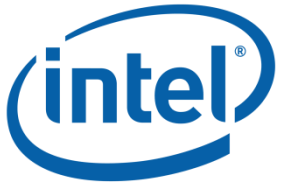


Multi-core Computing
Lecture 2

MADALGO Summer School 2012
Algorithms for Modern Parallel and Distributed Models

Phillip B. Gibbons
Intel Labs Pittsburgh

August 21, 2012

Multi-core Computing Lectures: Progress-to-date on Key Open Questions

- **How to formally model multi-core hierarchies?**
- **What is the Algorithm Designer's model?**
- **What runtime task scheduler should be used?**
- **What are the new algorithmic techniques?**
- **How do the algorithms perform in practice?**

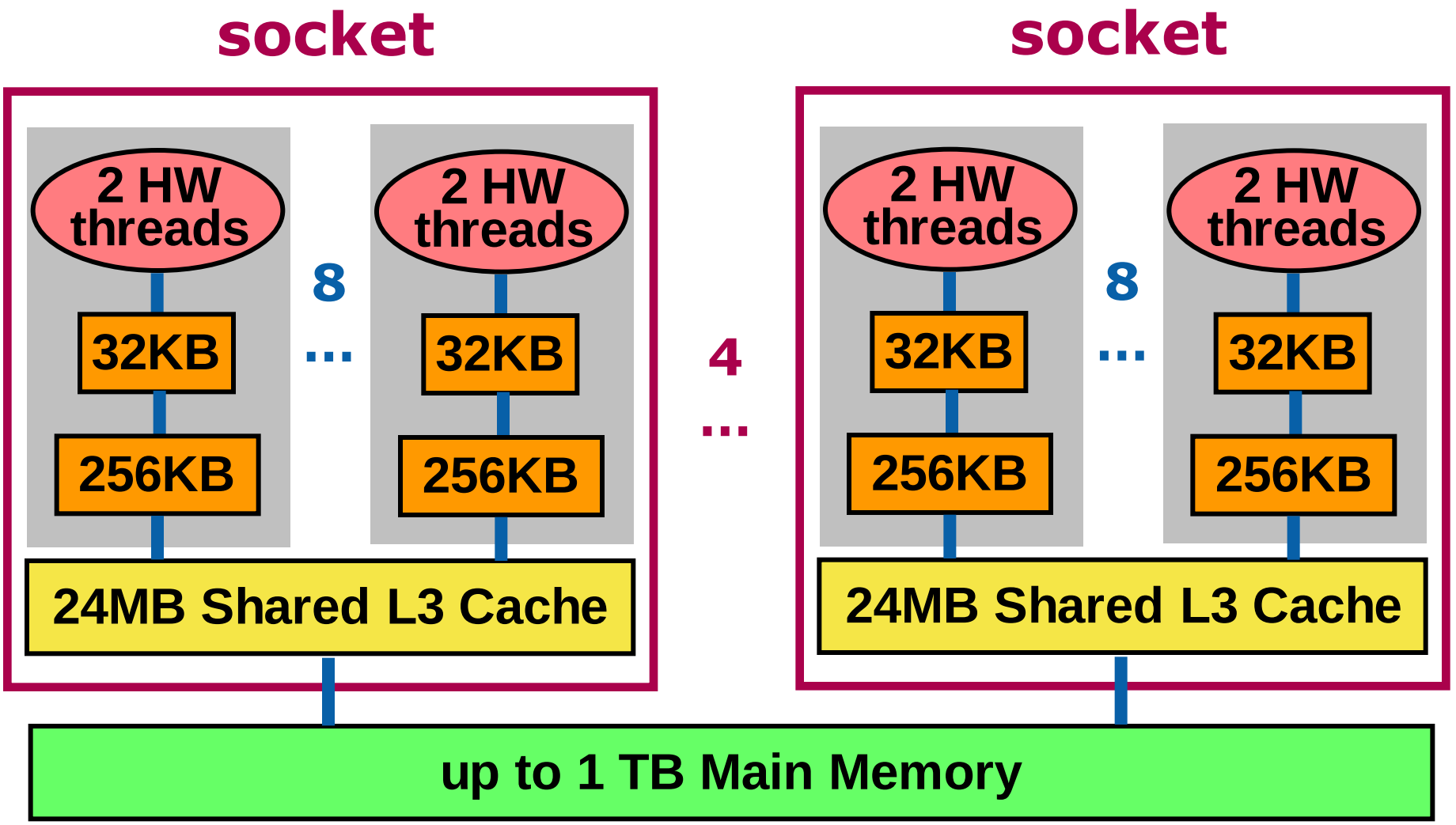
Lecture 1 Summary

- **Multi-cores: today, future trends, challenges**
- **Computations & Schedulers**
  - Modeling computations in work-depth framework
  - Schedulers: Work Stealing & PDF
- **Cache miss analysis on 2-level parallel hierarchy**
  - Private caches OR Shared cache
- **Low-depth, cache-oblivious parallel algorithms**
  - Sorting & Graph algorithms

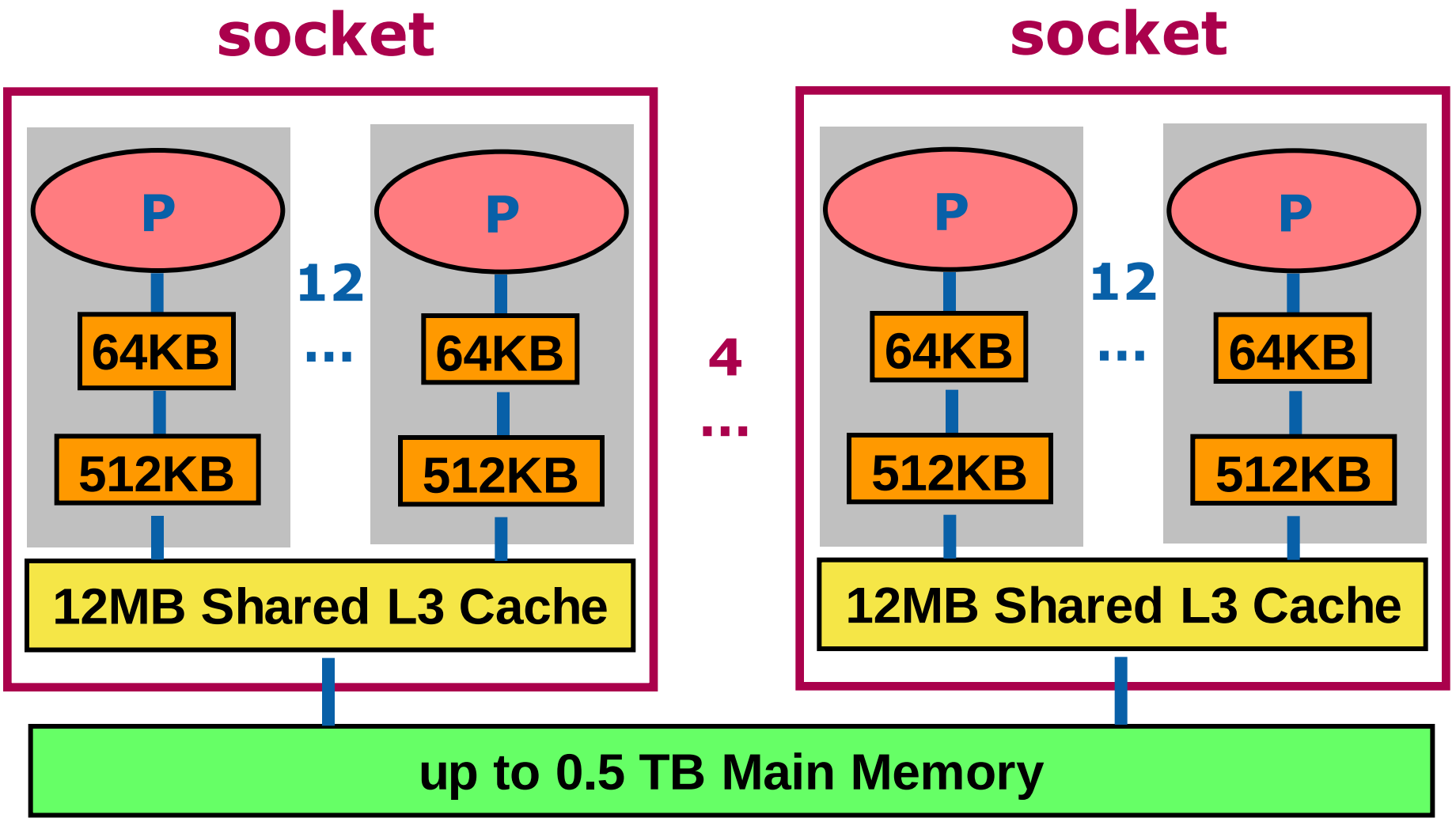
Lecture 2 Outline

- **Modeling the Multicore Hierarchy**
  - PMH model
- **Algorithm Designer's model exposing Hierarchy**
  - Multi-BSP model
- **Quest for a Simplified Hierarchy Abstraction**
- **Algorithm Designer's model abstracting Hierarchy**
  - Parallel Cache-Oblivious (PCO) model
- **Space-Bounded Schedulers**
  - Revisit PCO model

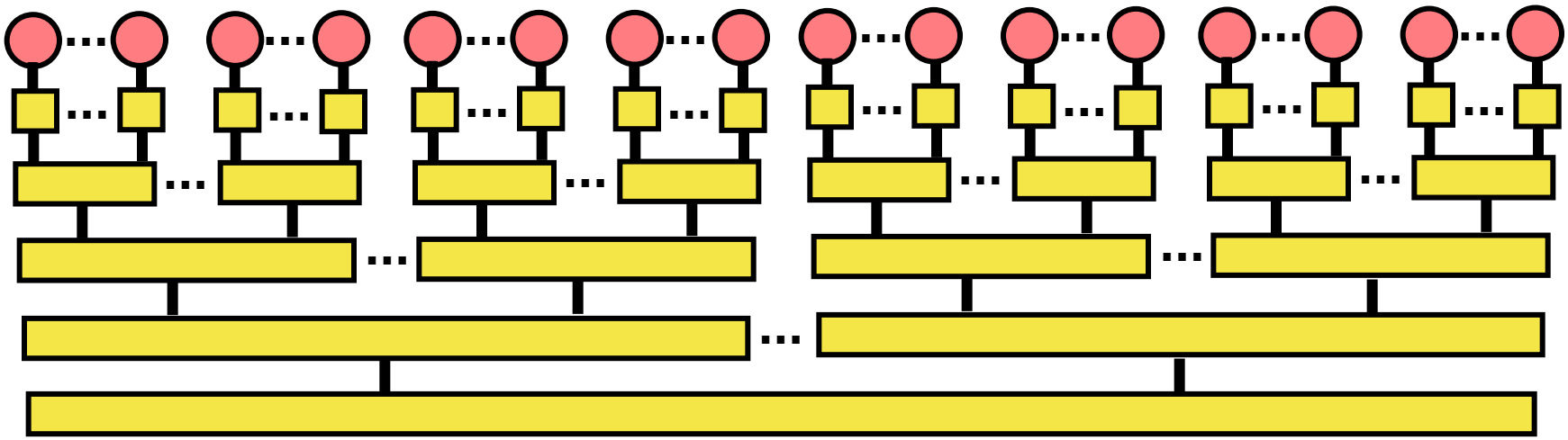
32-core Xeon 7500 Multi-core



48-core AMD Opteron 6100



How to Model the Hierarchy (?)

