Cost Models for Locality and Parallelism

Harsha Vardhan Simhadri
Carnegie Mellon → Lawrence Berkeley Lab

For scalability…

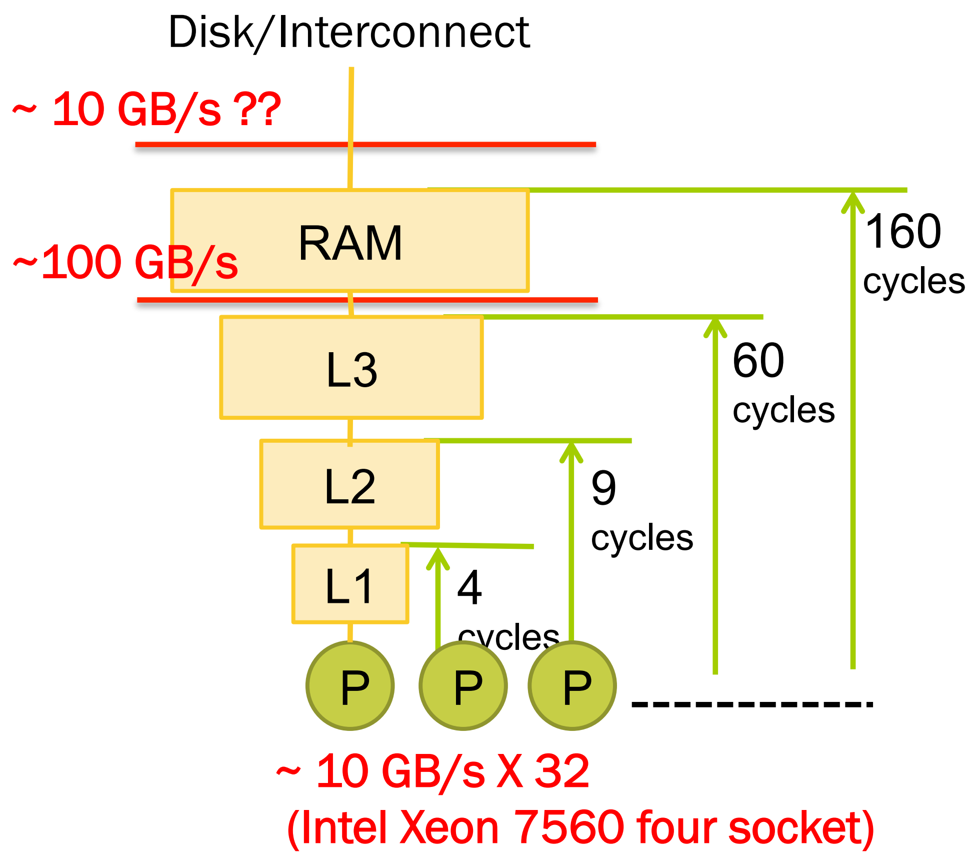
Algorithms need

- Locality

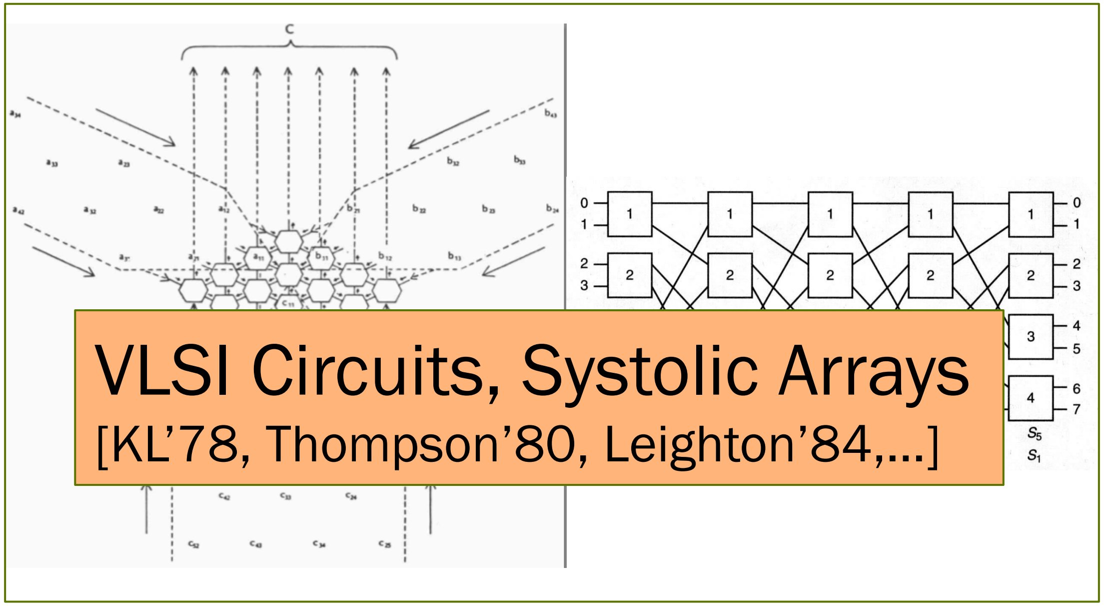
  minimize distance and amount of data moved

- Parallelism

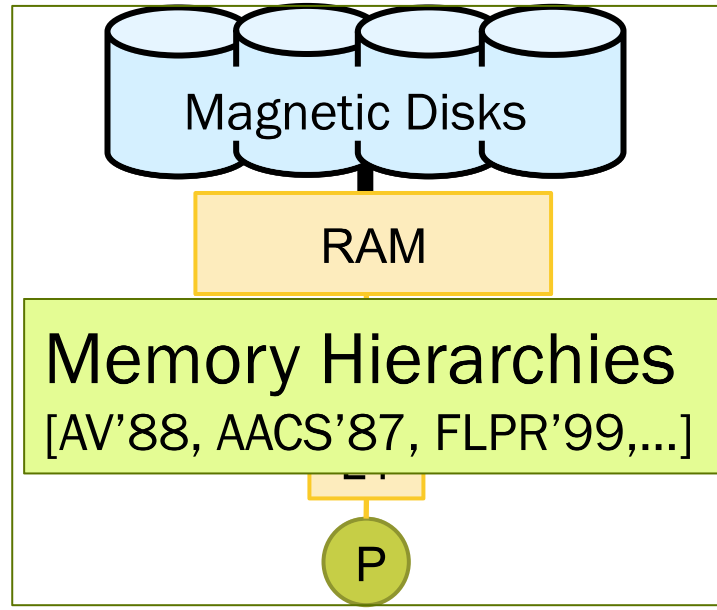
  maximize number of simultaneous operations



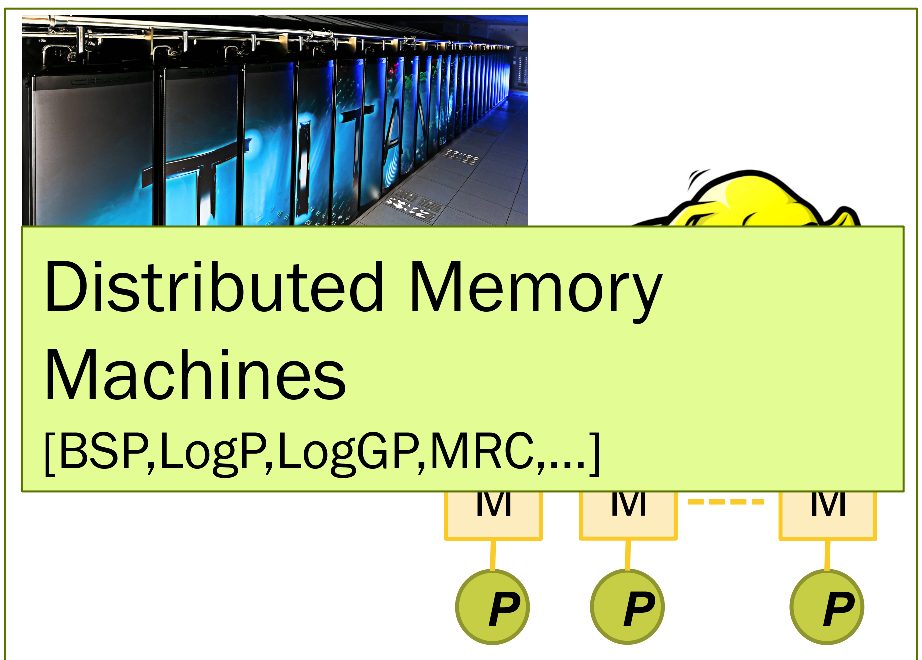
Formal definition in several contexts.



