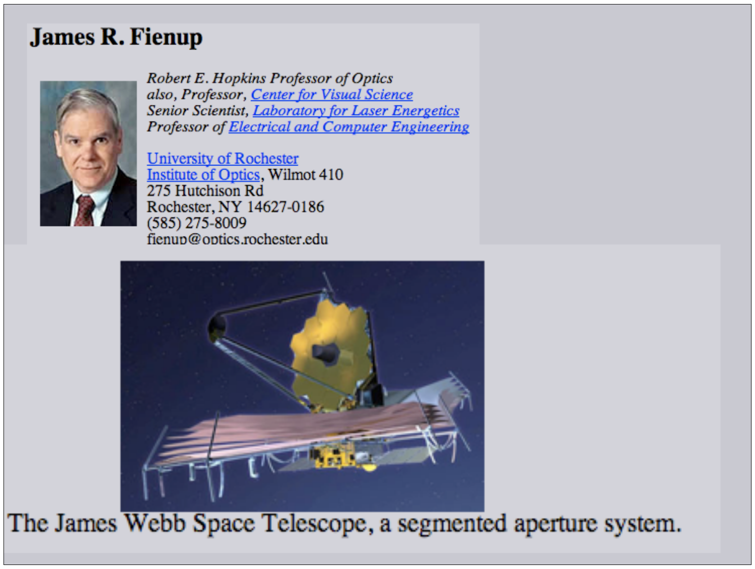
- Accurate methods need at least Omega(M log M) bits space
  - Proof sketch
    - a trace accessing M elements
    - at time t
      - M! possible orders of the last accesses
      - an accurate method must distinguish all possible orders
        - otherwise let T & R be two permutations where x is last accessed at different points in the permutation
        - reuse distance for Tx and Rx will be the same and contradiction
      - it needs Omega(M log M) bits
- Approximation seems necessary to improve upon Olken



Analysis Accuracy for FFT



Analysis Speed on 1.7GHz Pentium 4



James R. Fienup

Robert E. Hopkins Professor of Optics
also, Professor, Center for Visual Science
Senior Scientist, Laboratory for Laser Energetics
Professor of Electrical and Computer Engineering

University of Rochester
Institute of Optics, Wilmot 410
275 Hutchison Rd
Rochester, NY 14627-0186
(585) 275-8009
fienup@optics.rochester.edu

The James Webb Space Telescope, a segmented aperture system.

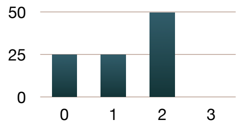
Full-scale model of the James Webb Space Telescope. Courtesy of ITT Industries Space Systems Division and the Rochester Museum and Science Center. Photo by Steven D. Adams.

Full-scale model of the James Webb Space Telescope. Courtesy of ITT Industries Space Systems Division and the Rochester Museum and Science Center. Photo by Steven D. Adams.

Whole-Program Locality

The Basic Tool Box

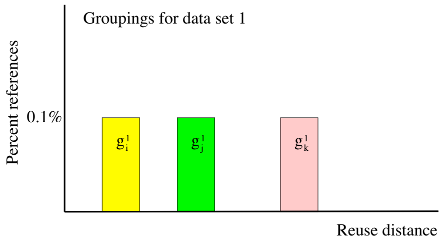
- Reuse distance
  - independent of coding styles, memory allocation, or hardware
  - possible to correlate between different runs
- Reuse signature is a spectrogram
  - behavior decomposition
  - pattern analysis
- Reuse distance trace is a signal
  - zooming in or out
  - period analysis

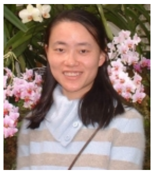


∞ ∞ ∞ 2 0 1 2
a b c a a c b

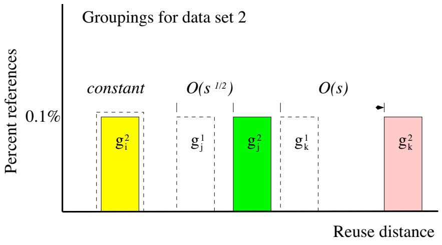
Pattern Recognition and Prediction

- Behavior decomposition
  - variable size: distance histogram
    - bins in distance histograms: logarithmic, log-linear
  - fixed size: reference histogram
    - divide the references into k, e.g. 1000, groups
- Pattern analysis
  - correlation among training inputs
    - constant, linear, sub-linear
  - single model or multi-model regression
  - input size defined computationally
    - memory footprint or distance-based sampling

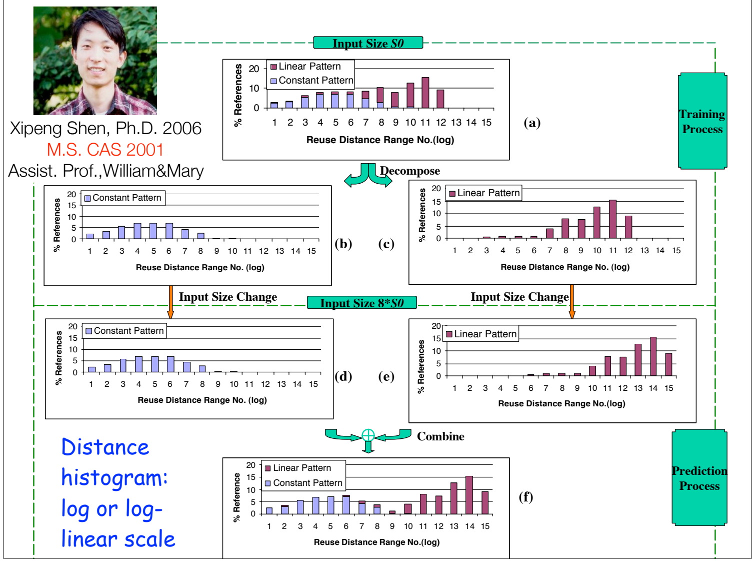




Yutao Zhong
Assist. Prof., George Mason U.
Fairfax, VA
Ph.D. Rochester 2005
B.S./M.S. Nanjing U. 2000