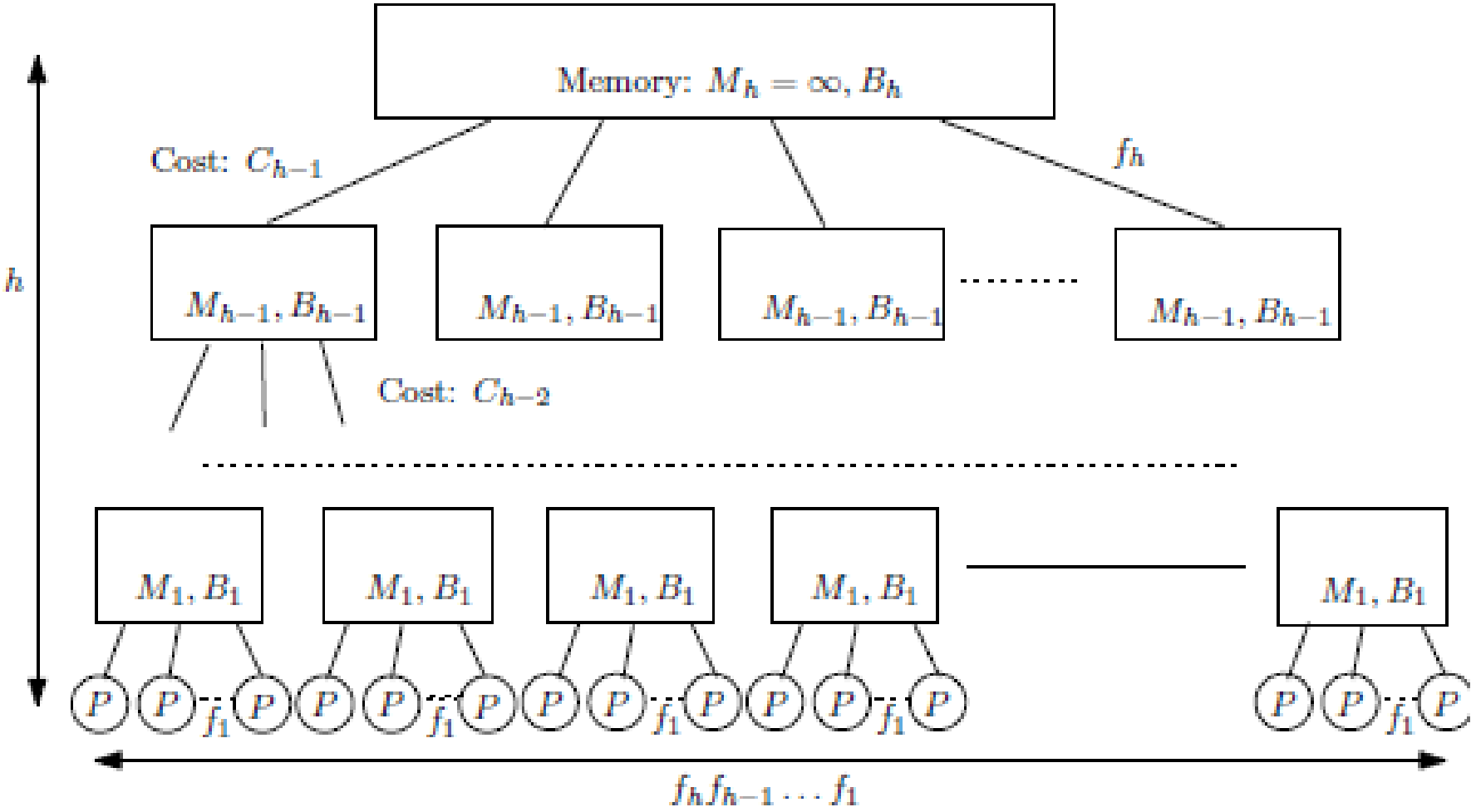


"Tree of Caches" abstraction captures existing multi-core hierarchies

Parallel Memory Hierarchy (PMH) model

[Alpern, Carter, Ferrante '93]

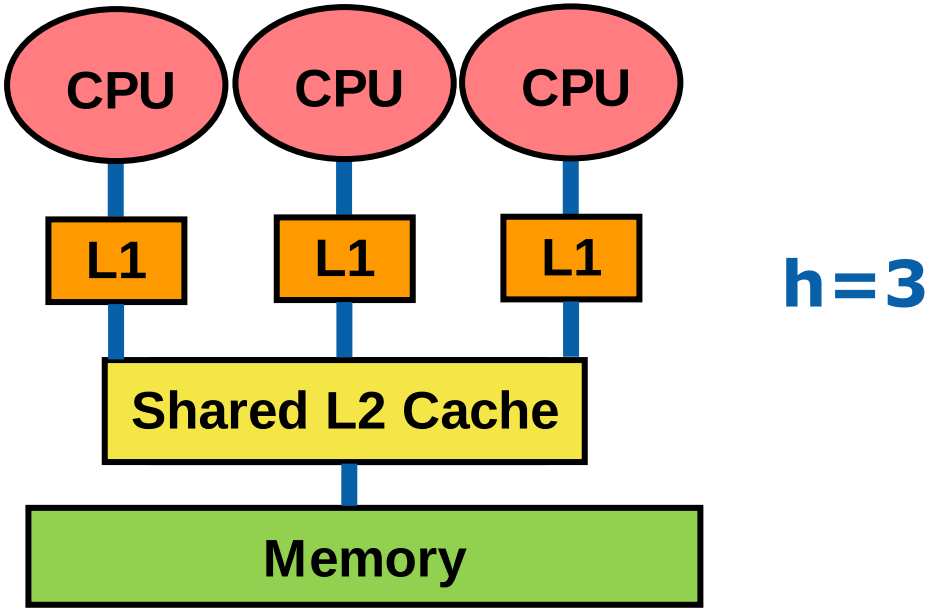
(Symmetric) PMH Model



capacity M_i block size B_i miss cost C_i fanout f_i

PMH captures

- **PEM model** [Arge, Goodrich, Nelson, Sitchinava '08]
  p-processor machine with private caches
- **Shared Cache model discussed in Lecture 1**
- **Multicore Cache model** [Blelloch et al. '08]



h=3

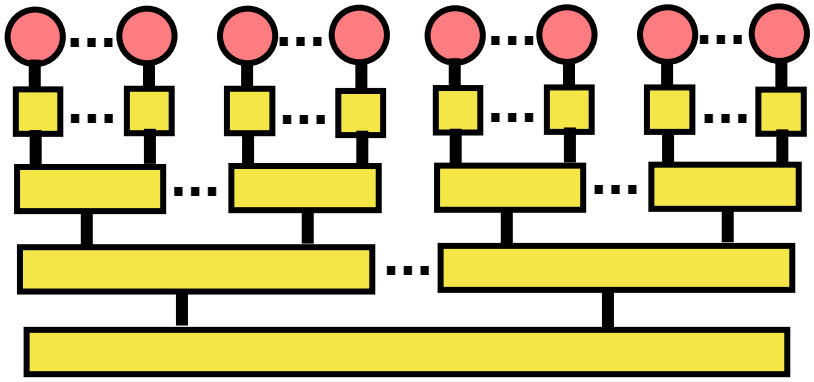
Lecture 2 Outline

- **Modeling the Multicore Hierarchy**
  - PMH model
- **Algorithm Designer's model exposing Hierarchy**
  - Multi-BSP model
- **Quest for a Simplified Hierarchy Abstraction**
- **Algorithm Designer's model abstracting Hierarchy**
  - Parallel Cache-Oblivious (PCO) model
- **Space-Bounded Schedulers**
  - Revisit PCO model

How to Design Algorithms (?)

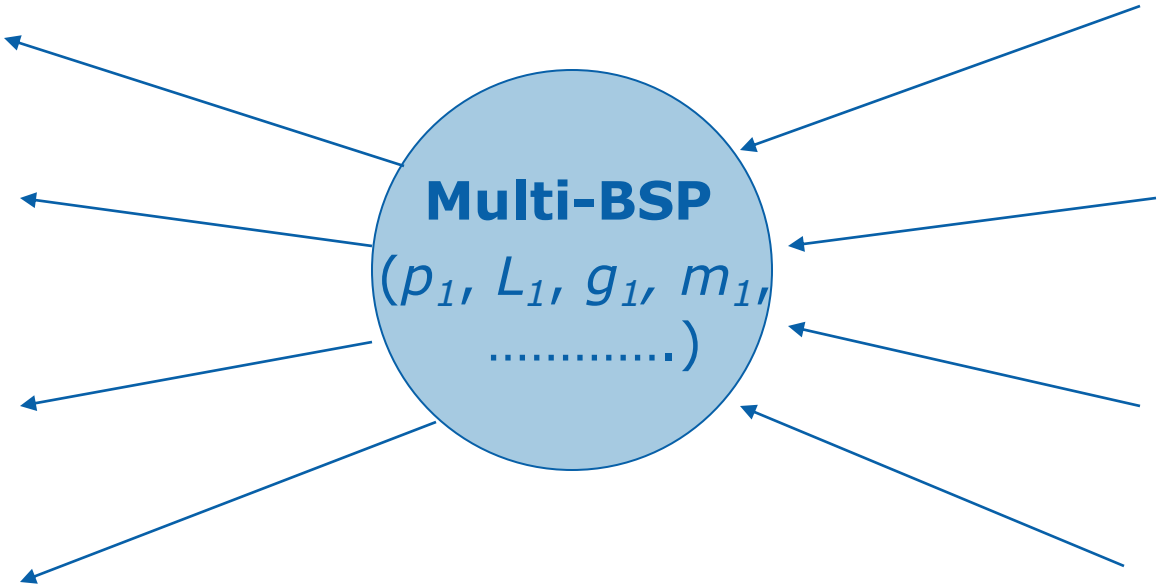
Design to Tree-of-Caches abstraction:

- **Multi-BSP Model** [Valiant '08]
  - 4 parameters/level: cache size, fanout, latency/sync cost, transfer bandwidth
  - Bulk-Synchronous