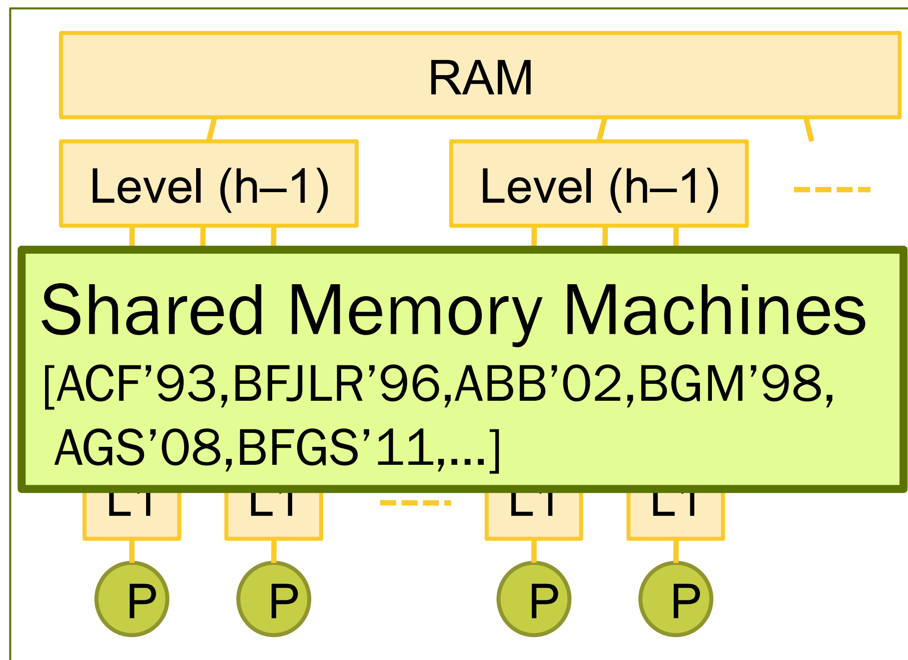
VLSI Circuits, Systolic Arrays
[KL'78, Thompson'80, Leighton'84,...]



Memory Hierarchies
[AV'88, AACS'87, FLPR'99,...]



Distributed Memory Machines
[BSP,LogP,LogGP,MRC,...]

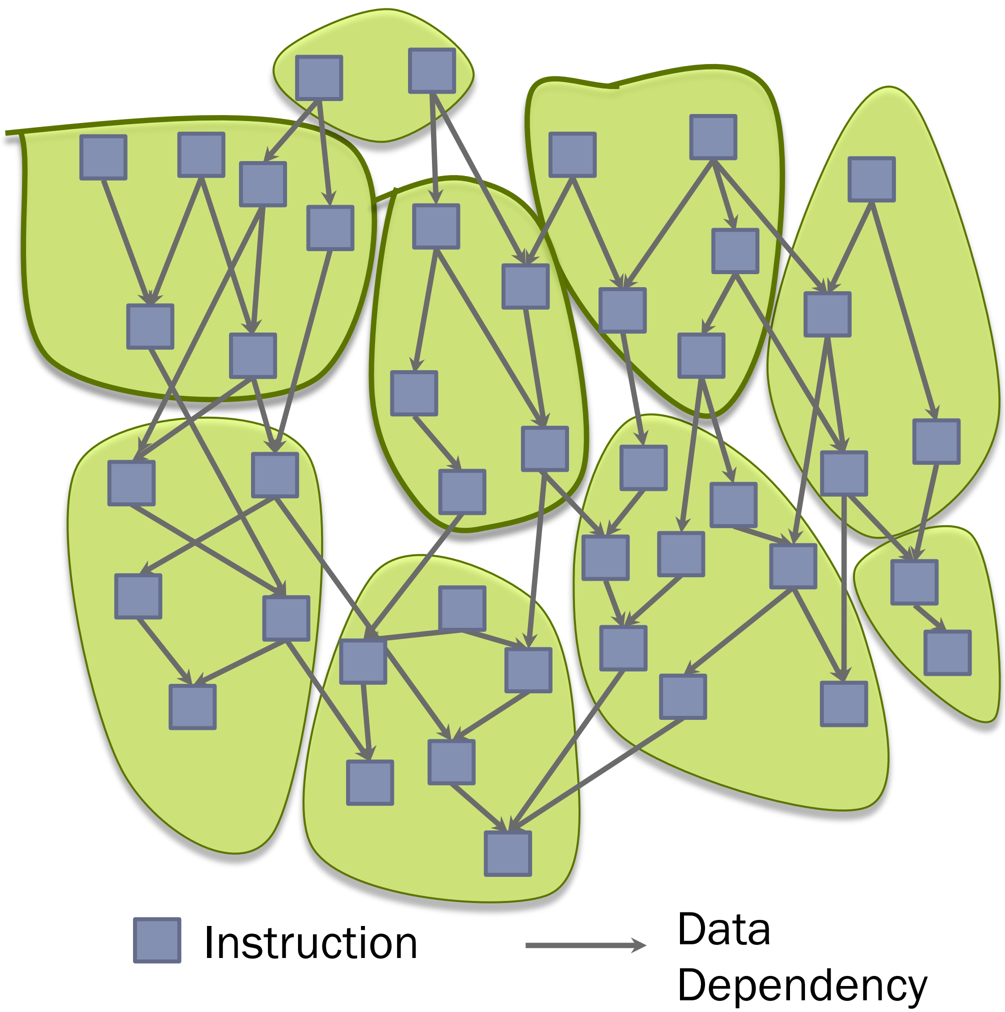


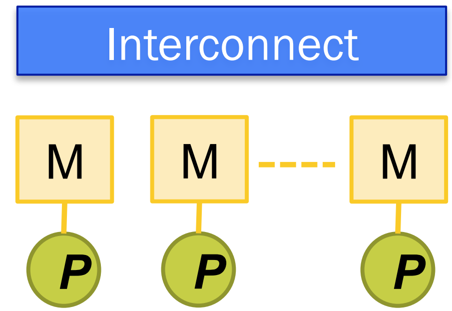
Shared Memory Machines
[ACF'93,BFJLR'96,ABB'02,BGM'98,
AGS'08,BFGS'11,...]

