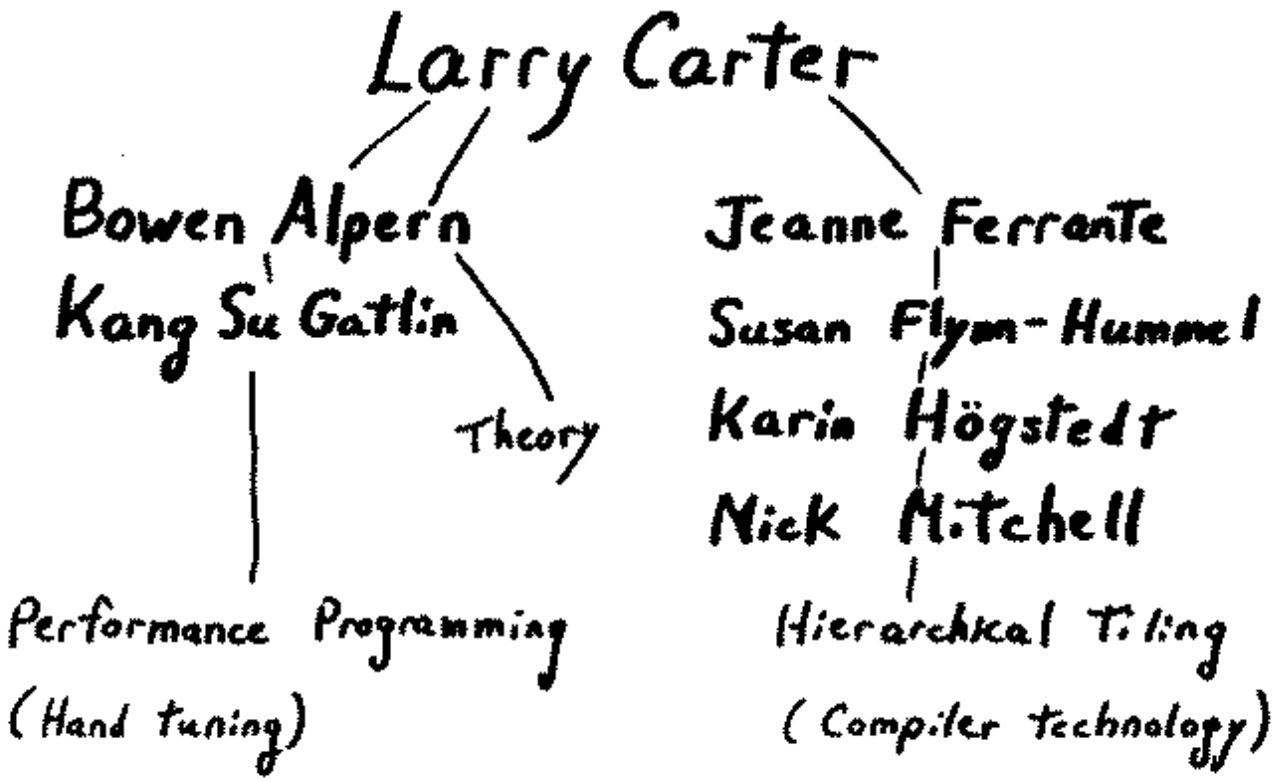
Yet Another Model of Computation: the Parallel Memory Hierarchy (PMH)



The Parallelism Hierarchy

- Microparallelism: partition words into fields
- Instruction level parallelism
  - Superpiplined functional units
    - Overlapped computation
  - Superscalar processors
    - Multiple functional units
- Symmetric multiprocessors (SMP)
  - Architectually supported shared memory
- Distributed memory multicomputer (MPP)
  - High performance switch
- Networked computers
  - Possibly heterogeneous

Possible Concurrency

The following can all occur simultaneously:

ALU transfers (arithmetic operations)

Cache to register transfers (loads and stores)