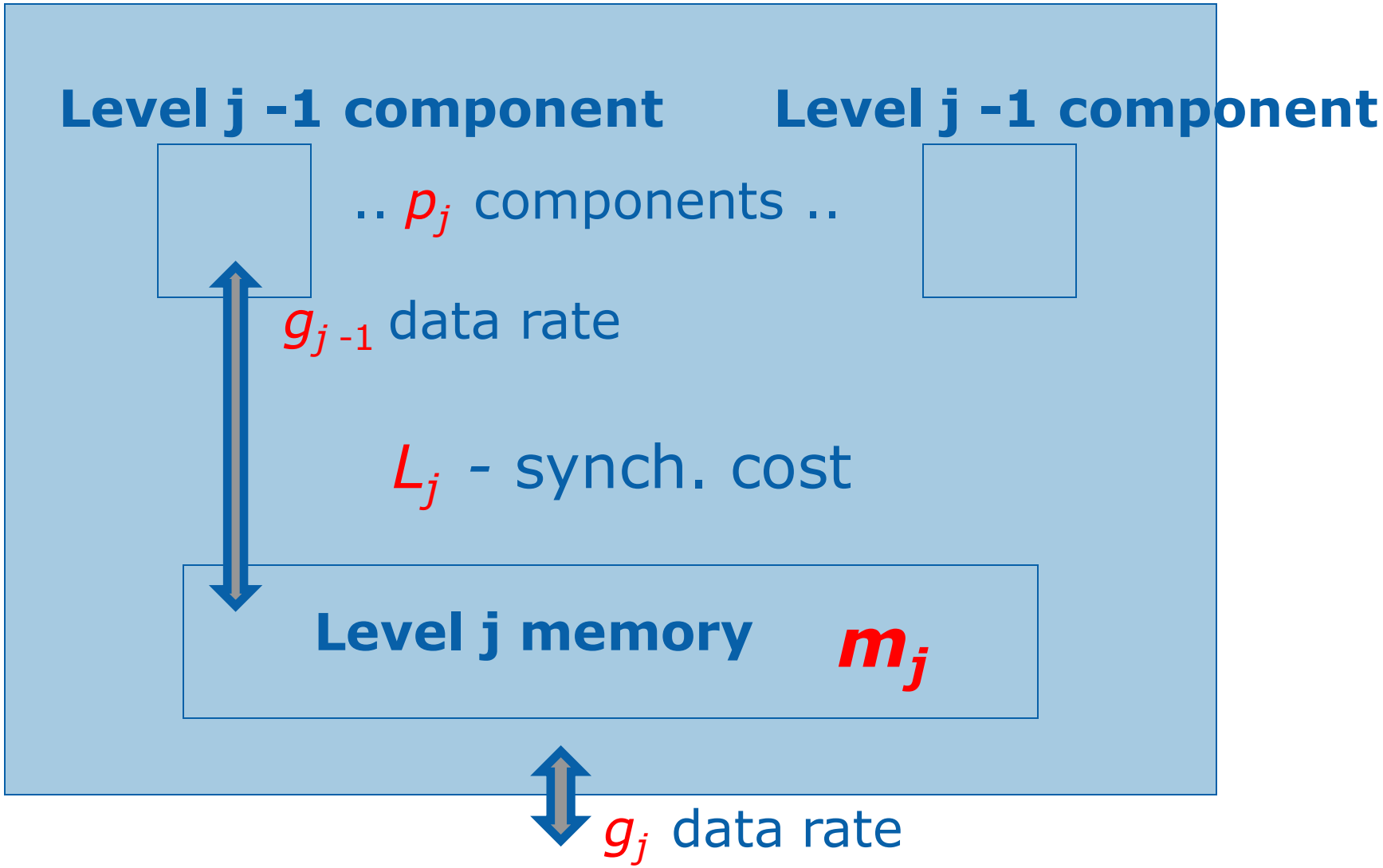
Bridging Models

Hardware **Software**

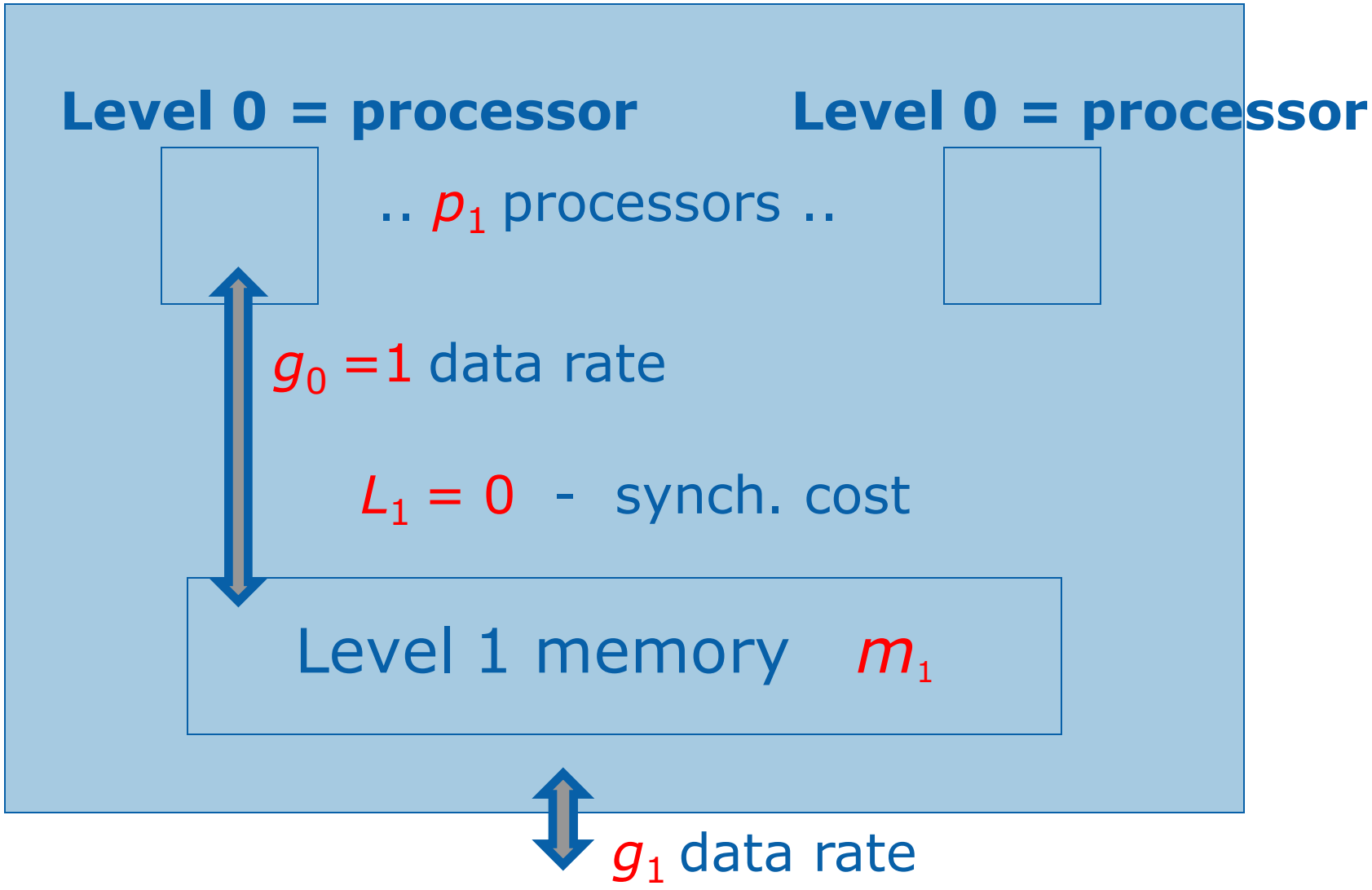


Key: "⟵" = "can efficiently simulate on"

Multi-BSP: Level j component



Multi-BSP: Level 1 component



Multi-BSP

Like BSP except,

1. Not 1 level, but *d* level tree

2. Has memory (cache) size *m* as further parameter at each level.

i.e. Machine *H* has $4d+1$ parameters:

e.g. $d = 3$, and

(p_1, g_1, L_1, m_1) (p_2, g_2, L_2, m_2) (p_3, g_3, L_3, m_3)

Optimal Multi-BSP Algorithms

A Multi-BSP algorithm *A is *optimal with respect to algorithm A* if**

(i) Comp(*A) = Comp(*A*) + low order terms,**

(ii) Comm(*A) = O(Comm(*A*))**

(iii) Synch(*A) = O(Synch(*A*))**

where Comm(*A*), Synch(*A*) are optimal among Multi-BSP implementations, and Comp is total computational cost, and O() constant is independent of the model parameters.

Presents optimal algorithms for Matrix Multiply, FFT, Sorting, etc. (simple variants of known algorithms and lower bounds)

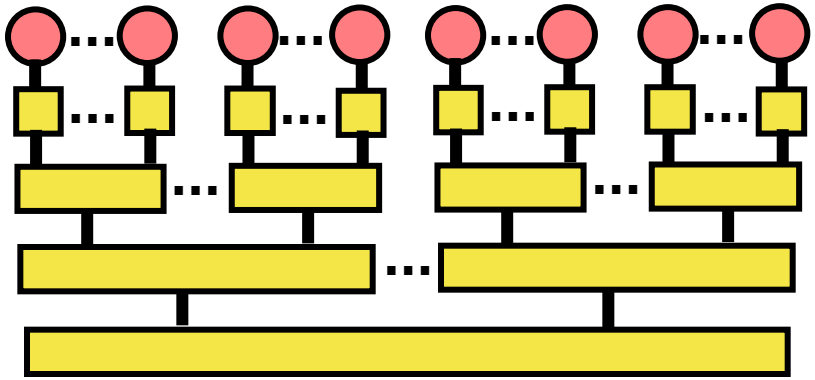
Lecture 2 Outline

- **Modeling the Multicore Hierarchy**
  - PMH model
- **Algorithm Designer's model exposing Hierarchy**
  - Multi-BSP model
- **Quest for a Simplified Hierarchy Abstraction**
- **Algorithm Designer's model abstracting Hierarchy**
  - Parallel Cache-Oblivious (PCO) model
- **Space-Bounded Schedulers**
  - Revisit PCO model

How to Design Algorithms (?)

Design to Tree-of-Caches abstraction:

- **Multi-BSP Model**
  - 4 parameters/level: cache size, fanout, latency/sync cost, transfer bandwidth
  - Bulk-Synchronous



Our Goal: Be Hierarchy-savvy

- **~ Simplicity of Cache-Oblivious Model**
  - Handles dynamic, irregular parallelism
  - Co-design with smart thread schedulers

Abstract Hierarchy: Simplified View

What yields good hierarchy performance?