This Talk

- Cost Models for
  – Distributed Memory, Shared Memory, Hierarchies
- Lower Bounds, Upper Bounds (Algorithms)
- Machine-Centric vs Program-Centric Models

Parallel Program = Directed Acyclic Graph

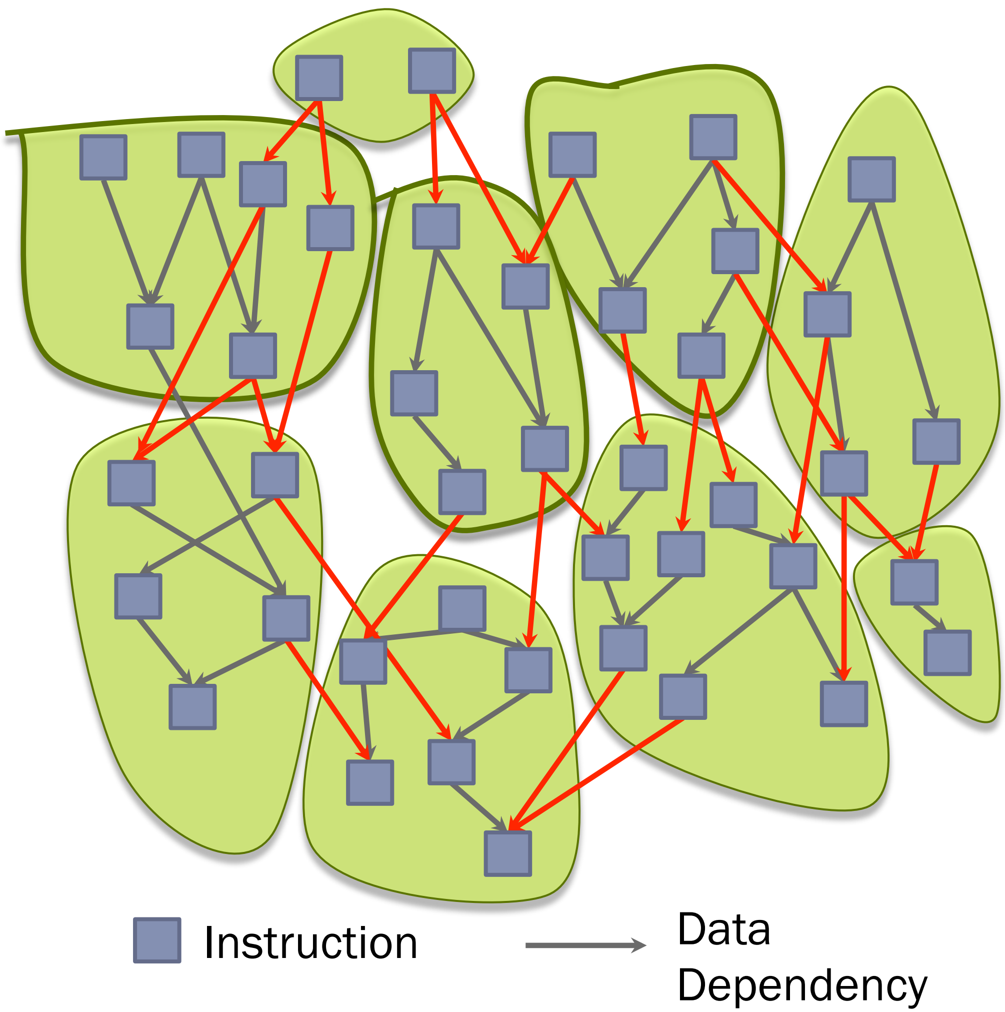


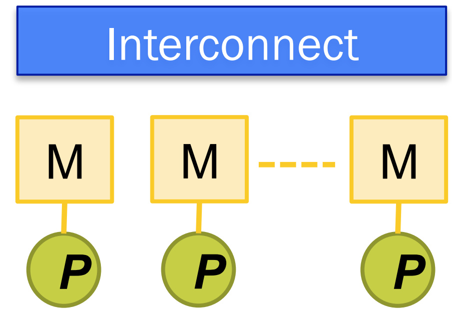


Execution =

Set of vertex partitions
s.t. partitions “fit” in M memory,
contiguous in DAG
(a<b<c, a,c2X (part.)) b2X)
+
mapping from partitions to processors

Communication cost





Data moved

Amount = #edges across partition
(#words, #messages)

Distance = length the edge travel
in the interconnect

Cost Model for Locality
(assume uniform internode cost)

¯#words + °#msgs

Cost Model for Time

Assume interconnect has same cost between all nodes

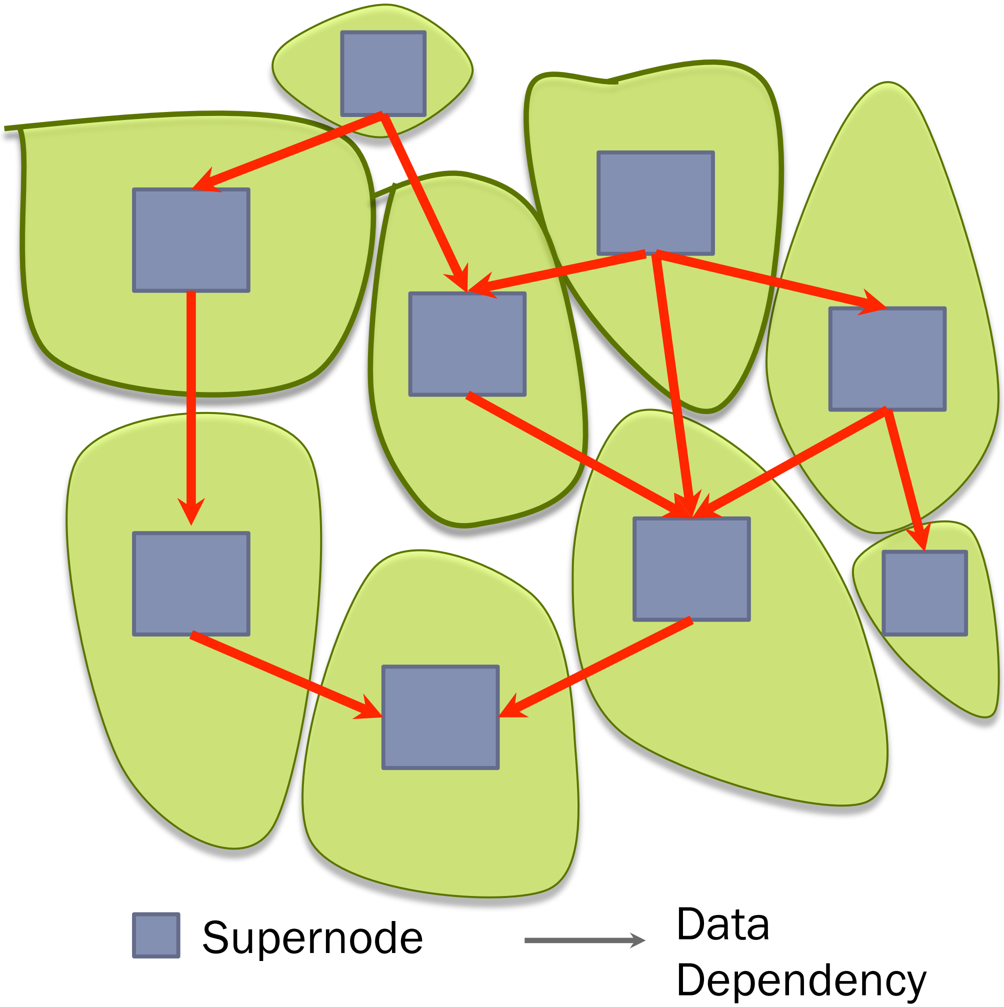
Cost of a DAG for on a schedule on **p** machines