Reference histogram: each bin has 0.1% accesses

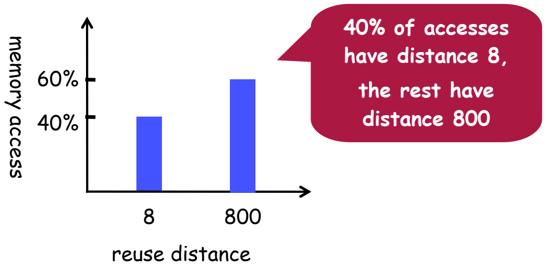


Xipeng Shen, Ph.D. 2006
M.S. CAS 2001
Assist. Prof.,William&Mary

Distance histogram: log or log-linear scale

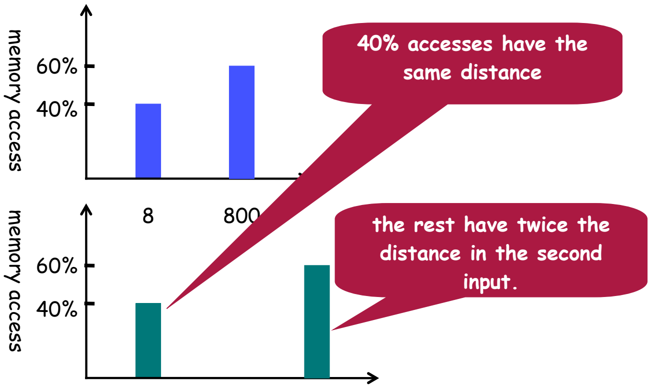
Divide and Unite

Whole-program pattern is an aggregate of all behavior groups. In multi-model prediction, each behavior group contains multiple pattern components.



40% of accesses have distance 8, the rest have distance 800

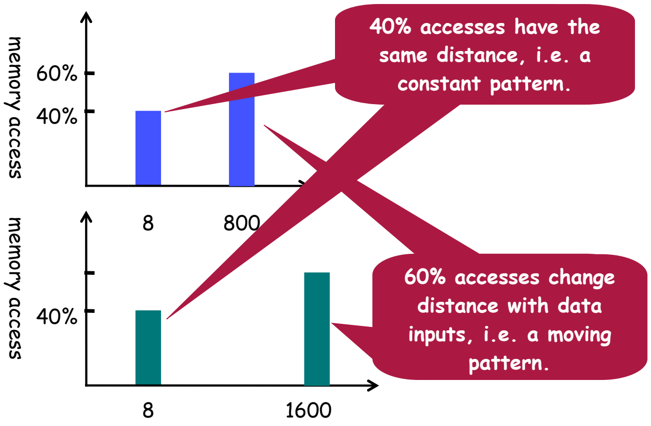
Correlation



40% accesses have the same distance

the rest have twice the distance in the second input.

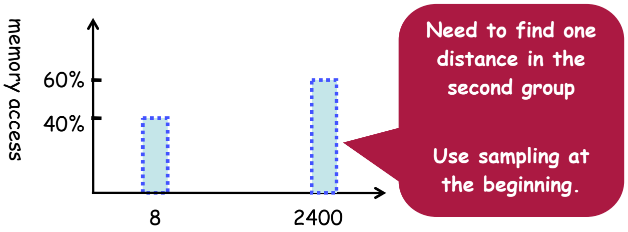
Pattern Recognition



40% accesses have the same distance, i.e. a constant pattern.

60% accesses change distance with data inputs, i.e. a moving pattern.