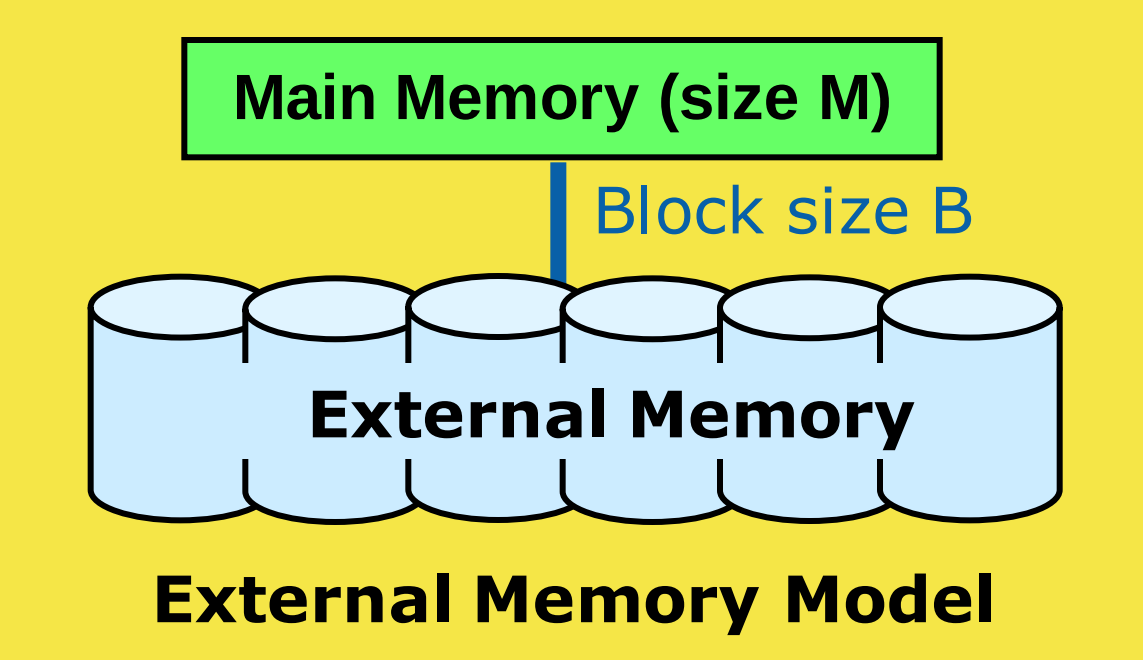
- **Spatial locality: use what's brought in**
  - Popular sizes: Cache lines 64B; Pages 4KB
- **Temporal locality: reuse it**
- **Constructive sharing: don't step on others' toes**

How might one simplify the view?

- **Approach 1: Design to a 2 or 3 level hierarchy (?)**
- **Approach 2: Design to a sequential hierarchy (?)**
- **Approach 3: Do both (??)**

Sequential Hierarchies: Simplified View

- **External Memory Model**
  - See [Vitter '01]

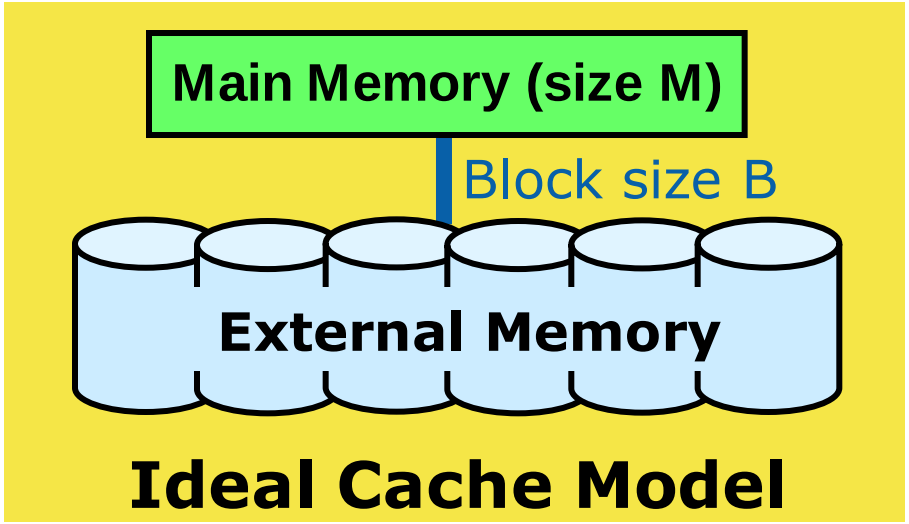


- **Simple model**
- **Minimize I/Os**
- **Only 2 levels**
- **Only 1 "cache"**

Can be good choice if bottleneck is last level

Sequential Hierarchies: Simplified View

- **Cache-Oblivious Model** [Frigo et al. '99]



Twist on EM Model:
M & B unknown to Algorithm

simple model

Key Algorithm Goal: Good performance for any M & B

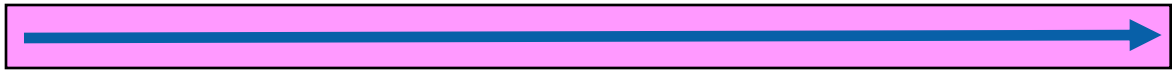
Key Goal ⇨ Guaranteed good cache performance at all levels of hierarchy

Single CPU only (All caches shared)

Encourages Hierarchical Locality

Example Paradigms Achieving Key Goal

- **Scan: e.g., computing the sum of N items**



N/B misses, for any B (optimal)

- **Divide-and-Conquer: e.g., matrix multiply C=A*B**

$A_{11}*B_{11}$ + $A_{12}*B_{21}$	$A_{11}*B_{12}$ + $A_{12}*B_{22}$	=	A_{11}	A_{12}	*	B_{11}	B_{12}
$A_{21}*B_{11}$ + $A_{22}*B_{21}$	$A_{21}*B_{12}$ + $A_{22}*B_{22}$		A_{21}	A_{22}		B_{21}	B_{22}

Divide: Recursively compute $A_{11}*B_{11}$,..., $A_{22}*B_{22}$

Conquer: Compute 4 quadrant sums

$O(N^2/B + N^3/(B*\sqrt{M}))$ misses (optimal)

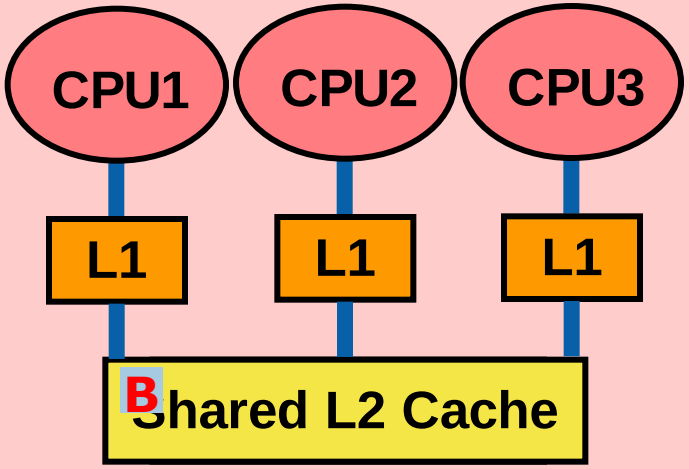
Uses Recursive Z-order Layout

Multicore Hierarchies: Key Challenge

- **Theory underlying Ideal Cache Model falls apart once introduce parallelism:**

Good performance for any M & B on 2 levels
DOES NOT
imply good performance at all levels of hierarchy

Key reason: Caches not fully shared



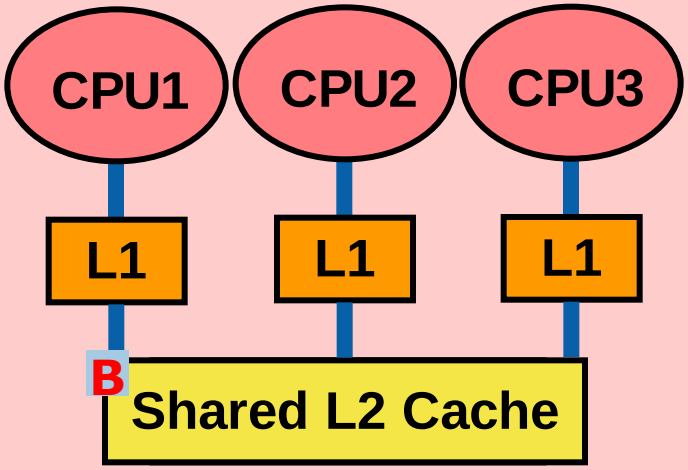
What's good for CPU1 is often bad for CPU2 & CPU3

e.g., all want to write B at ≈ the same time

Multicore Hierarchies
Key New Dimension: Scheduling

Key new dimension:
Scheduling of parallel threads

Has LARGE impact on cache performance



Recall our problem scenario:

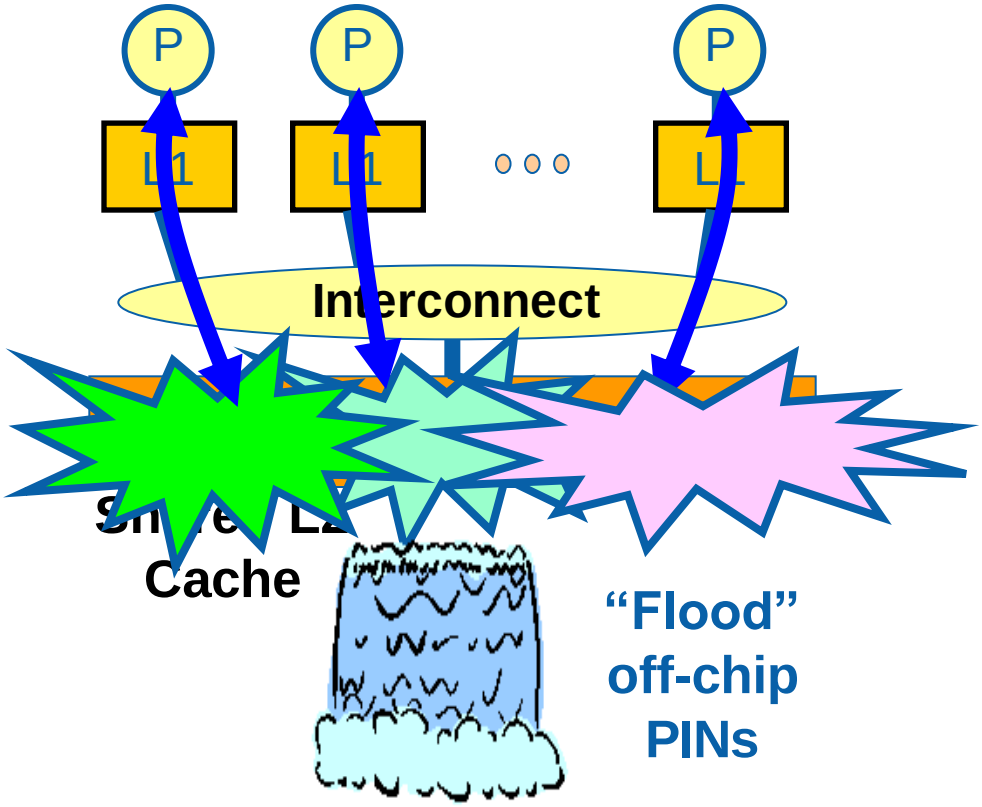
all CPUs want to write B at ≈ the same time