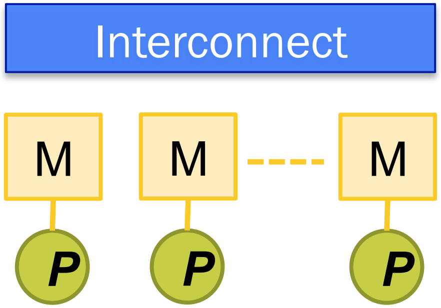
partitioning $\{X_i\}$,

and mapping ***f*** : partitions→procs

T = $\max_p \left\{ \sum_{f(X)=p}$ ®#instrs + ¯#words +°#msgs $\right\}$

Parallelizability



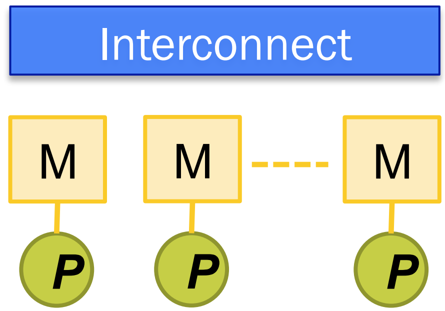


Parallelizability

Max #procs where T stops improving (for best partitioning and schedule)

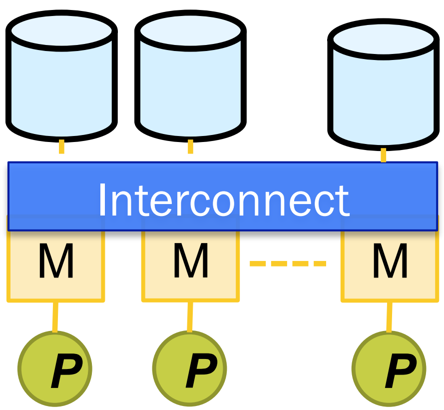
Depends on length of critical path in Partitioned DAG (effective depth)

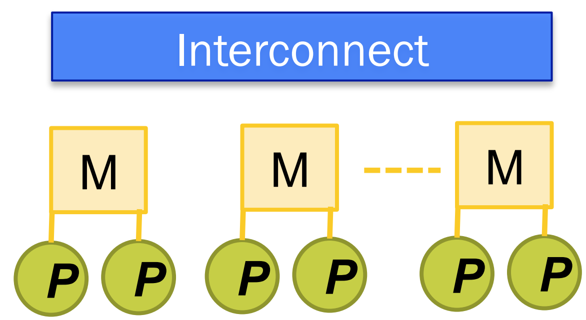
Extensions...



Different Interconnect topologies

Relatable to some form of graph partitioning

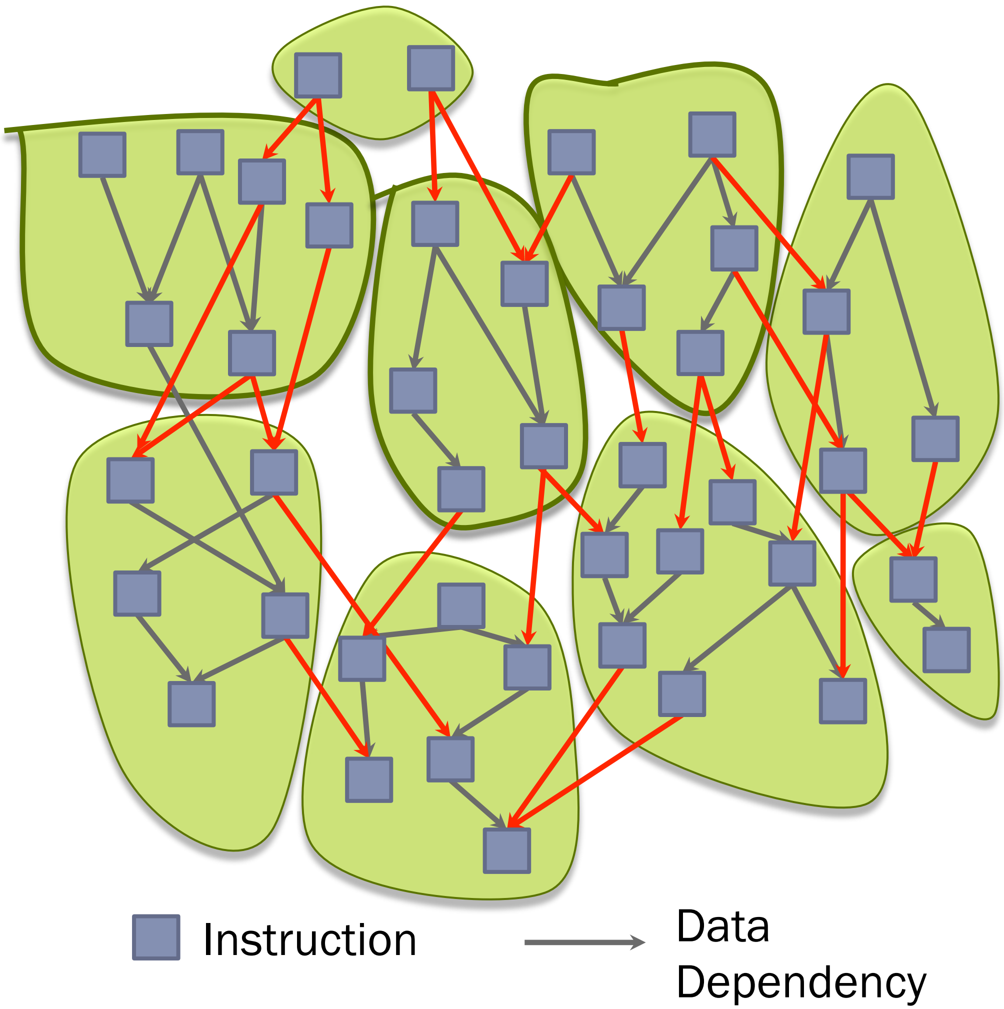




Upper Bounds (Algorithms)

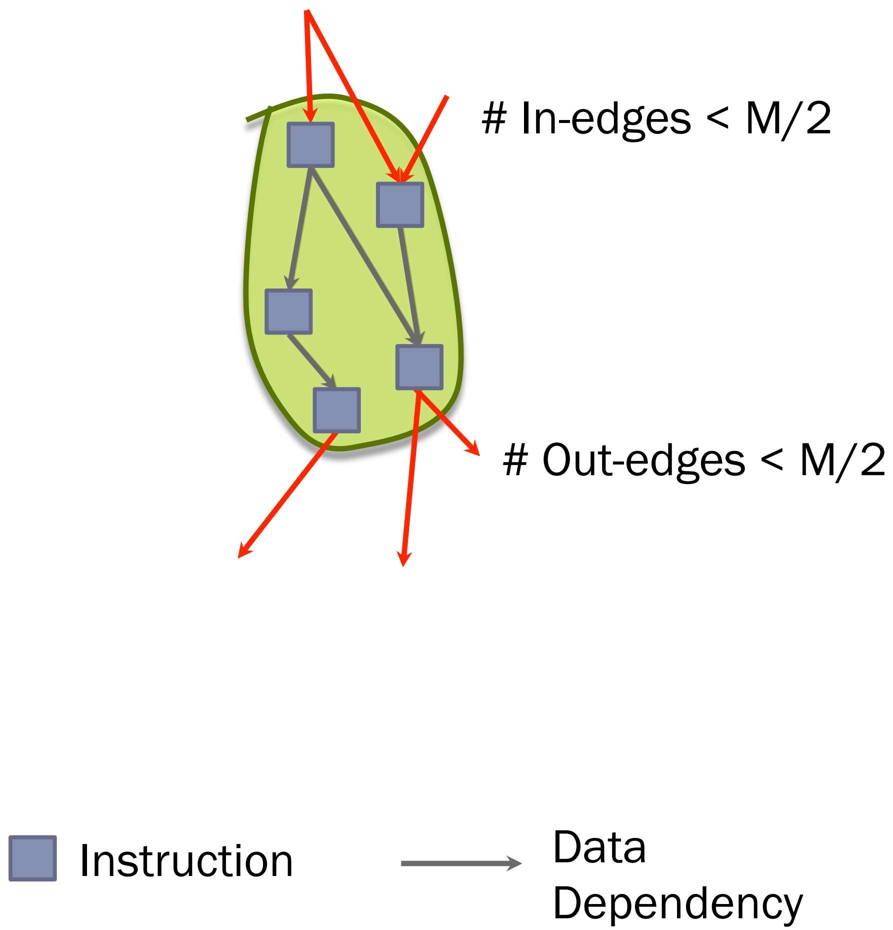
- Upper Bound = Algorithm DAG
  + Schedule (partitioning, mapping)
- Dense Matrix Multiplication (n = M*p)
  - $O(n^{1.5})$ Instr
  - $O(n^{1.5}/M^{0.5})$ Words
  - $O(n^{1.5}/M^{1.5})$ Messages
  - T = $n^{1.5}$ * (® + ¯/$M^{0.5}$ + °/$M^{1.5}$) / p
- Sorting (n = M*p)
  - O(n log n) Instr
  - O(n log n/log M) Words
  - O(n log n/M log M) Messages
  - T = n log n £ (® + ¯/log M + °/M log M) / p