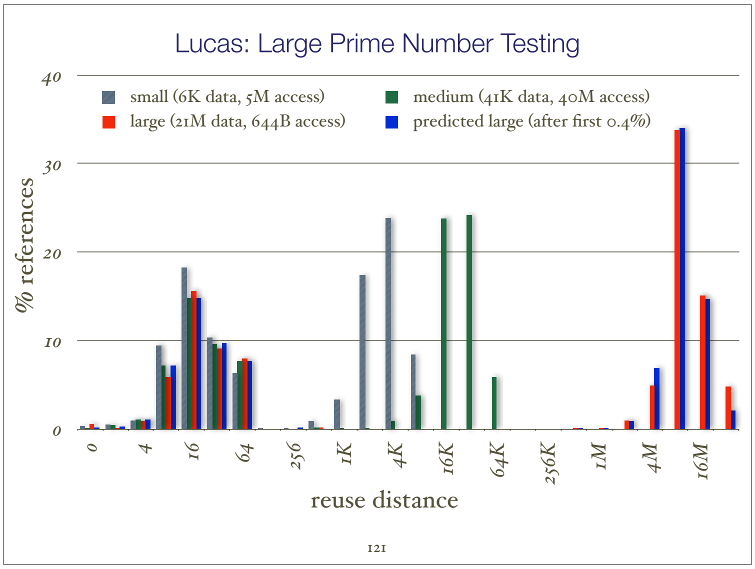
Pattern Prediction



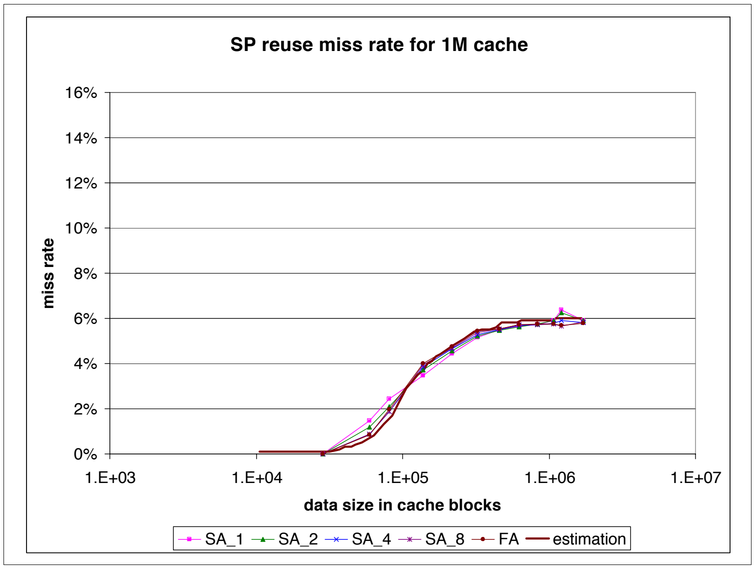
Need to find one distance in the second group

Use sampling at the beginning.

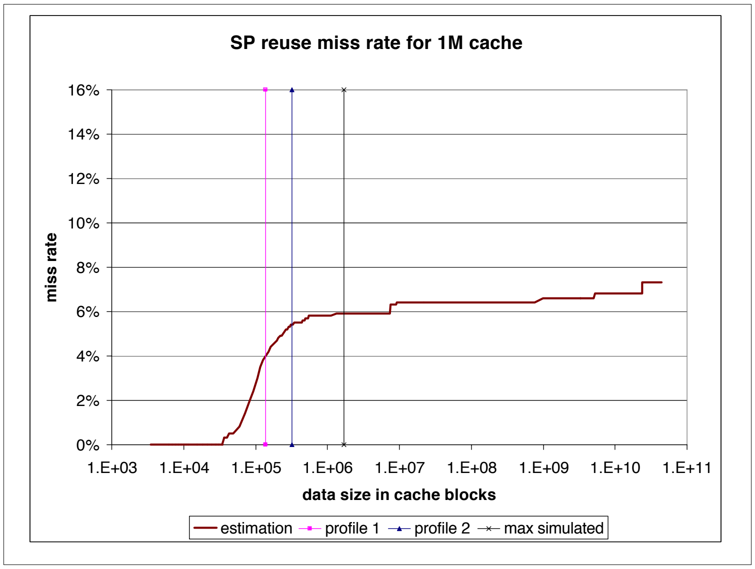
- **Observations**
  - code and data independent
  - does not predict execution time
  - not all programs have a consistent pattern