Can mitigate (but not solve) if can schedule the writes to be far apart in time

Constructive Sharing

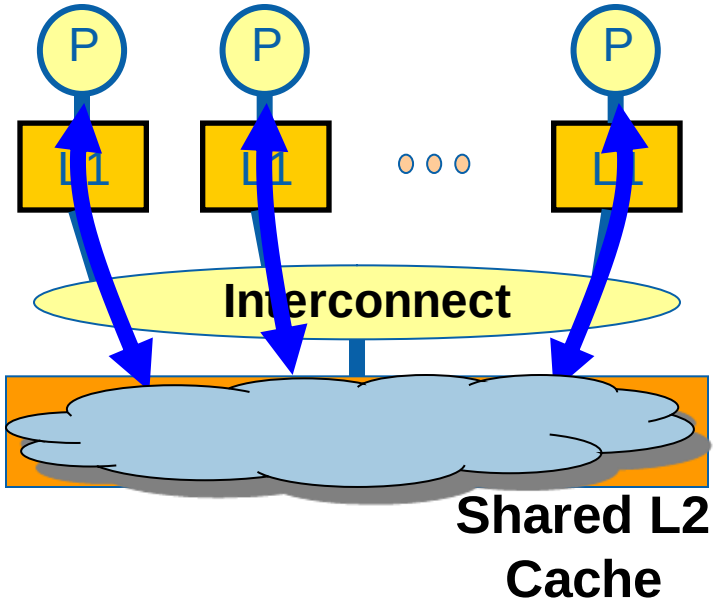
Destructive

- compete for the limited on-chip cache



Constructive

- share a largely overlapping working set



Recall: Low-Span + Cache-Oblivious

- **Guarantees on scheduler's cache performance depend on the computation's depth D**
  - E.g., Work-stealing on single level of private caches:

Thrm: For any computation w/ fork-join parallelism, O(M P D / B) more misses on P cores than on 1 core

- **Approach: Design parallel algorithms with**
  - Low span, and
  - Good performance on Cache-Oblivious Model

Thrm: For any computation w/ fork-join parallelism for each level i, only $O(M_i P D / B_i)$ more misses than on 1 core, for **hierarchy of private caches**

But: No such guarantees for general tree-of-caches

Lecture 2 Outline

- **Modeling the Multicore Hierarchy**
  - PMH model
- **Algorithm Designer's model exposing Hierarchy**
  - Multi-BSP model
- **Quest for a Simplified Hierarchy Abstraction**
- **Algorithm Designer's model abstracting Hierarchy**
  - Parallel Cache-Oblivious (PCO) model
- **Space-Bounded Schedulers**
  - Revisit PCO model

Handling the Tree-of-Caches

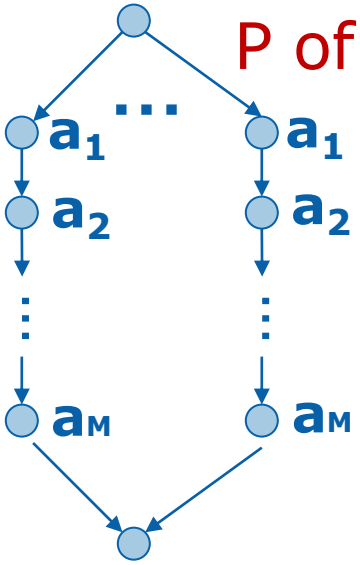
To obtain guarantees for general tree-of-caches:

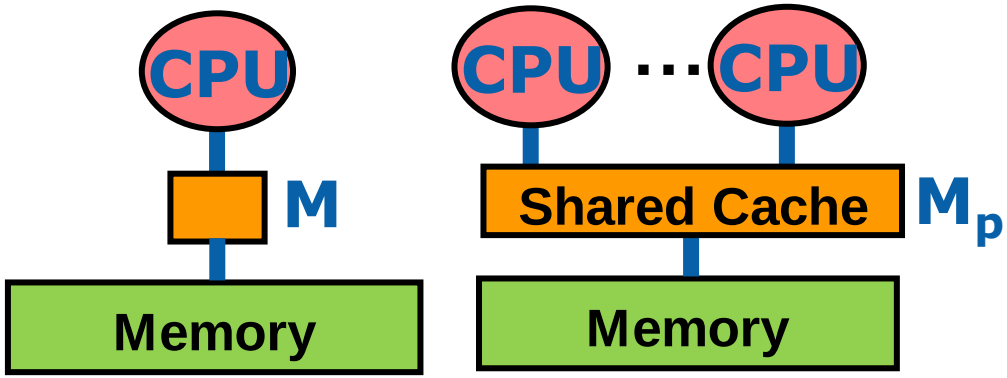
- **We define a Parallel Cache-Oblivious Model**

and

- **A corresponding Space-Bounded Scheduler**

A Problem with Using CO Model





P subtasks: each reading same M/B blocks in same order

Carry Forward rule is too optimistic

Misses in CO model

- **M/B** misses

Any greedy parallel schedule ($\mathbf{M_p} = \mathbf{M}$):

All processors suffer all misses in parallel

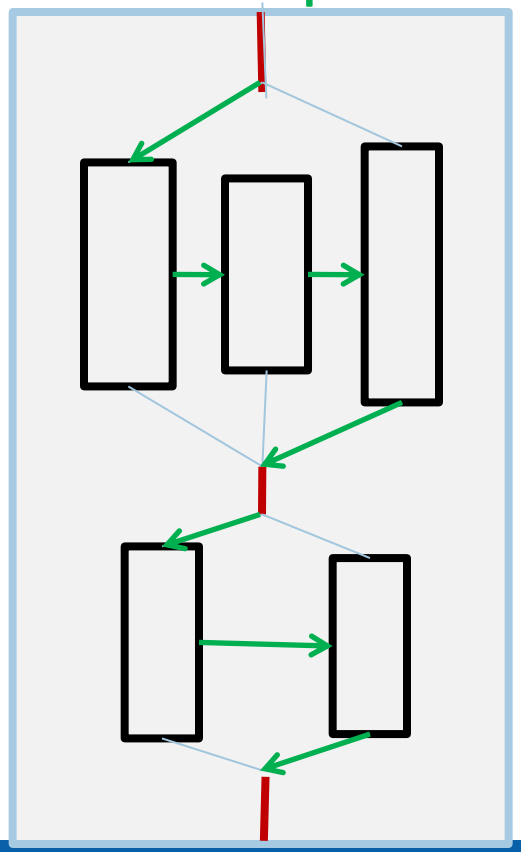
- **P M/B** misses

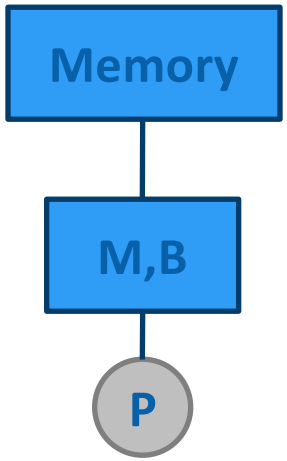
Parallel Cache-Oblivious (PCO) Model

[Blelloch, Fineman, G, Simhadri '11]

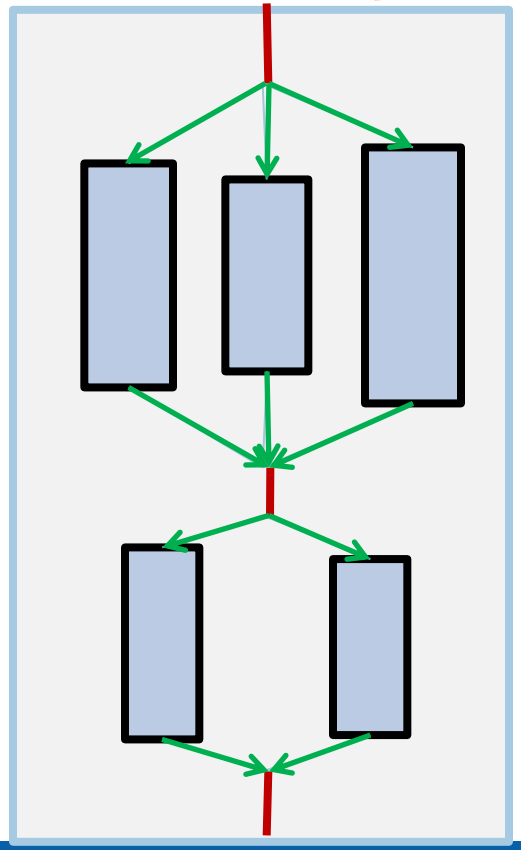
- **Differs from cache-oblivious model in how cache state is carried forward**

Carry forward cache state according to some sequential order





Case 1: task fits in cache

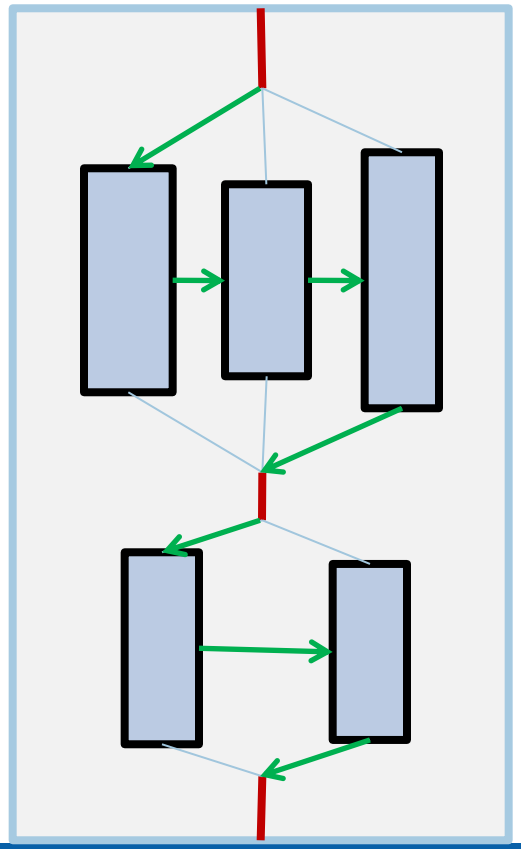


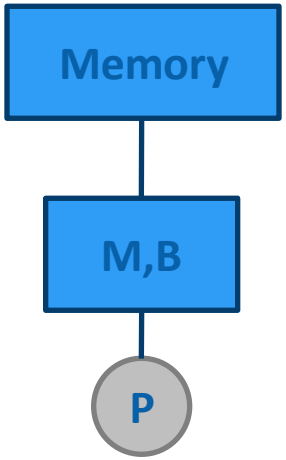
All three subtasks start with same state