Lower Bounds



Minimum #Words

Lower Bounds



Minimum #Words

Partition such that in-edges and out-edges < M/2 for each partition

What is the partitioning with fewest transfers?

Matrix Mul.:
$\Omega(n^{1.5}/M^{0.5})$
Why? $O(M^{1.5})$ instr. with M data

Sorting
$\Omega(n \log n/\log M)$
Why? $O(M \log M)$ info. with M data

So far

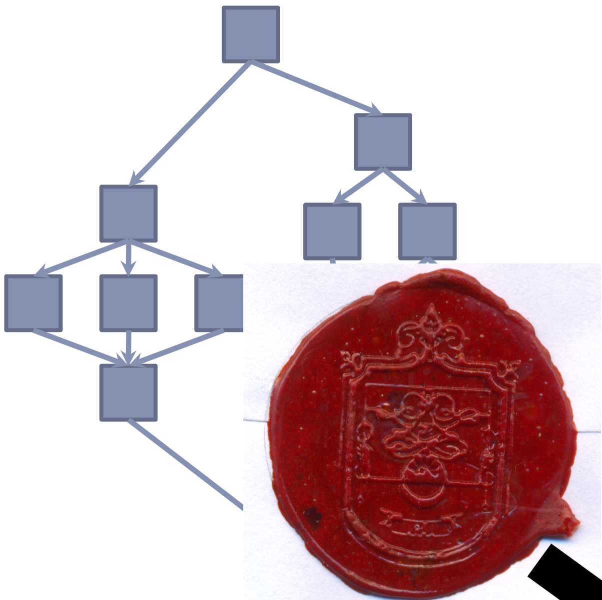
- Program = DAG
- Executions = partitions and mapping of DAG
- A cost model for distributed memory
- Lower bounds / Upper bounds

BUT..

Problem: Unwieldy Model

- Cost model is machine-centric
  - Cost depends on DAG, Machine & Schedule
- What is the best schedule (partition + mapping) for a DAG?
- If cost is tied to machine and schedule, how to study problem complexity?
- Need a cleaner and more portable cost model

Separate Cost Model & Schedule



Scheduler

Scheduler 2

Algorithm designer restructures DAG and optimizes for a general cost model.

Runtime designer translates metrics to performance: Time, Space, Scalability, Communication Costs.

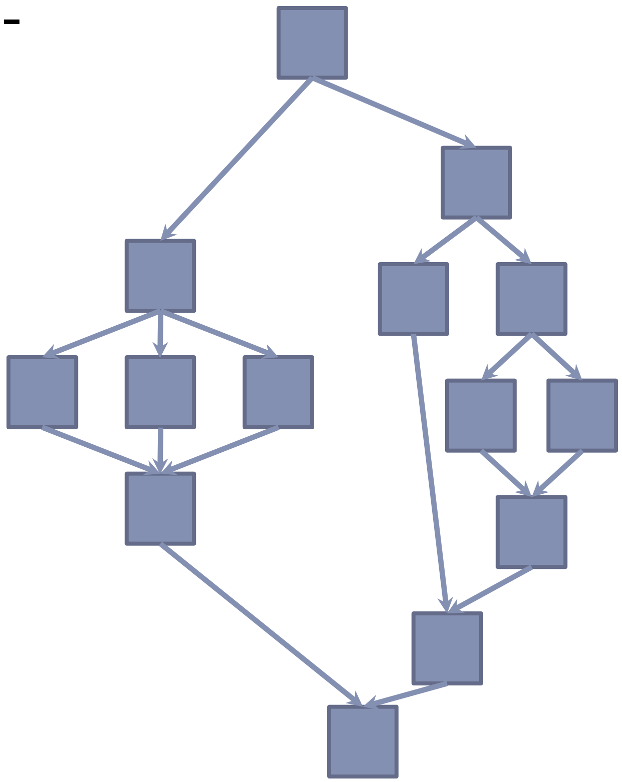
Machine 2

Program-Centric Cost Models

- Choose portable program description - dynamic Directed Acyclic Graph (DAG)

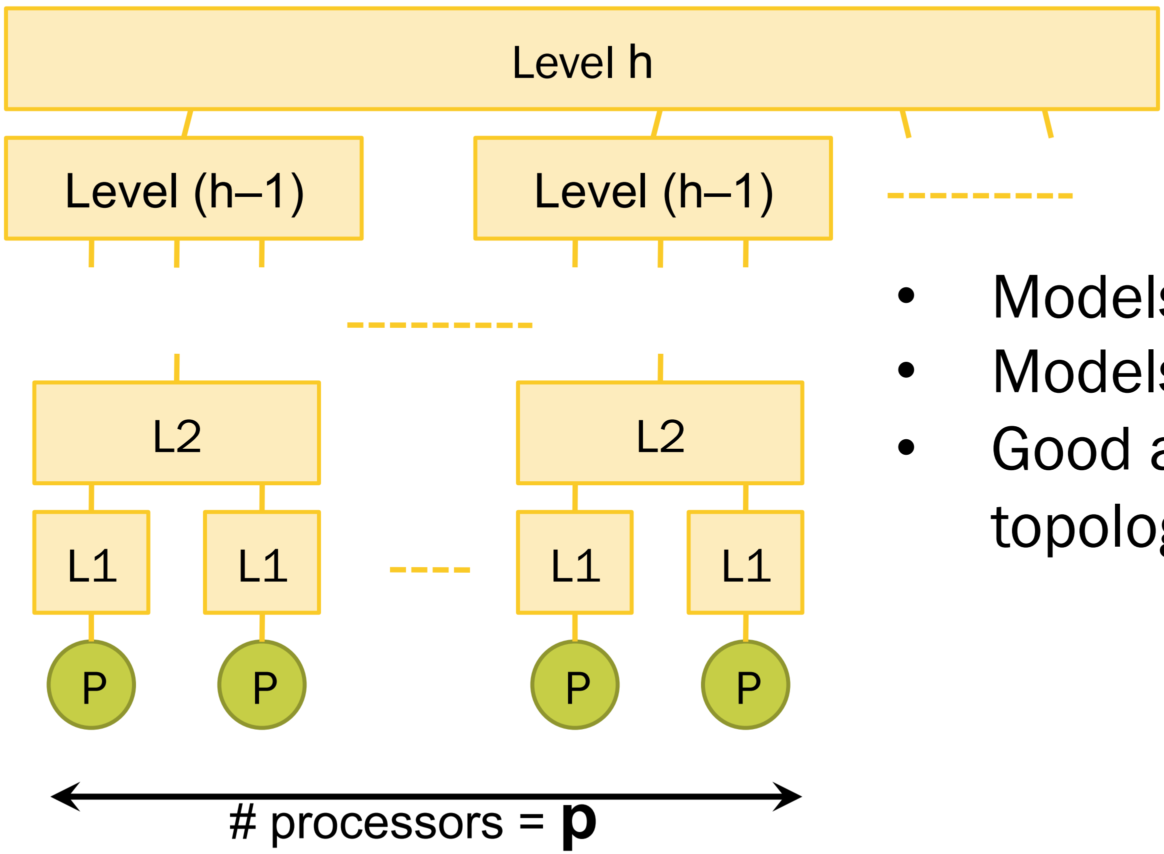
- Analyze DAG with out reference to processors, caches, connections…

- Examples of program-centric metrics
  - Number of operations (Work, W)
  - Length of Critical Path (Depth, D)
  - Data reuse patterns (Locality)



Do program-centric metrics say anything about performance of realistic machine models?