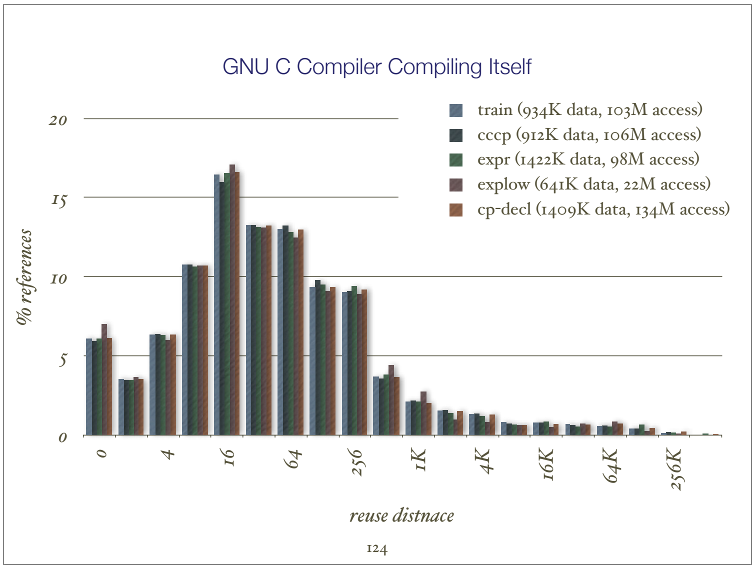
Lucas: Large Prime Number Testing



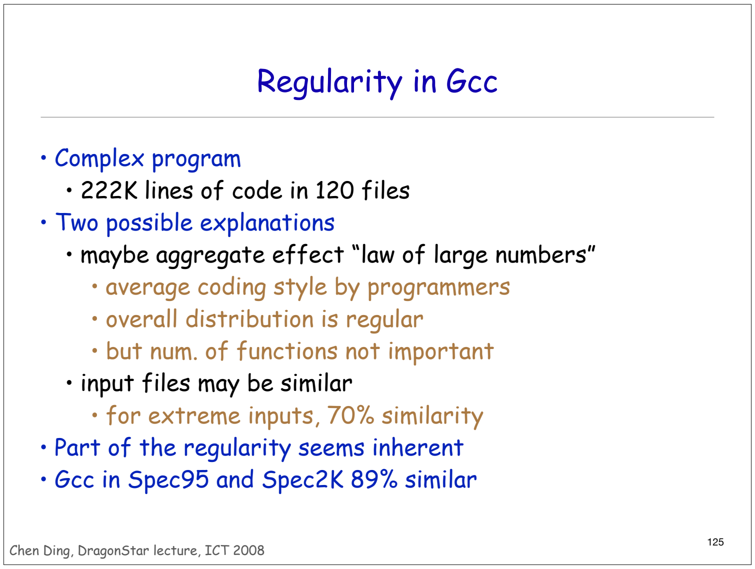
SP reuse miss rate for 1M cache



SP reuse miss rate for 1M cache

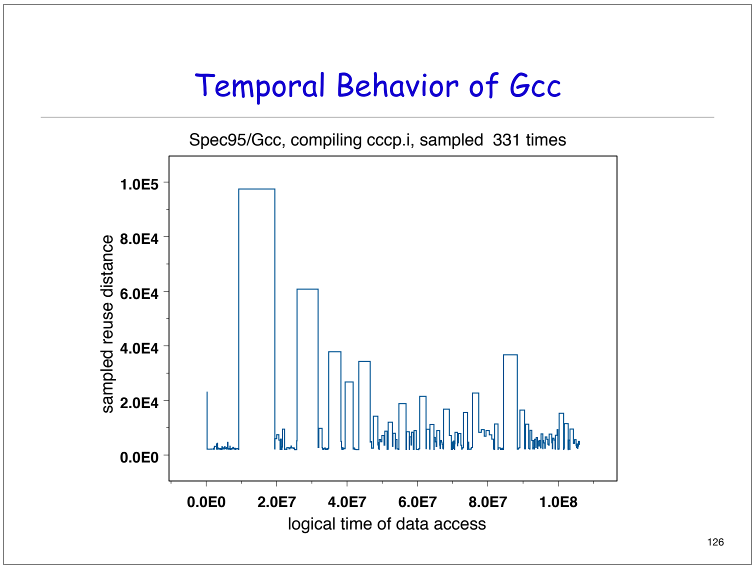


GNU C Compiler Compiling Itself



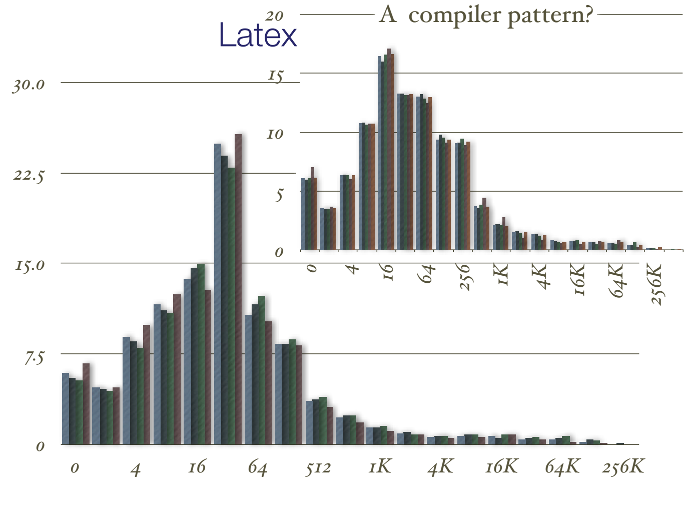
Regularity in Gcc

- Complex program
  - 222K lines of code in 120 files
- Two possible explanations
  - maybe aggregate effect "law of large numbers"
    - average coding style by programmers
    - overall distribution is regular
    - but num. of functions not important
  - input files may be similar
    - for extreme inputs, 70% similarity
- Part of the regularity seems inherent
- Gcc in Spec95 and Spec2K 89% similar

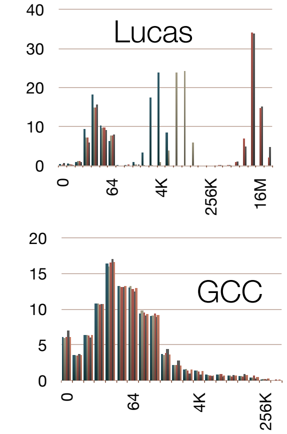


Temporal Behavior of Gcc

Spec95/Gcc, compiling cccp.i, sampled 331 times



Whole-Program Locality



- Reuse signature
  - differ by programs
  - consistent within the same program
- Emergent behavior
  - an observation
  - a computational discovery
  - implementation independent
- Limitations
  - not complete
  - no structure yet

Summary So Far

- Program behavior analysis
  - the composite effect of complex code
  - modeling and prediction based on past observations
  - very much like physical and biological sciences
- Strengths
  - behavior-based decomposition
  - discovery of major behavior components
  - cross-execution modeling and statistical analysis
- Later lectures
  - behavioral dimensions
  - relation with program analysis

Taming High Performance Computing with Compiler Technology

John Mellor-Crummey

Department of Computer Science
Center for High Performance Software Research

Instruction Based Memory Distance Analysis and its Application to Optimization

- Changpeng Fang
- Steve Carr
- Soner Önder
- Zhenlin Wang

Exercises

- Give the definition of machine and program balance
  - on modern systems, which balance is greater?
  - why is balance important for performance?
    - specifically, if machine M has a balance of 1 byte per flop and program P has a balance of 4 bytes per flop, what is the maximal performance of running program P on machine M?
- Give the definition of reuse distance
- Give the definition of temporal/spatial reuse
  - How does reuse distance explain them?
  - Which one is more precise and why?
- In what sense do we say reuse distance is machine independent?
- How did Denning define the primary working set?