Memory to cache transfers

- prefetches and poststores
- out-of-order loads and stores

DMA transfers (I/O adaptors to memory)

- time-sharing the memory bus

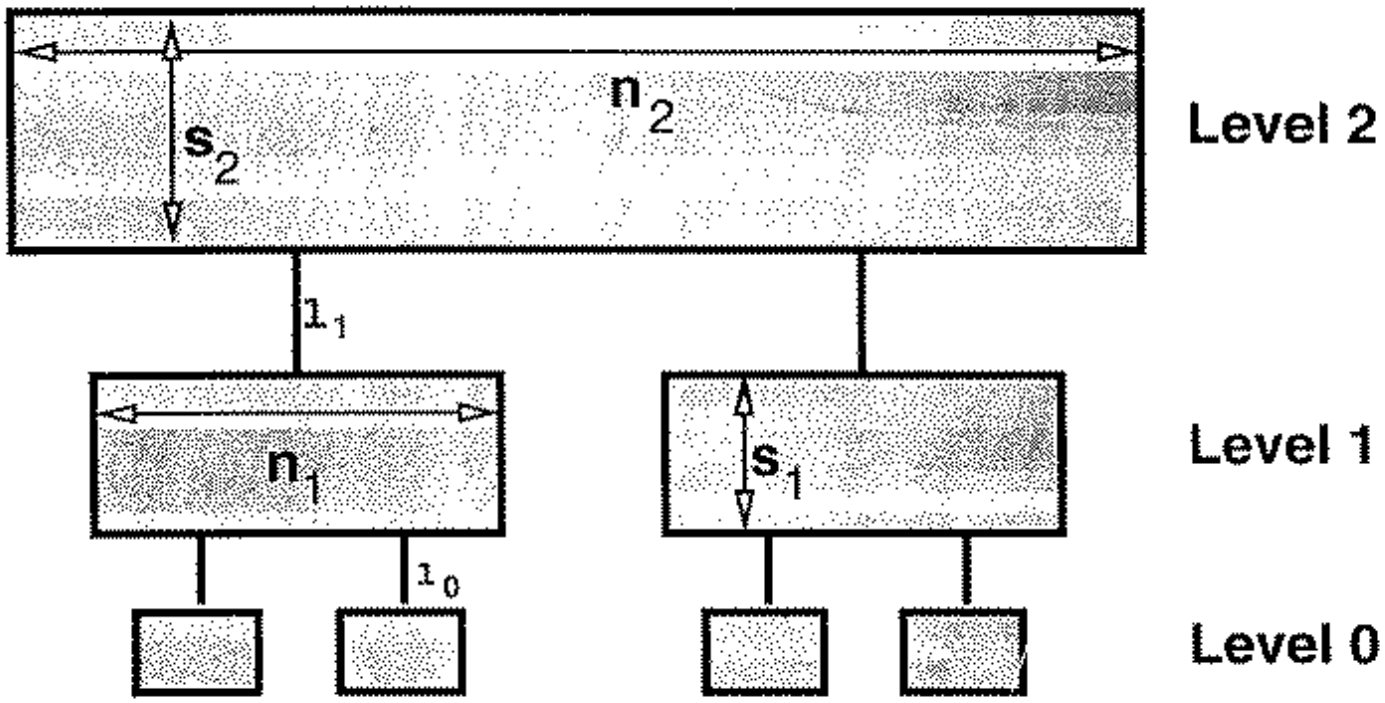
Interprocessor communication

- asynchronous messages

Disk controller to SCSI adaptor transfers

Disk access (platter to disk controller)

PMH Model



Tree of "modules" (RAM with limited memory)

Connected by "channels" (data moves in fixed-size blocks)