Realistic Machine Model: Tree of caches

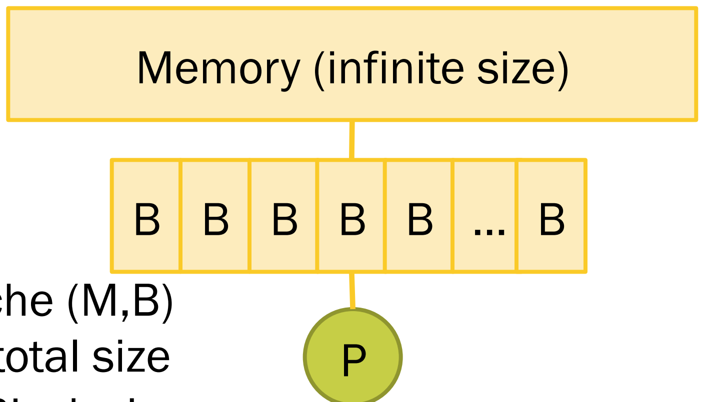


- Models hierarchical locality
- Models resource sharing
- Good approximation for other topologies

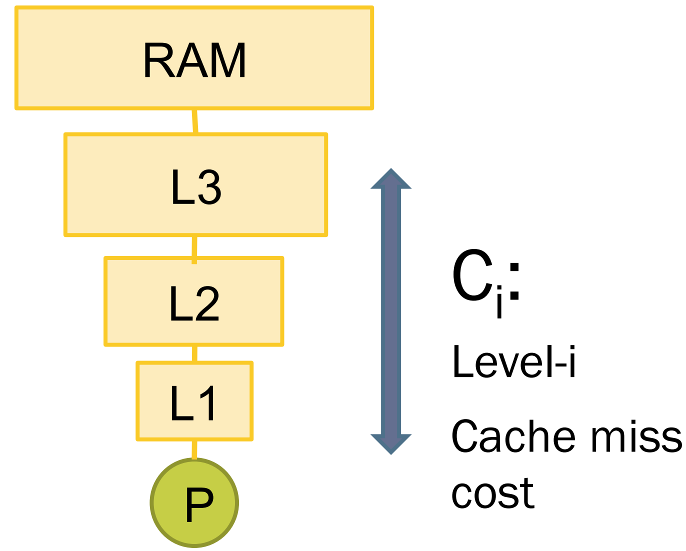
Program-Centric Cost Model for Sequential Algorithms

Cache Oblivious Framework [FLPR'99]

Sequential program



Cache (M,B)
M: total size
B: Block size



Locality: Cache complexity

$\mathbf{Q_1 (M,B)}$ **= #Cache Misses**

(don't use M,B in program)

Running time = $\sum_i Q_1(M_i,B) \times C_i$

Scan: $Q_1 (M,B) = O(n/B)$

Recursive MatMul: $Q_1 (M,B) = O(n^{1.5}/BM^{0.5})$

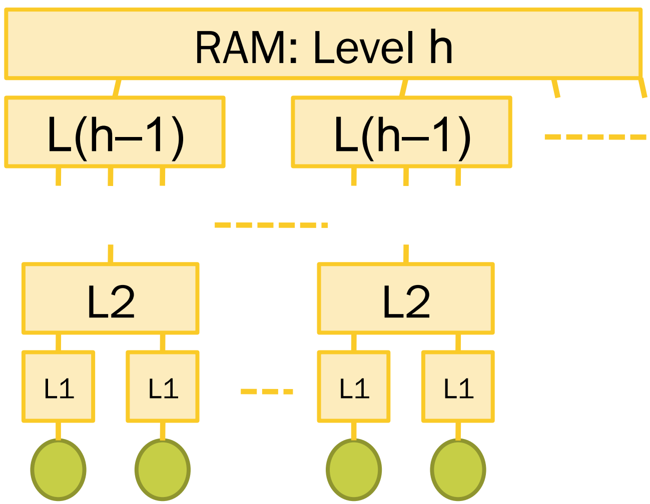
Distribution Sort: $Q_1 (M,B) = O(n/B \log_M n)$

Program-Centric Metrics for Parallel Programs

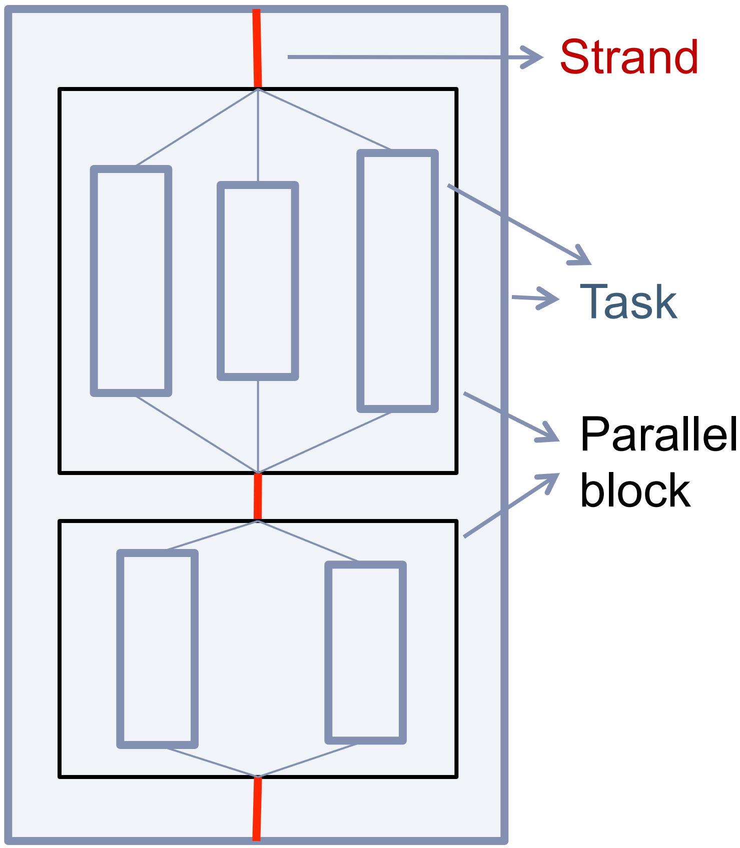
For nested parallel programs on shared memories.