Exercises (cont'd)

- What is the relation between reuse distance and program balance?
  - how to compute program balance?
- What is the relation between reuse distance and dependence?

Taming High Performance Computing with Compiler Technology

John Mellor-Crummey

Department of Computer Science
Center for High Performance Software Research

Why Performance Modeling?

- Insight into applications
  - —barriers to scalability
  - —insight into optimizations
- Mapping applications to systems
  - —Grid resource selection & scheduling
  - —intelligent run-time adaptation
- Workload-based design of future systems

Modeling Challenges

- Performance depends on:
  - —architecture specific factors
  - —application characteristics
  - —input data parameters
- Difficult to model execution time directly
- Collecting data at scale is expensive

Building Scalable Models

- Collect data from multiple runs
  - —*n*+1 runs to compute a model of degree *n*
- Approximation function:

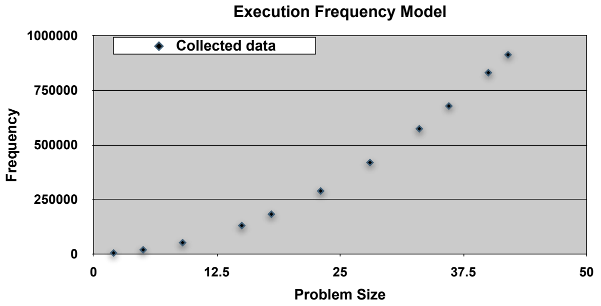
$$F(X) = c_n*B_n(X)+c_{n-1}*B_{n-1}(X)+\dots+c_0*B_0(X)$$

- A set of basis functions
- Include constraints
- Goal: determine coefficients

Use quadratic programming

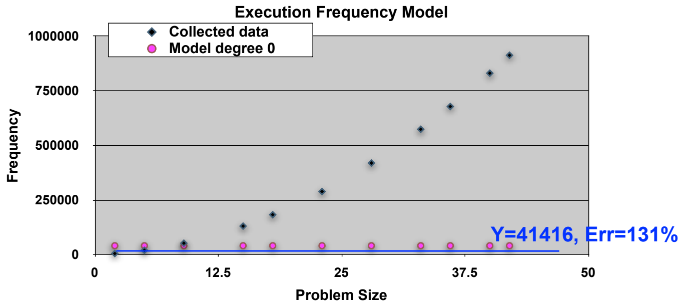
Execution Frequency Modeling Example

X	2	5	9	15	18	23	…
Count	5784	20244	53020	131104	183160	289200	



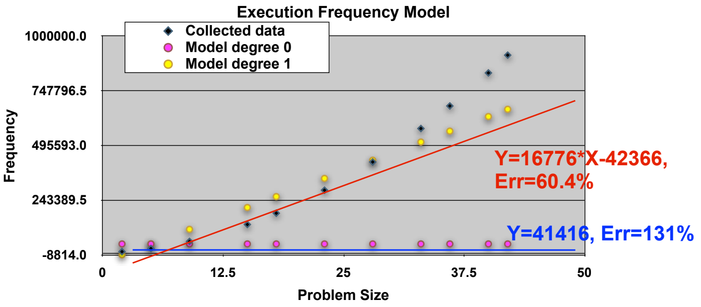
Execution Frequency Modeling Example

X	2	5	9	15	18	23	…
Count	5784	20244	53020	131104	183160	289200	



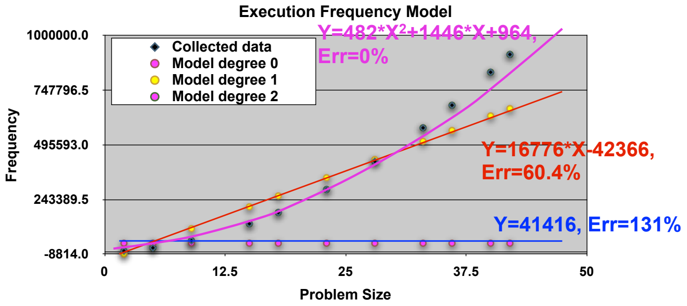
Execution Frequency Modeling Example

X	2	5	9	15	18	23	…
Count	5784	20244	53020	131104	183160	289200	



Execution Frequency Modeling Example

X	2	5	9	15	18	23	…
Count	5784	20244	53020	131104	183160	289200	



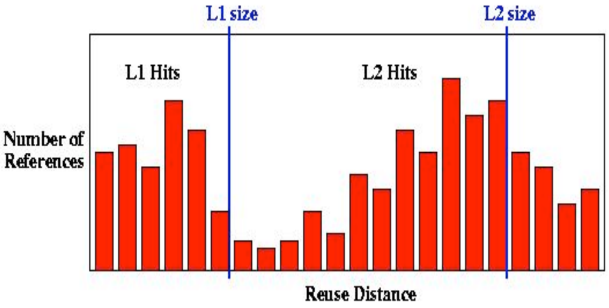
Memory Reuse Distance

- MRD: # unique data blocks referenced since target block last accessed

reference	I_1	I_2	I_3	I_2	I_3	I_2	I_3
memory block	A	B	A	C	A	B	B
MRD	∞	∞	1	∞	1	2	0