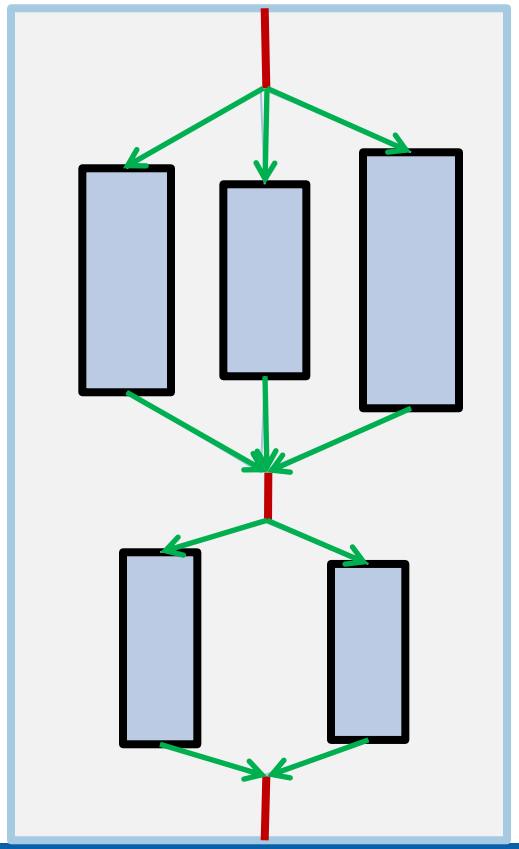
At join, merge state and carry forward

Parallel Cache-Oblivious Model (2)

- **Differs from cache-oblivious model in how cache state is carried forward**







Case 2: Task does not fit in cache

All three subtasks start with empty state

Cache set to empty at join

PCO Cache Complexity Q*

Problem	Span	Cache Complexity Q^*
Scan (prefix sums, etc.)	$O(\log n)$	$O(\lceil n/B \rceil)$
Matrix Transpose ($n \times m$ matrix) [20]	$O(\log(n+m))$	$O(\lceil nm/B \rceil)$
Matrix Multiplication ($\sqrt{n} \times \sqrt{n}$ matrix) [20]	$O(\sqrt{n})$	$O(\lceil n^{1.5}/B \rceil / \sqrt{M+1})$
Matrix Inversion ($\sqrt{n} \times \sqrt{n}$ matrix)	$O(\sqrt{n})$	$O(\lceil n^{1.5}/B \rceil / \sqrt{M+1})$
Quicksort [22]	$O(\log^2 n)$	$O(\lceil n/B \rceil (1 + \log\lceil n/(M+1) \rceil))$
Sample Sort [10]	$O(\log^2 n)$	$O(\lceil n/B \rceil \lceil \log_{M+2} n \rceil)$
Sparse-Matrix Vector Multiply [10] (m nonzeros, n^ϵ edge separators)	$O(\log^2 n)$	$O(\lceil m/B + n/(M+1)^{1-\epsilon} \rceil)$
Convex Hull (e.g., see [8])	$O(\log^2 n)$	$O(\lceil n/B \rceil \lceil \log_{M+2} n \rceil)$
Barnes Hut tree (e.g., see [8])	$O(\log^2 n)$	$O(\lceil n/B \rceil (1 + \log\lceil n/(M+1) \rceil))$

- **Bounds assume M = $\Omega(B^2)$**
- **All algorithms are work optimal**
- **Q* bounds match both CO bounds and best sequential algorithm bounds**

See [Blelloch, Fineman, G, Simhadri '11] for details

Lecture 2 Outline

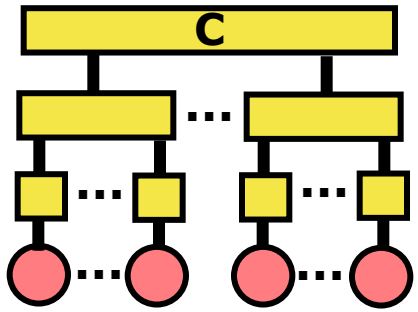
- **Modeling the Multicore Hierarchy**
  - PMH model
- **Algorithm Designer's model exposing Hierarchy**
  - Multi-BSP model
- **Quest for a Simplified Hierarchy Abstraction**
- **Algorithm Designer's model abstracting Hierarchy**
  - Parallel Cache-Oblivious (PCO) model
- **Space-Bounded Schedulers**
  - Revisit PCO model

Space-Bounded Scheduler

[Chowdhury, Silvestri, Blakeley, Ramachandran '10]

Key Ideas:

- Schedules a dynamically unfolding parallel computation on a tree-of-caches hierarchy
- Computation exposes lots of parallelism
- Assumes space use (working set sizes) of tasks are known (can be suitably estimated)
- Assigns a task to a cache C that fits the task's working set. Reserves the space in C. Recurses on the subtasks, using the CPUs and caches that share C (below C in the diagram)



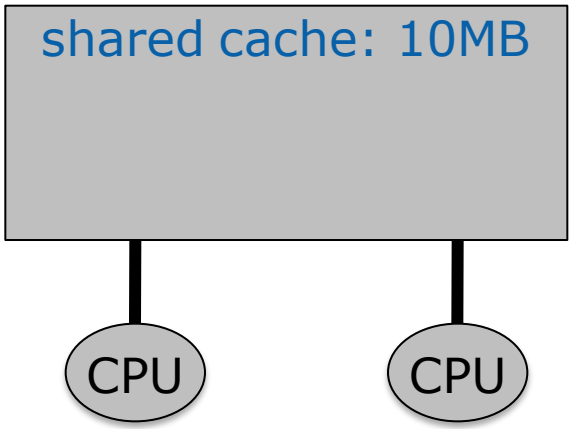
Space-Bounded Scheduler

Advantages over WS scheduler

- **Avoids cache overloading for shared caches**
- **Exploits cache affinity for private caches**

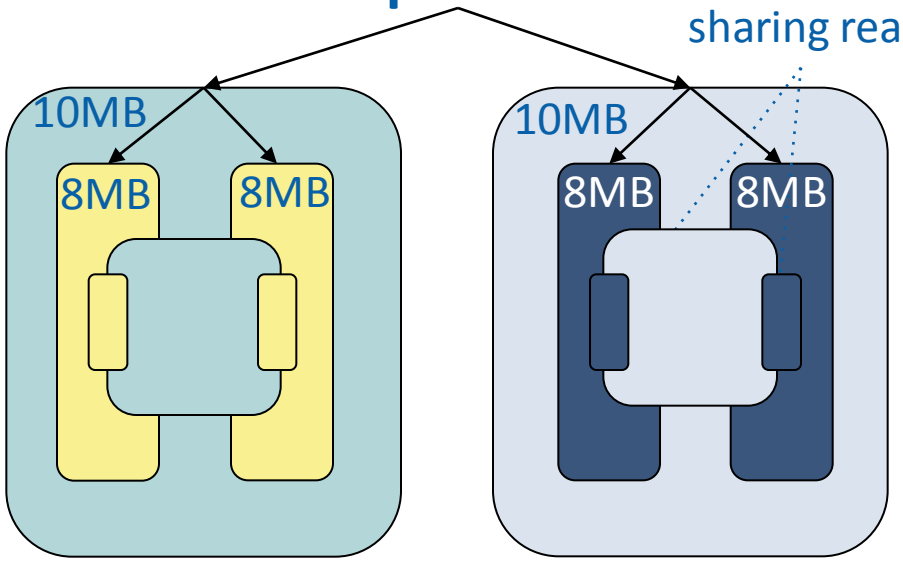
Problem with WS Scheduler: Cache overloading

Hierarchy
(focus on 1 cache)