Smaller, faster modules towards leaves

Parameterized model: each module has

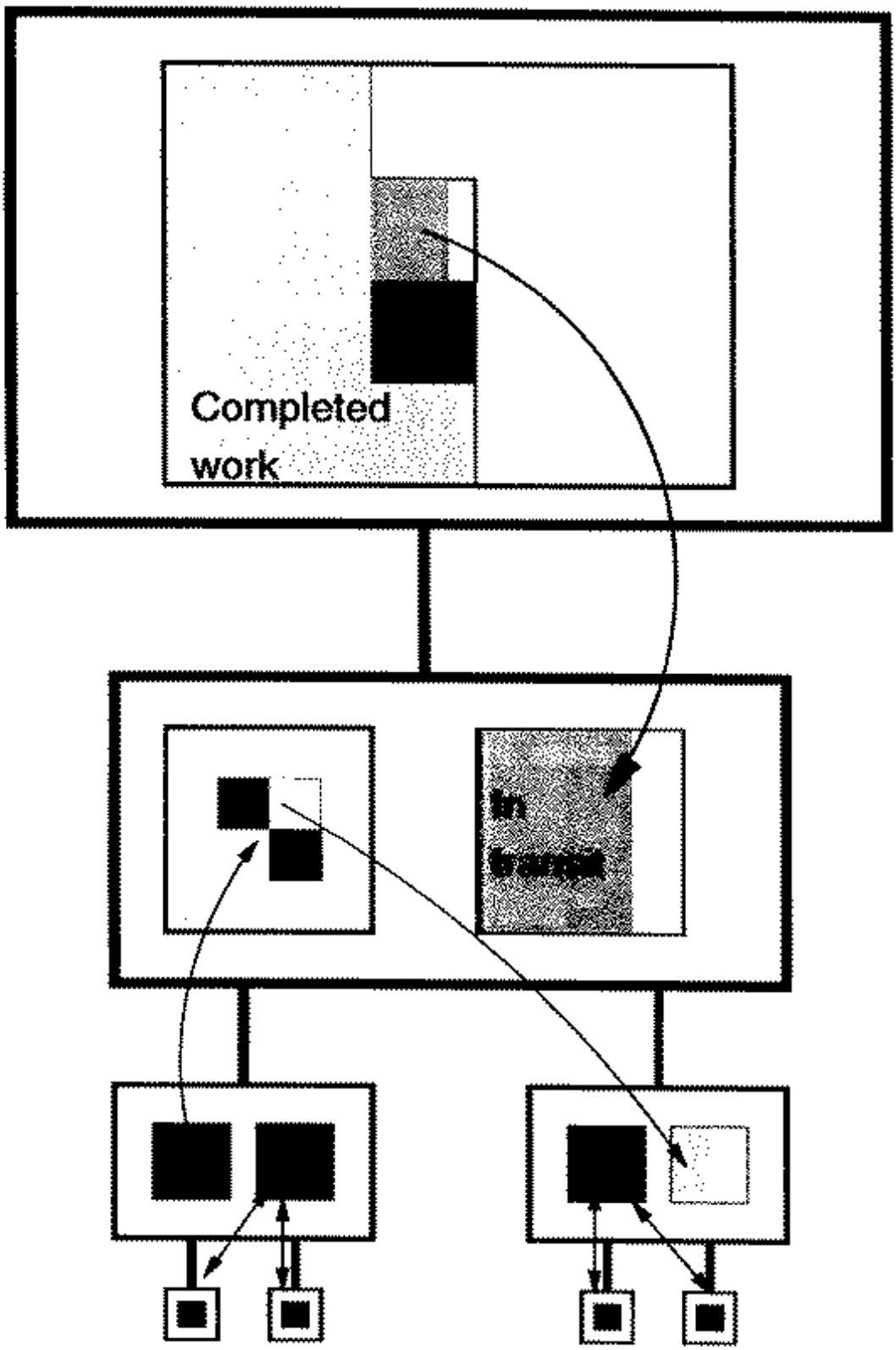
block count n_i

block size s_i

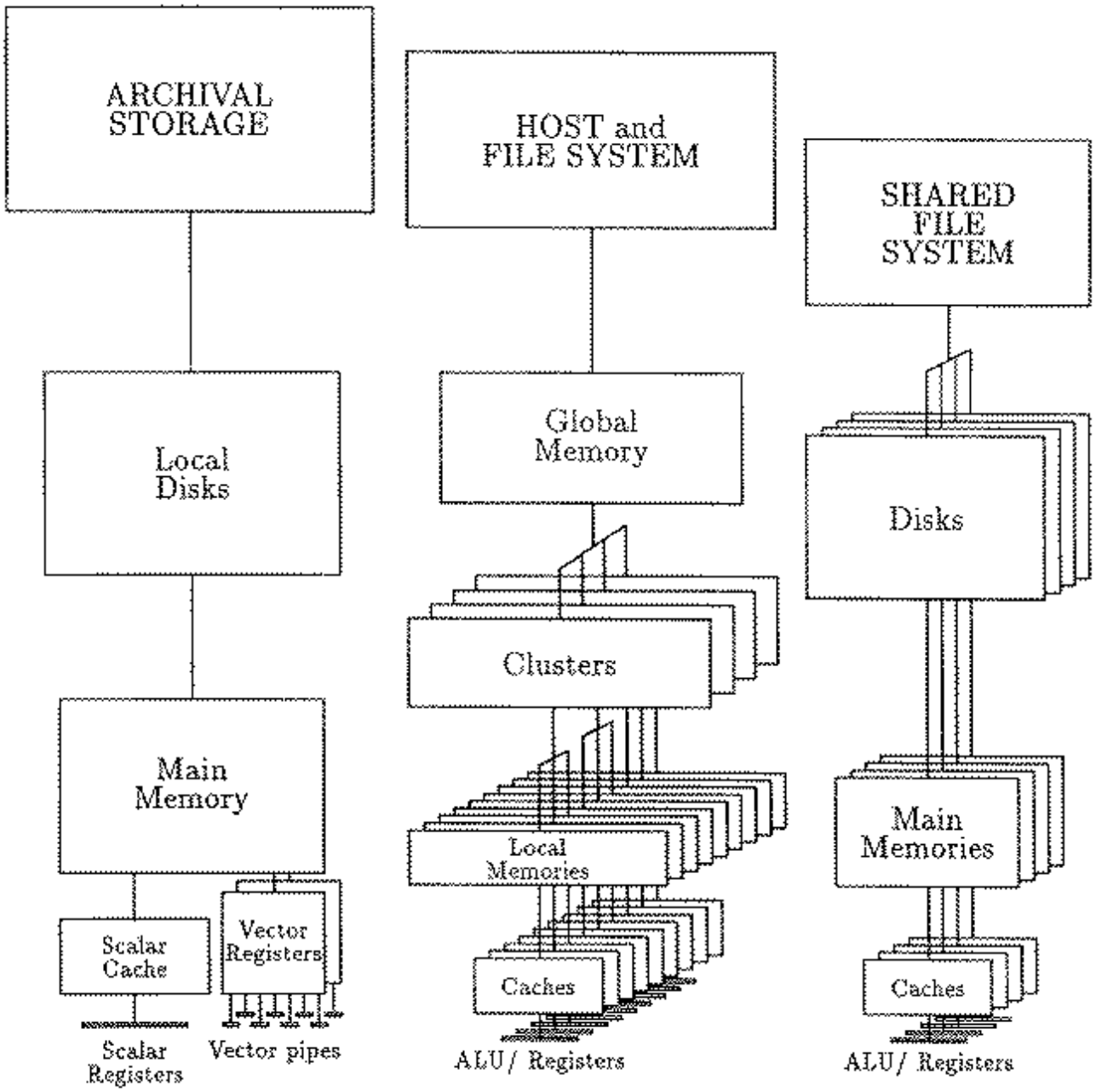
transfer time l_i - time to move a block to parent

child count

(speed) (every module except leaves: slow)


Completed
work

Some Parallel Architectures



PMH Analysis

"Communication-efficient" algorithm : $\frac{\text{PMH time}}{\text{PRAM time}} < c$

Thm If (blah blah) then { rational permutations, matrix multiply, FFT } are communication-efficient.