- I_1: 1 cold miss
- I_2: 2 cold misses, 1 @ distance 2
- I_3: 1 @ distance 0, 2 @ distance 1

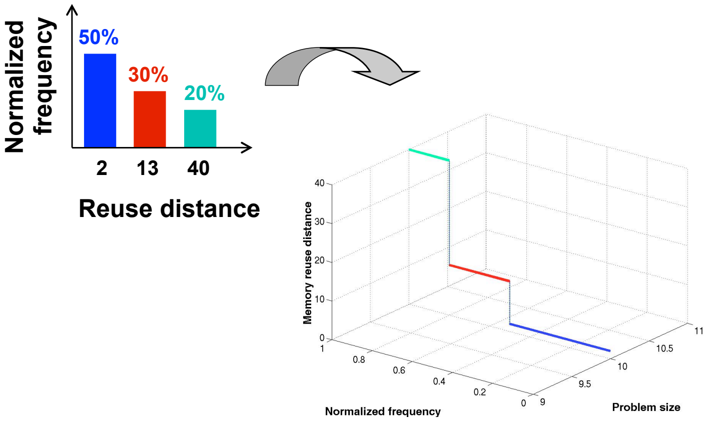
Memory reuse distance



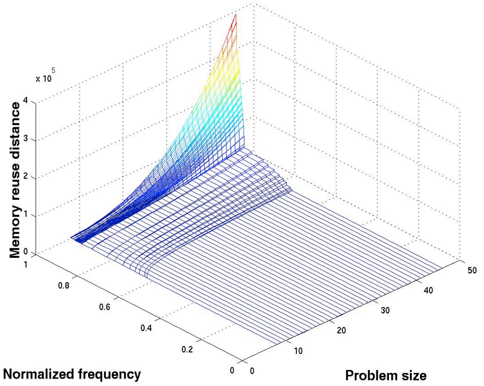
Modeling Memory Reuse Distance

- More complex than execution frequency
  - —cold misses
  - —histogram of reuse distances
    - – number of bins not constant
- Average reuse distance is misleading
  - —1 access with distance 10,000
  - —3 accesses with distance 0 } 2500 average
  - —cache has 1024 blocks

Modeling Memory Reuse Distance

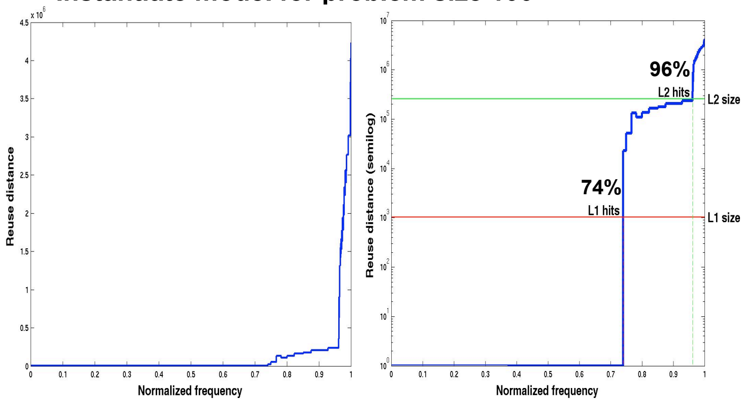


Modeling Memory Reuse Distance



Predict Number of Cache Misses

- Instantiate model for problem size 100



Fully Associative ⇒ Set Associative Model

From reuse distance histogram,
predict misses in a set associative cache

- Probability that access with reuse distance d misses in a cache with s sets and associativity k

$$P_{miss}(d,s,k) = 1 - \sum_{i=0}^{k-1} \left(\tfrac{1}{s}\right)^{i} \left(\tfrac{s-1}{s}\right)^{d-i} \binom{d}{i}$$

- Number of misses for a reuse distance histogram

$$Num_{misses}(Hist,s,k) = \sum_{bin_i \in Hist} (P_{miss}(D_{bin_i},s,k)F_{bin_i})$$

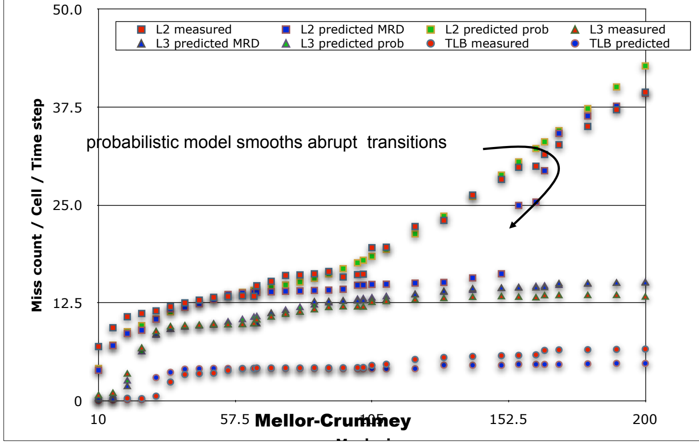
D_{bin_i} = average MRD for bin_i

F_{bin_i} = count for bin_i

Based on probabilistic model for set associativity by Hill & Smith (TOC 38:12, ‘89)