- Parallel Cache Complexity Framework [BFGS'11]
  - Parallel Cache Complexity: Q*
    For locality
  - Effective Cache Complexity: $Q^*_{\circledR}$
    For locality + load balance cost
    Leads to definition of parallelizability

Efficiently mapped



Nested Parallel DAGs



a.k.a. Fork-Join Parallel DAGs

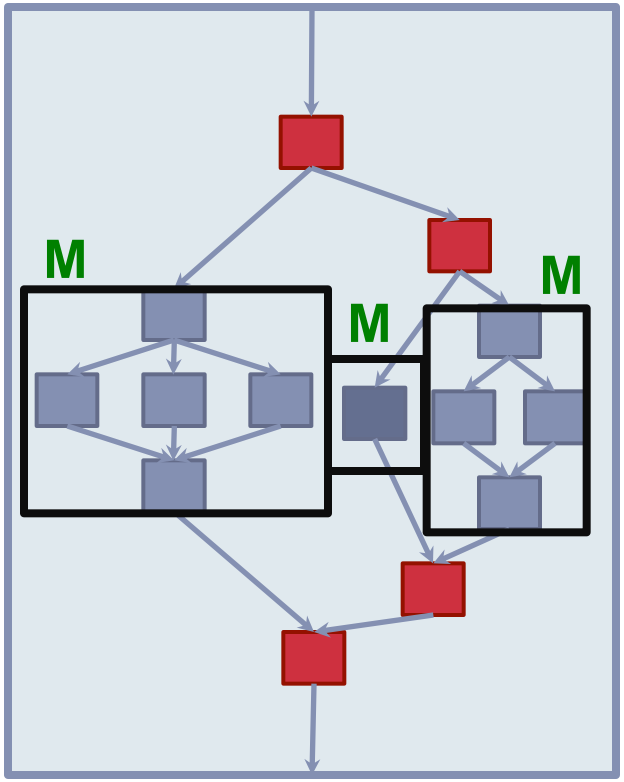
a.k.a Series-Parallel DAGs

Recursive definition :

- A task consists of alternating strands and parallel blocks.
- A strand is a sequential computation (chain of instructions).
- A **parallel block** is a parallel composition of tasks.

No data dependencies between Parallel subtasks

Locality: Parallel Cache Complexity



- Decompose task into maximal **subtasks** that fit in space **M** and glue operations.
  - Decomposition unique and easy to find for nested-parallel DAGs
- Parallel Cache Complexity:

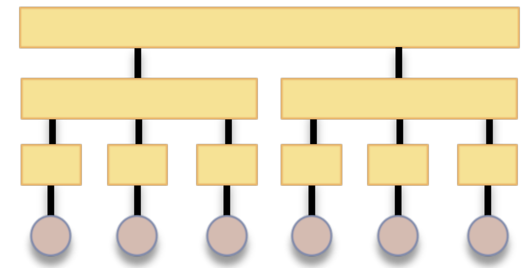
  Q*(M,B) =

  Σ Space for M-fitting subtasks

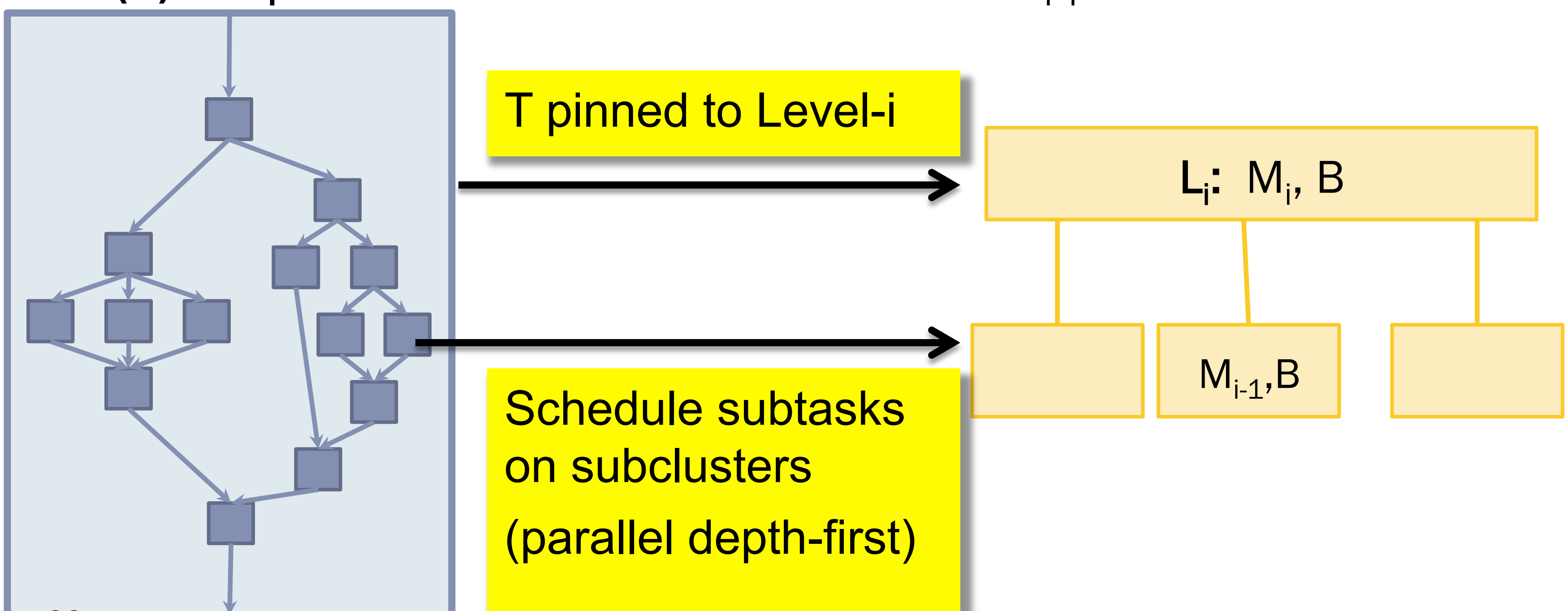
  + Σ Cache miss for every access in glue