("blah-blah" says mainly that $n_i > s_i$ and bandwidths are sufficiently high to move data <u>once</u> and there aren't too many processors)

Thm Quicksort is better than ~~merge~~ heap sort

I.e. it requires less bandwidth

Pf Merge sort makes random accesses

Quicksort has better locality

Thm (?) Recursion is better than iteration

Thm (?) Call-by-value is better than call-by-reference.

Why PMH may Remain Valid

- Fast memory is expensive
- Very fast memory must be small
- Latency is unavoidable
- Block moves increase banduidth
- Control logic is cheap

Architectural Trends

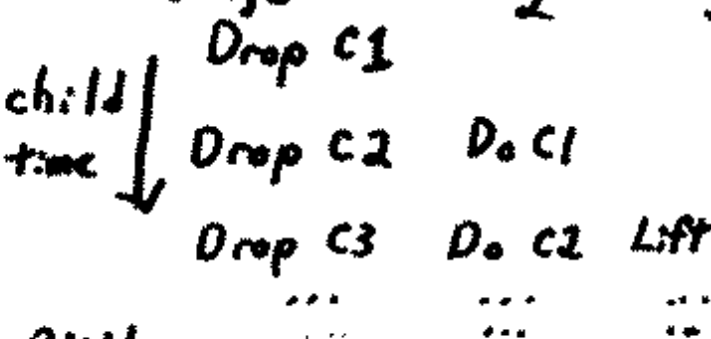
- More levels of memory hierarchy (L2, L3 caches)
- More levels of parallelism (MMX, coherent caches, NOW's, Metacomputing)
- Heterogeneity (Partial upgrades, NOWs, Metacomputing, contention)

How To Ignore the Trends :

- Never calculate useful operations/cycle
- Report relative speedup
- Simulate on uniform model
- Find (or build) machine with slow processors, fast ne
- Never upgrade nodes
- Run small problems
- Ignore superlinear speedup
- Trust the compiler

Programming a PMH (a la hierarchical tili

- Problem starts in smallest module that it fits
- Break problem into "chores". Give each child a sequence of chores.
  - chores being executed concurrently must be independe
  - surface sharing
  - 3-stage pipeline per child



  - chore size depends on PMH parameters
  - skeletons may help coordinate !
- Continue recursively

Hierarchical Tiling's goal is to automate writing PMH programs.

Compiler's back end translates to Clevel-dependent code.

Principle of Portable Performance

For near-peak performance, different computers will run different sequences of source-language statements.

Example: `DGEMV` (matrix-vector product)

- Scalar processors: `DDOT` based
  - Fewer `stores`
- Vector processors: `DAXPY` based
  - Independent `fmas`
- Superscalar processors: hybrid based
  - Some of both

How this is accomplished?

- Tuned libraries (LAPACK, ScaLAPACK, etc.)
- Optimizing compiler (FORTRAN90, HPF, etc.)
- Ad-hoc compiler directives and options
- Explicit program variants

Space-Limited Procedures

Recursive procedures

- Recursive calls **must** use less space
  - Promotes locality

Ambiguous argument passing semantics

- Even for arrays!
- call-by-reference
  - Allows aggressive inlining (within a memory module)
- call-by-value
  - Allows explicit data movement (between memory modules)

Procedure name overloading

- Interchangable *versions*

Explicit tuning parameters

- *Machine parameters* of the PMH model
- *Problem parameters* describe problem instances
- *Free parameters* are deferred tuning choices

Explicit parallelism

PMH vs BSP

BSP : Easier to program
Tools exist today

PMH : Solves whole performance programming problem
Better performance
Accomodates heterogeneity
Computing on reconfigurable Metacomputer