Memory Behavior: NAS BT 3.0

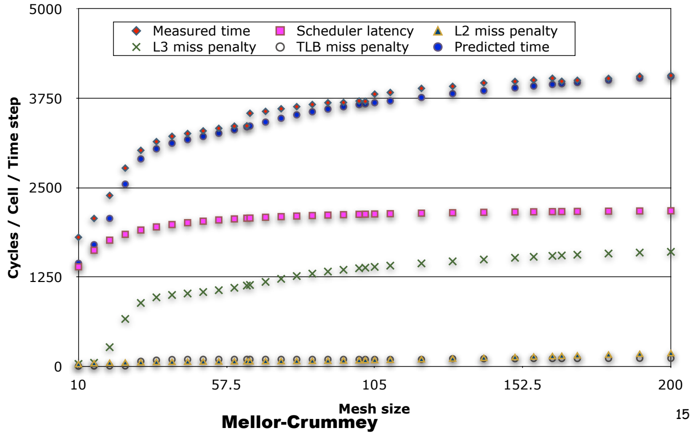
Itanium2 (256KB L2, 1.5MB L3)



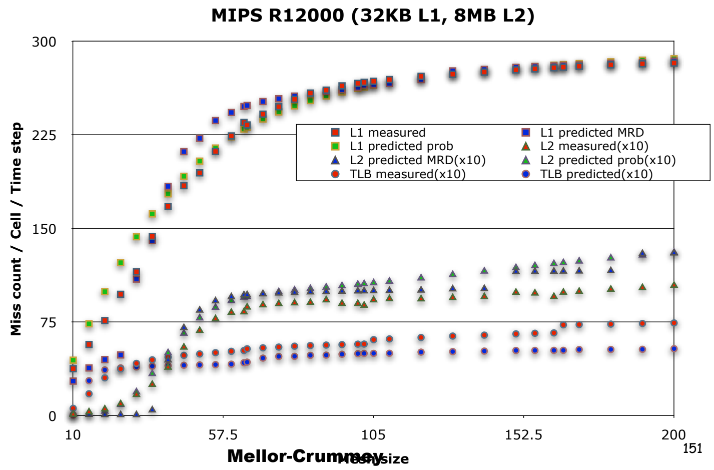
probabilistic model smooths abrupt transitions

Execution Behavior: NAS BT 3.0

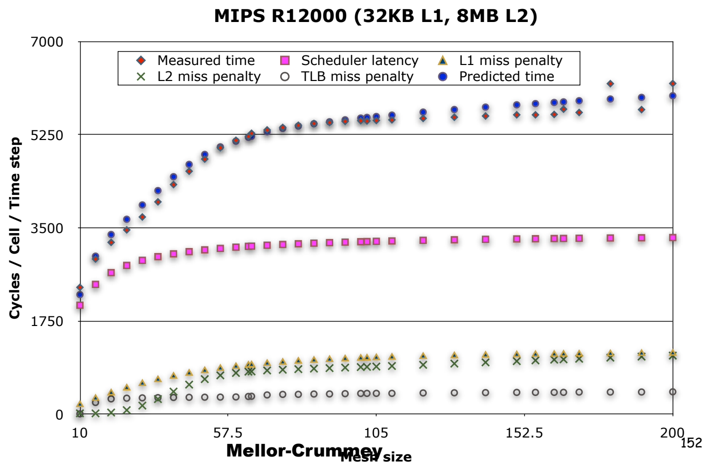
Itanium2 (256KB L2, 1.5MB L3)



Memory Behavior: NAS BT 3.0



Execution Behavior: NAS BT 3.0



Open Performance Modeling Issues

- Short term
  - —Better modeling of memory subsystem
    - – # outstanding loads to accurately predict memory latency
  - —Explore modeling of irregular applications
- Long term
  - —Model parallel applications
    - – Present modeling applies between synchronization points
    - – Combine with manually constructed parallel models
    - – Semi-automatically recover parallel trends
  - —Understand dynamic parallelism

Modeling Related Work

- Reuse distance
  - —Cache utilization [Beyls & D'Hollander]
  - —Investigating optimizations [Ding et al.]
- Program instrumentation
  - —EEL, QPT [Ball, Larus, Schnarr]
- Scalable analytic models
  - —[Vernon et al; Hoisie et al.]
- Cross-architecture models at scale
  - —[Snavely et al.; Cascaval et al.]
- Simulation (trace-based and execution-driven)

None yield semi-automatically derived scalable models

Instruction Based Memory Distance Analysis and its Application to Optimization

- Changpeng Fang
- Steve Carr
- Soner Önder
- Zhenlin Wang

Motivation

- Memory distance
  - A dynamic quantifiable distance in terms of memory reference between tow access to the same memory location.
  - reuse distance
  - access distance
  - value distance
- Is memory distance predictable across both integer and floating-point codes?
  - predict miss rates
  - predict critical instructions
  - identify instructions for load speculation