Scheduling on Tree of Caches

- Annotate tasks with size, schedule based on size

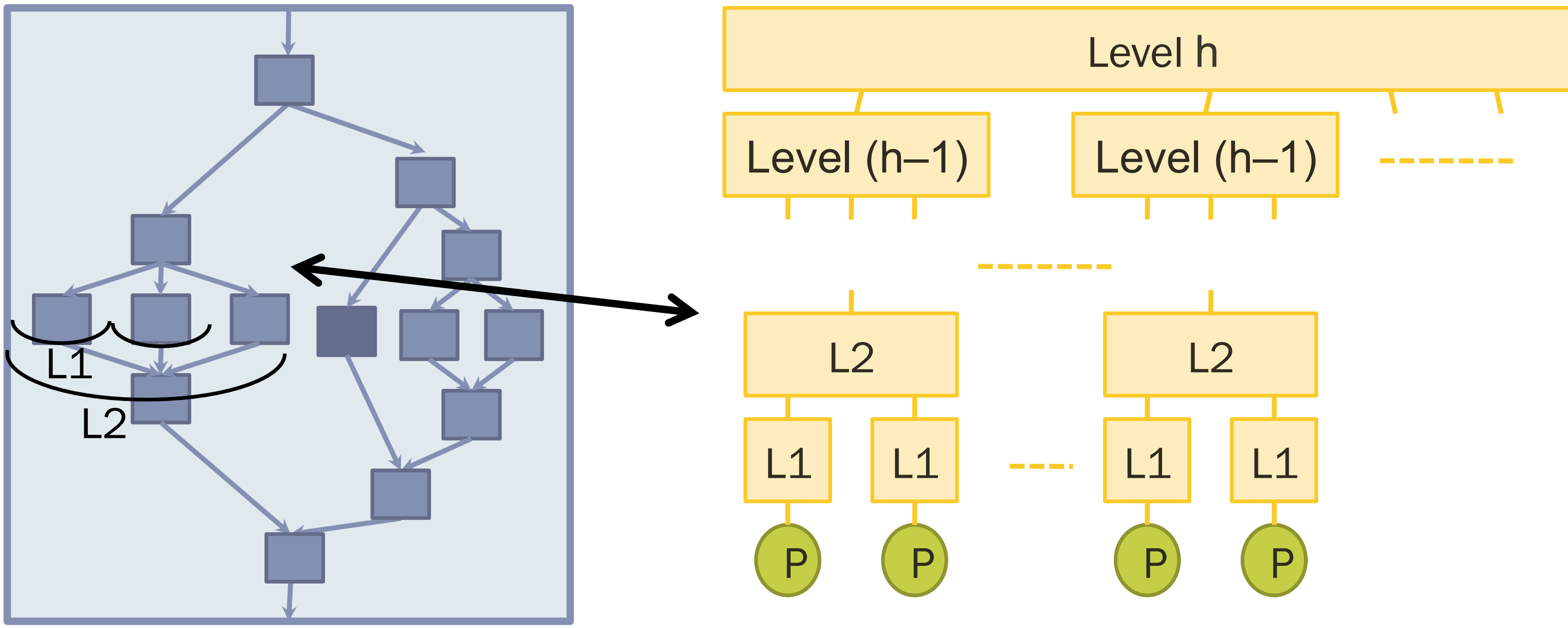


Size(T) ~ M_i : Unroll until subtasks have size ~M_{i-1}



Scheduling: Cost Model to Machines

Nested wavefronts based on hierarchy



Scheduler specifies Size, number and location of wavefronts based on working set size of tasks

Schedule preserves locality

- Communication Costs:

  Cache misses at level-i <= **Q*(M$_i$, B)**

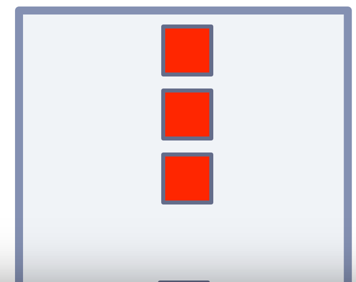
- Time?
  - # Processors assigned to a task tied to its space
  - Schedulers are good if computation is balanced:

    Work, Parallelism related to space

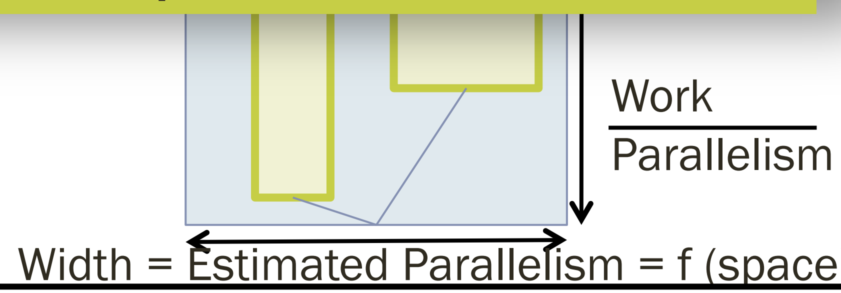
Effective Cache Complexity [BFGS'11]

Extend Q* to include cost of "imbalance"

- Long strands (Amdahl's law)



Definition based solely on Q* and the composition rules of nested-paralel DAGs



Given n® processors (input size: n),

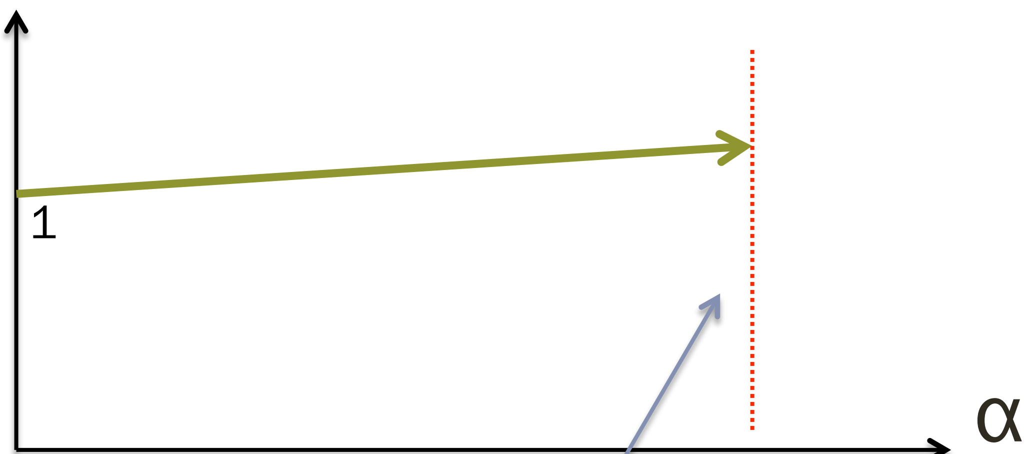
Q*®(n; M,B) = Q*(n; M,B) + the extra cost of imbalance

Parallelizability of algorithm

If algorithm has $O(n^w)$ work, $O(n^d)$ depth, imbalanced "only in parts"

$Q^*_{®} = O(Q^*)$ if and only if ® < **w-d**

$$Lt_{n \to \infty} \frac{Q^*_\alpha(n;\, M,B)}{Q^*(n;\, M,B)}$$



Parallelizability of an algorithm,

Samplesort: 1;   Recursive MatMul: 1.5

Effective cache complexity and parallelizability subsume all previous metrics: W,D,Q,….

Mapping Cost model to Communication Cost, Time & Space [S.-Thesis'13]

The scheduler preserves locality (matches $Q^*_{®}$) and is good at load balancing

- Communication Cost: $Q^*(M_i,B)$
- For most "reasonable" algorithms, the asymptotic running time is

$$\frac{\sum_{i=1}^{h} Q^*(M_i,B) \times C_i}{p}$$

- Also space Bounds

Low-Depth Cache-Oblivious Algorithms [BGS'10]

Low depth + good Parallel Cache Complexity Q*

=

good Effective Cache Complexity $Q^*_{®}$

Problem	Parallelizability	Parallel Cache Complexity
Prefix Sums	1	O(n/B)
Merge	1	O(n/B)
Sort (deterministic)	1	$O(n/B \log_M n)$
Sort (randomized; bounds are w.h.p.)	1	$O(n/B \log_M n)$
Sparse Matrix X Vector (m entries, n^k separators)	<1	$O(m/B + n/M^{1-k})$
Matrix transpose (n X m size)	1	O(nm/B)

Parallel overdesign (*polylog* depth, ®→1) improves performance

Algorithms scale really well in practice, even on tree of caches!

Low-Depth Cache-Oblivious Algorithms [BG**S**'10]

Graph Algorithms

Problem	Parallelizability	Parallel Cache Complexity
List Ranking	1	$O(Q_{sort}(n))$
Euler Tour on Trees	1	$O(Q_{sort}(n))$
Tree Contraction	1	$O(Q_{sort}(n))$
Least Common Ancestors (k queries)	1	$O((k/n)Q_{sort}(n))$
Connected Components	1	$O(Q_{sort}(\|E\|) \log(\|V\|/M^{1/2}))$
Minimum Spanning Forest	1	$O(Q_{sort}(\|E\|) \log(\|V\|/M^{1/2}))$

Combinatorial

Problem	Parallelizability	Parallel Cache Complexity
Set Cover (1+ε)-log n approx [BST'12]	1	$O(Q_{sort}(n))$

Summary

- Important to quantify locality and parallelism.

- Program-centric models are more portable
  - Scheduler design a separate problem

- Parallel Cache Complexity Framework [S.-thesis'13]
  - Translatable to performance on realistic machine models
  - Optimal algorithms can be designed
  - Theory works well in practice

- Locality and parallelism design translates across models

Questions?