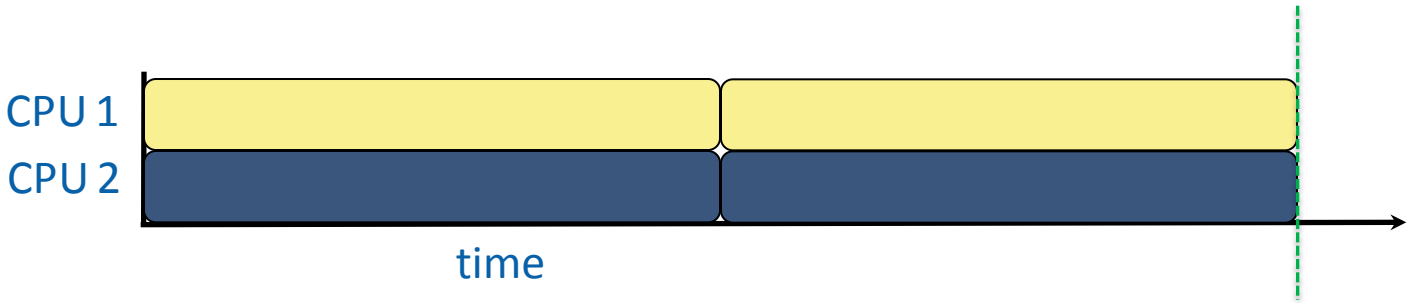


Computation

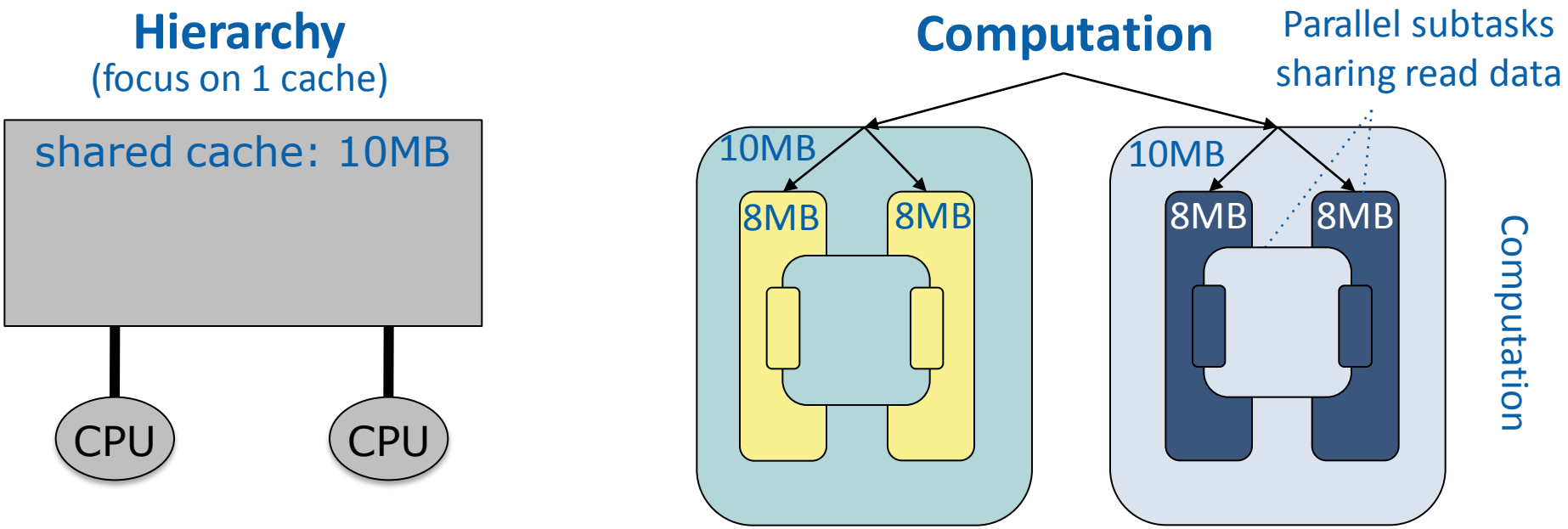
Parallel subtasks sharing read data



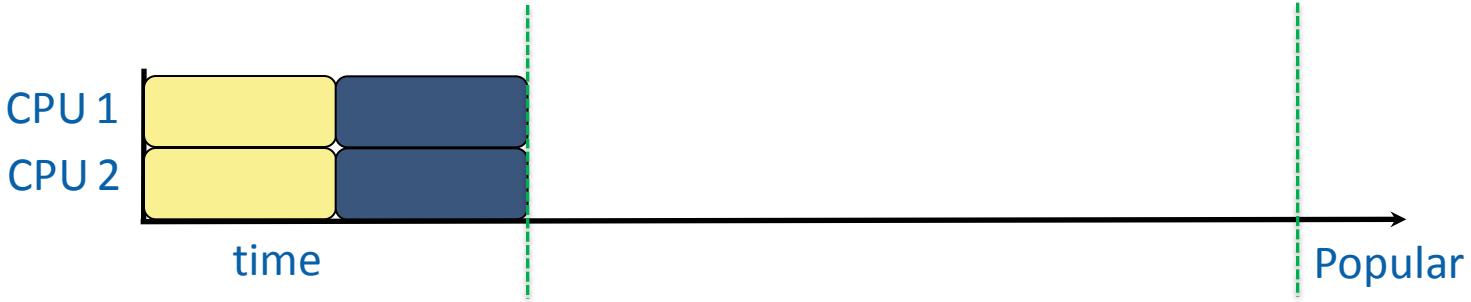
Overloaded cache introduces more cache (capacity) misses



Space-Bounded Scheduler avoids cache overloading

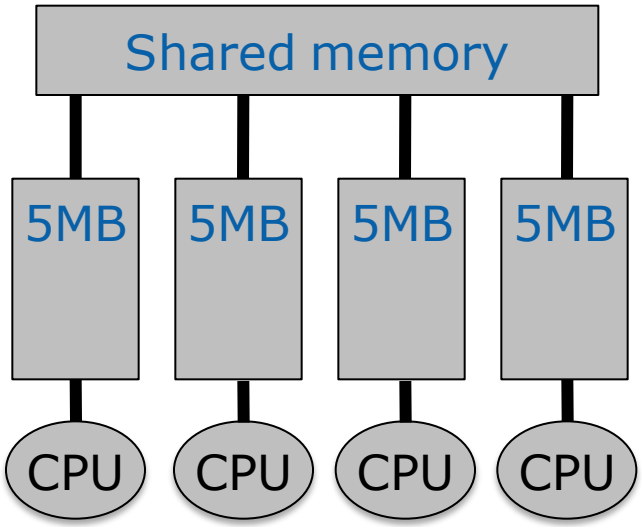


Does not overload the cache, so fewer cache misses

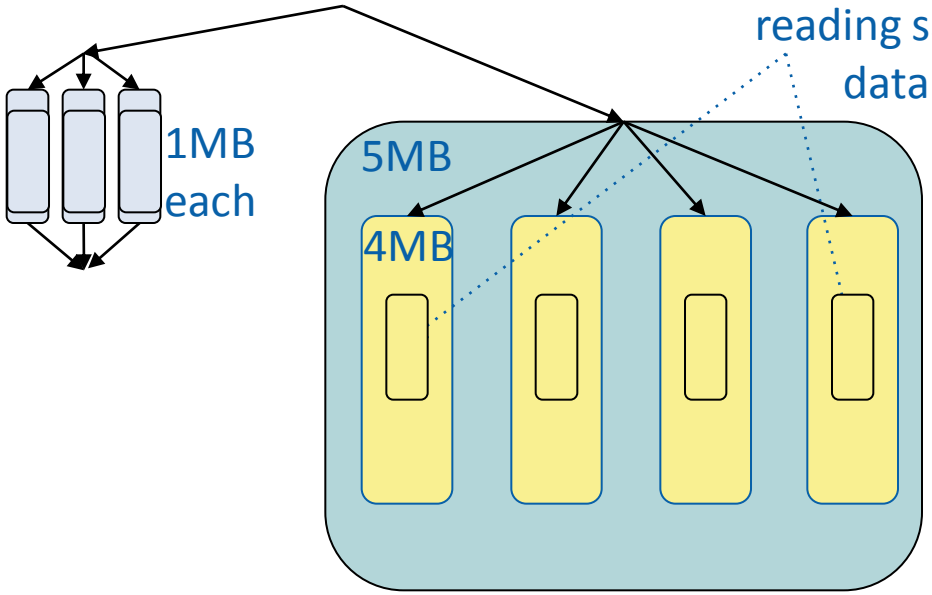


Problem with WS Scheduler (2): Ignoring cache affinity

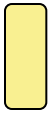
Hierarchy

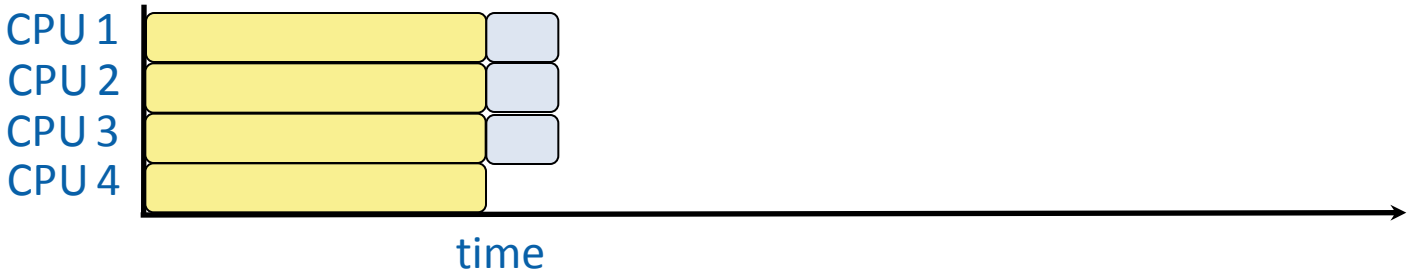


Computation

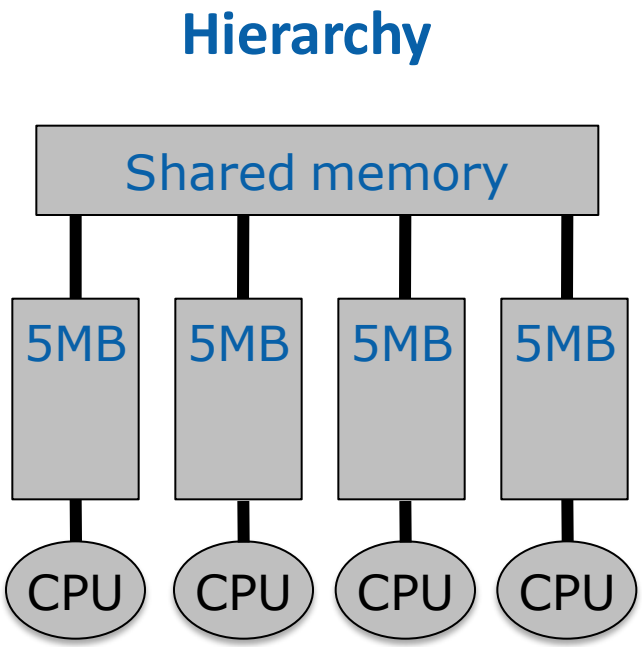


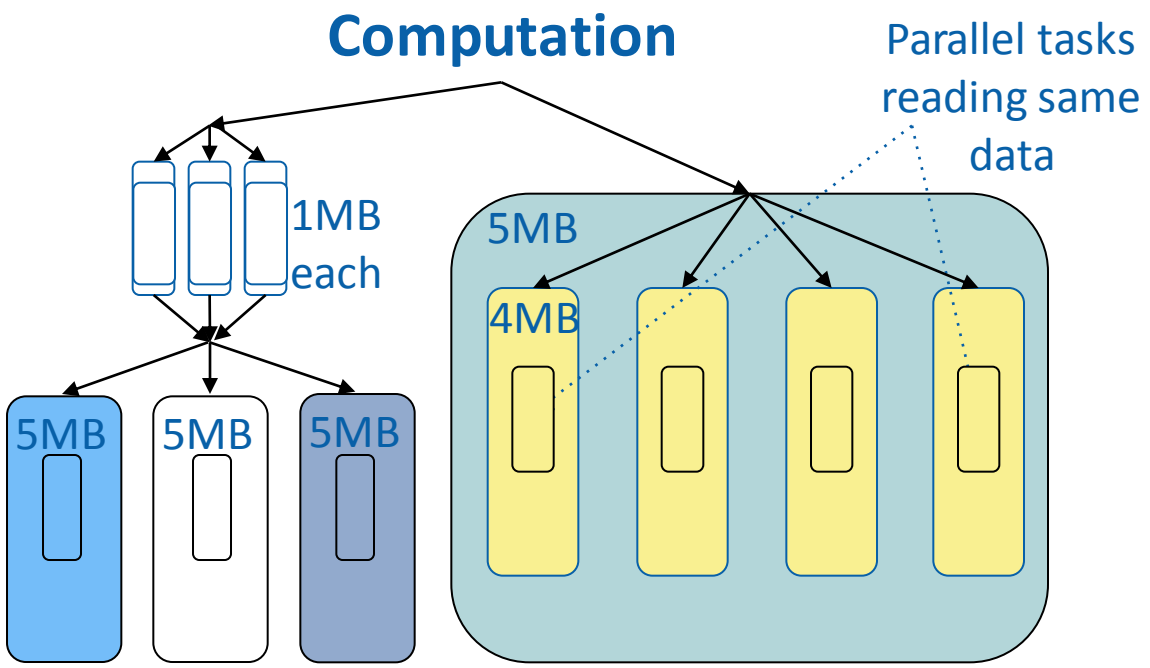
Schedules any available task when a processor is idle