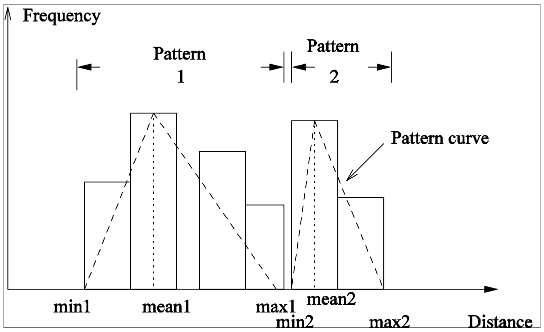
Instruction Based Memory Distance Analysis

- How can we represent the memory distance of an instruction?
  - For each active interval, we record 4 words of data
    - min, max, mean, frequency
  - Some locality patterns cross interval boundaries
    - merge adjacent intervals, i and i + 1, if

$$\min_{i+1} - \max_i \le \max_i - \min_i$$

    - merging process stops when a minimum frequency is found
    - needed to make reuse distance predictable
  - The set of merged intervals make up memory distance patterns

Merging Example



Experimental Methodology

- Use 11 CFP2000 and 11 CINT2000 benchmarks
  - others don't compile correctly
- Use ATOM to collect reuse distance statistics
- Use test and train data sets for training runs
- Evaluation based on dynamic weighting
- Report reuse distance prediction accuracy
  - value and access very similar

Reuse Distance Prediction

Suite	Patterns		Coverage %	Accuracy %
	%constant	%linear		
CFP2000	85.1	7.7	93.0	97.6
CINT2000	81.2	5.1	91.6	93.8

Coverage issues

- Reasons for no coverage
  1. instruction does not appear in at least one test run
  2. reuse distance of test is larger than train
  3. number of patterns does not remain constant in both training runs

Suite	Reason 1	Reason 2	Reason 3
CFP2000	4.2%	0.3%	2.5%
CINT2000	2.2%	4.4%	1.8%

Number of Patterns

Suite	1	2	3	4	≥5
CFP2000	81.8%	10.5%	4.8%	1.4%	1.5%
CINT2000	72.3%	10.9%	7.6%	4.6%	5.3%

Miss Rate Prediction Methodology

- Three miss-rate prediction schemes
  - TCS – test cache simulation
    - Use the actual miss rates from running the program on a the test data for the reference data miss rates
  - RRD – reference reuse distance
    - Use the actual reuse distance of the reference data set to predict the miss rate for the reference data set
    - An upper bound on using reuse distance
  - PRD –predicted reuse distance
    - Use the predicted reuse distance for the reference data set to predict the miss rate.

Cache Configurations

config no.	L1	L2	
1	32K, fully assoc.	1M	fully assoc.
2	32K, 2-way	1M	8-way
3			4-way
4			2-way

L1 Miss Rate Prediction Accuracy

Suite	PRD	RRD	TCS
CFP2000	97.5	98.4	95.1
CINT2000	94.4	96.7	93.9

L2 Miss Rate Accuracy

Suite	2-way			Fully Associative		
	PRD	RRD	TCS	PRD	RRD	TCS
CFP2000	91%	93%	87%	97%	99.9%	91%
CINT2000	91%	95%	87%	94%	99.9%	89%

Critical Instructions

- Given reuse distance for an instruction
  - Can we determine which instructions are critical in terms of cache performance?
- An instruction is critical if it is in the set of instructions that generate the most L2 cache misses
  - Those top miss-rate instructions whose cumulative total misses account for 95% of the misses in a program.
- Use the execution frequency of one training run to determine the relative contribution number of misses for each instruction
- Compare the actual critical instructions with predicted
  - Use cache configuration 2

Critical Instruction Prediction

Suite	PRD	RRD	TCS	%pred	%act
CPF2000	92%	98%	51%	1.66%	1.67%
CINT2000	89%	98%	53%	0.94%	0.97%

Critical Instruction Patterns

Suite	1	2	3	4	≥5
CFP2000	22.1	38.4	20.0	12.8	6.7
CINT2000	18.7	14.5	25.5	22.5	18

Miss Rate Discussion

- PRD performs better than TCS when data size is a factor
- TCS performs better when data size doesn't change much and there are conflict misses
- PRD is much better at identifying the critical instructions than TCS
  - these instructions should be targets of optimization

Value-based Prediction

- Memory dependence only if addresses and values match

  store a_1, v_1
  store a_2, v_2
  store a_3, v_3
  load a_4, v_4

  Can move ahead if $a_1=a_2=a_3=a_4$, $v_2=v_3$ and $v_1 \neq v_2$
- The access distance of a load to the first store in a sequence of stores storing the same value is called the value distance

Summary

- Over 90% of memory operations can have reuse distance predicted with a 97% and 93% accuracy, for floating-point and integer programs, respectively
- We can accurately predict miss rates for floating-point and integer codes
- We can identify 92% of the instructions that cause 95% of the L2 misses
- Access- and value-distance-based memory disambiguation are competitive with best hardware techniques without a hardware table

Other Distance-Based Studies

- Register and cache performance modeling
  - Li et al. from Purdue, Interact 1996
  - Huang and Shen, CMU, Micro 1996
  - Beyls and D'Hollander from Ghent, PDCS 2001
  - Almasi et al. from Illinois, MSP 2002
  - Zhong et al. from Rochester, LCR 2002
- File caching
  - Zhou et al., USENIX 2001
  - Jiang and Zhang, SIGMetrics 2002