All  experience all cache misses and run slowly



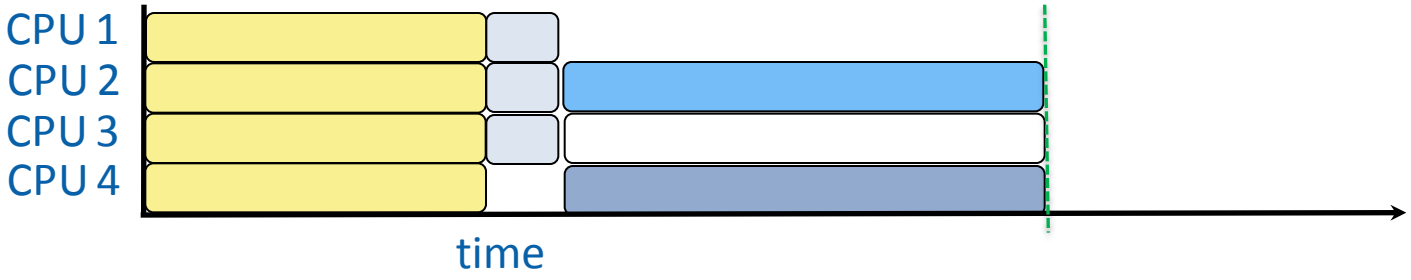
Problem with WS Scheduler: Ignoring cache affinity





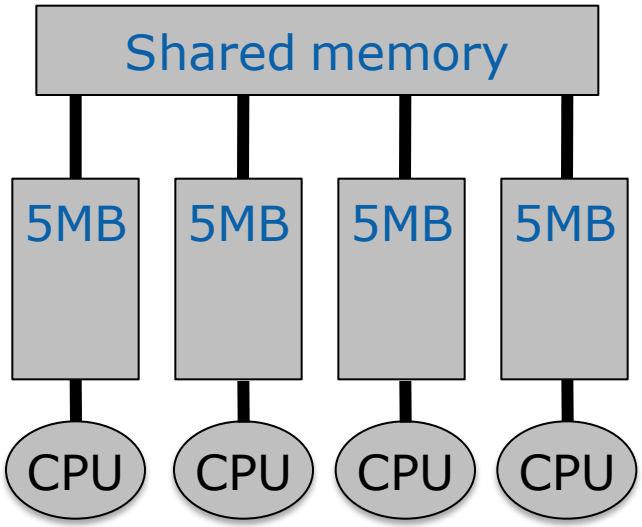
Schedules any available task when a processor is idle

All ▮ experience all cache misses and run slowly

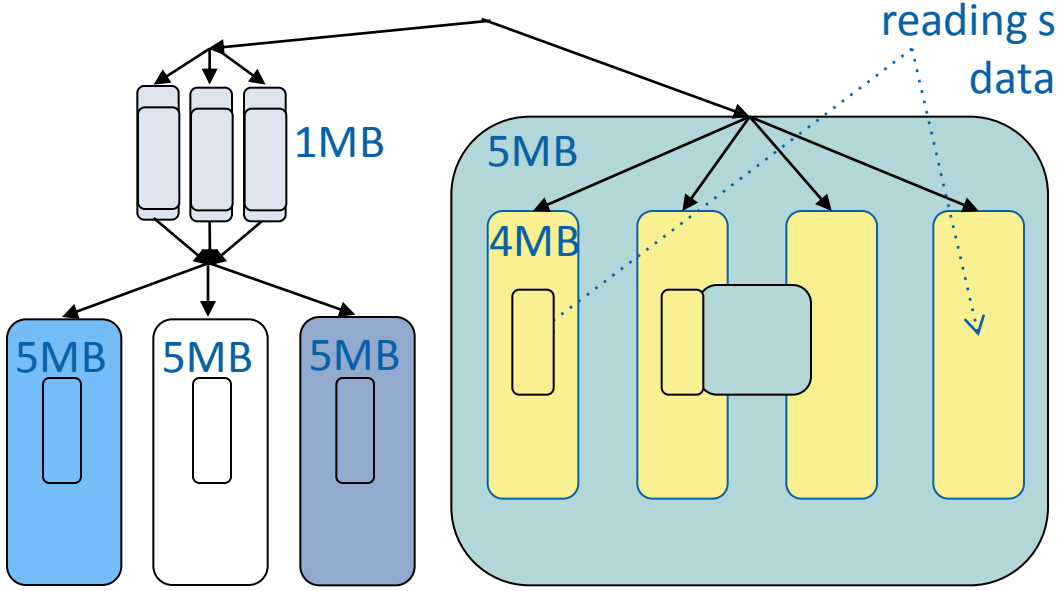


Space-Bounded Scheduler exploits cache affinity

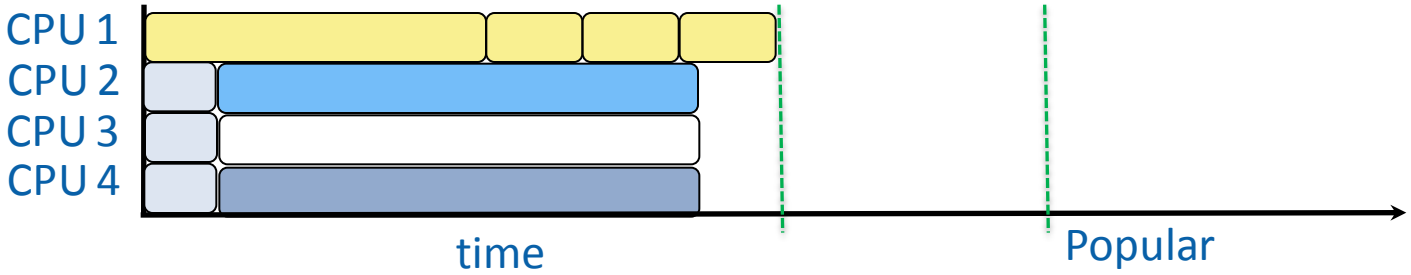
Hierarchy



Computation

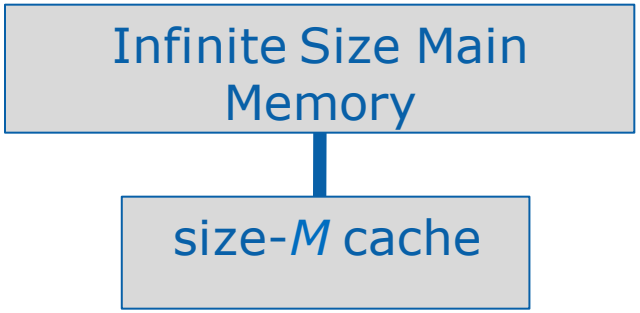


Pin 5MB task to cache to exploit affinity among subtasks

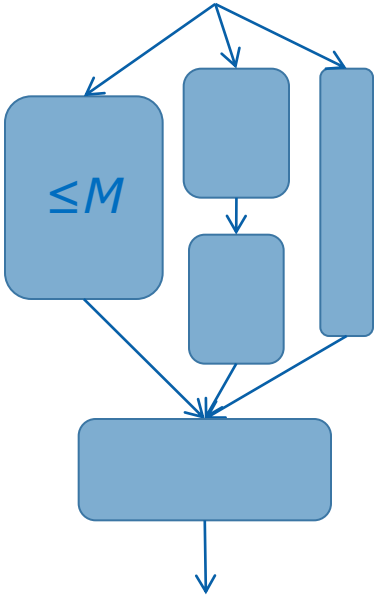


Analysis Approach

Goal: Algorithm analysis should remain lightweight and agnostic of the machine specifics



Analyze for a single cache level using PCO model



Unroll algorithm to tasks that fit in cache
Analyze each such task separately, starting from an empty cache

Cache complexity Q*(M) = Total # of misses, summed across all tasks

Analytical Bounds

[Blelloch, Fineman, G, Simhadri '11]

- **Guarantees provided by our Space-Bounded Scheduler:**

Cache costs: optimal $\sum_{\text{levels}} Q^*(M_i) \times C_i$

where C_i is the miss cost for level i caches

Running time: for "sufficiently balanced" computations:

optimal $O(\sum_{\text{levels}} Q^*(M_i) \times C_i / P)$ time on P cores

Our theorem on running time also allows arbitrary imbalance, with the performance depending on an imbalance penalty

Motivation for Imbalance Penalty

Tree-of-Caches

- **Each subtree has a given amount of compute & cache resources**
- **To avoid cache misses from migrating tasks, would like to assign/pin task to a subtree**
- **But any given program task may not match both**
  - E.g., May need large cache but few processors
- **We extend PCO with a cost metric that charges for such space-parallelism imbalance**
  - Attribute of algorithm, not hierarchy
  - Need minor additional assumption on hierarchy

Multi-core Computing Lectures:
Progress-to-date on Key Open Questions

- **How to formally model multi-core hierarchies?**
- **What is the Algorithm Designer's model?**
- **What runtime task scheduler should be used?**
- **What are the new algorithmic techniques?**
- **How do the algorithms perform in practice?**

NEXT UP

Lecture #3: Extensions

References

[Alpern, Carter, Ferrante '93] B. Alpern, L. Carter, and J. Ferrante. Modeling parallel computers as memory hierarchies. Programming Models for Massively Parallel Computers, 1993

[Arge, Goodrich, Nelson, Sitchinava '08] L. Arge, M. T. Goodrich, M. Nelson, and N. Sitchinava. Fundamental parallel algorithms for private-cache chip multiprocessors. ACM SPAA, 2008

[Blelloch et al. '08] G. E. Blelloch, R. A. Chowdhury, P. B. Gibbons, V. Ramachandran, S. Chen, and M. Kozuch. Provably good multicore cache performance for divide-and-conquer algorithms. ACM-SIAM SODA, 2008

[Blelloch, Fineman, G, Simhadri '11] G. E. Blelloch, J. T. Fineman, P. B. Gibbons, and H. V. Simhadri. Scheduling Irregular Parallel Computations on Hierarchical Caches. ACM SPAA, 2011

[Chowdhury, Silvestri, Blakeley, Ramachandran '10] R. A. Chowdhury, F. Silvestri, B. Blakeley, and V. Ramachandran. Oblivious algorithms for multicores and network of processors. IPDPS, 2010

[Frigo et al. '99] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-Oblivious Algorithms. IEEE FOCS 1999

[Valiant '08] L. G. Valiant. A bridging model for multi-core computing. ESA, 2008

[Vitter '01] J. S. Vitter. External memory algorithms and data structures. ACM Computing Surveys 33:2, (2001)