

*Invited Presentation to SIAM 2006 Mini-Symposium  
(MS49) :*

# How Can Computer Architecture Revolutionize Parallel Scientific Computing

**Thomas Sterling**  
Louisiana State University  
and  
California Institute of Technology

February 24, 2006



AT LOUISIANA STATE UNIVERSITY



# Challenges to Petascale Scientific Computing that architecture can help

- Bringing orders of magnitude greater sustained performance and memory capacity to real-world scientific applications
  - Many problems are Exa(fl)ops scale or greater
- Exploiting ultra-massive parallelism at all levels
  - Either automatic discovery or ease of representation
  - Hardware use for reduced time or latency hiding
- Breaking the “barrier”
  - Moving away from global barrier synchronization
  - Over constrained
- Removing burden of explicit manual resource allocation
  - Locality management
  - Load balancing
- Memory wall
  - Accelerating memory intense problems
  - Hardware structures for latency tolerance
  - Enabling efficient sparse irregular data structures manipulation
- Greater availability and cheaper machines



# Challenges to Computer Architecture

- Expose and exploit extreme fine-grain parallelism
  - Possibly multi-billion-way
  - Data structure-driven
- State storage takes up much more space than logic
  - 1:1 flops/byte ratio infeasible
  - Memory access bandwidth is the critical resource
- Latency
  - can approach a hundred thousand cycles
  - All actions are local
  - Contention due to inadequate bandwidth
- Overhead for fine grain parallelism must be very small
  - or system can not scale
  - One consequence is that global barrier synchronization is untenable
- Reliability
  - Very high replication of elements
  - Uncertain fault distribution
  - Fault tolerance essential for good yield
- Design complexity
  - Impacts development time, testing, power, and reliability



## Metric of Physical Locality, $\tau$

- Locality of operation dependent on amount of logic and state that can be accessed round-trip within a single clock cycle
- Define  $\tau$  as ratio of number of elements (e.g., gates, transistors) per chip to the number of elements accessible within a single clock cycle
- Not just a speed of light issue
- Also involves propagation through sequence of elements
- When I was an undergrad,  $\tau = 1$
- Today,  $\tau > 30$
- For SFQ at 100 GHz,  $100 < \tau < 1000$
- At nano-scale,  $\tau > 100,000$



## A Sustainable Strategy for Long Term Investment in Technology Trends

- Message-driven split-transaction parallel computing
  - Alternative parallel computing model
  - Parallel programming languages
  - Adaptable ultra lightweight runtime kernel
  - Hardware co-processors that manage threads and “active messages”
- Memory accelerator
  - Exploits embedded DRAM technology
  - Lightweight *MIND* data processing accelerator co-processors
- Heterogeneous System Architecture
  - Very high speed numeric processor for high temporal locality
  - Plus eco-system co-processor for parcel/multithreading
  - Plus memory accelerator chips
  - Data pre-staging

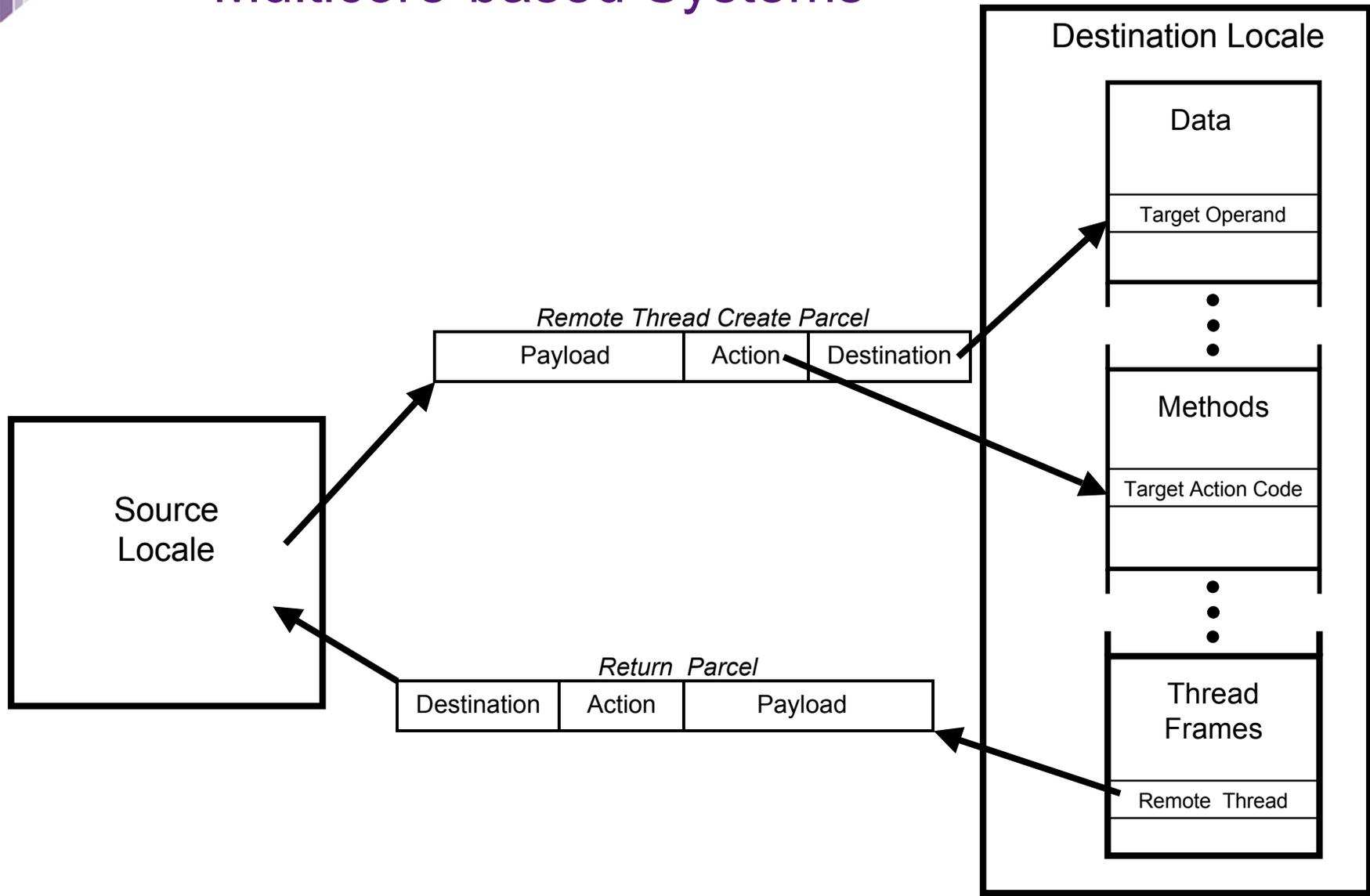


# Asynchronous Message Driven Split-Transaction Computing

- Powerful mechanism for hiding system-wide latency
- All operations are local
- Transactions performed on local data in response to incident messages (parcels)
- Results may include outgoing parcels
- No waiting for response to remote requests



# Parcels for Latency Hiding in Multicore-based Systems





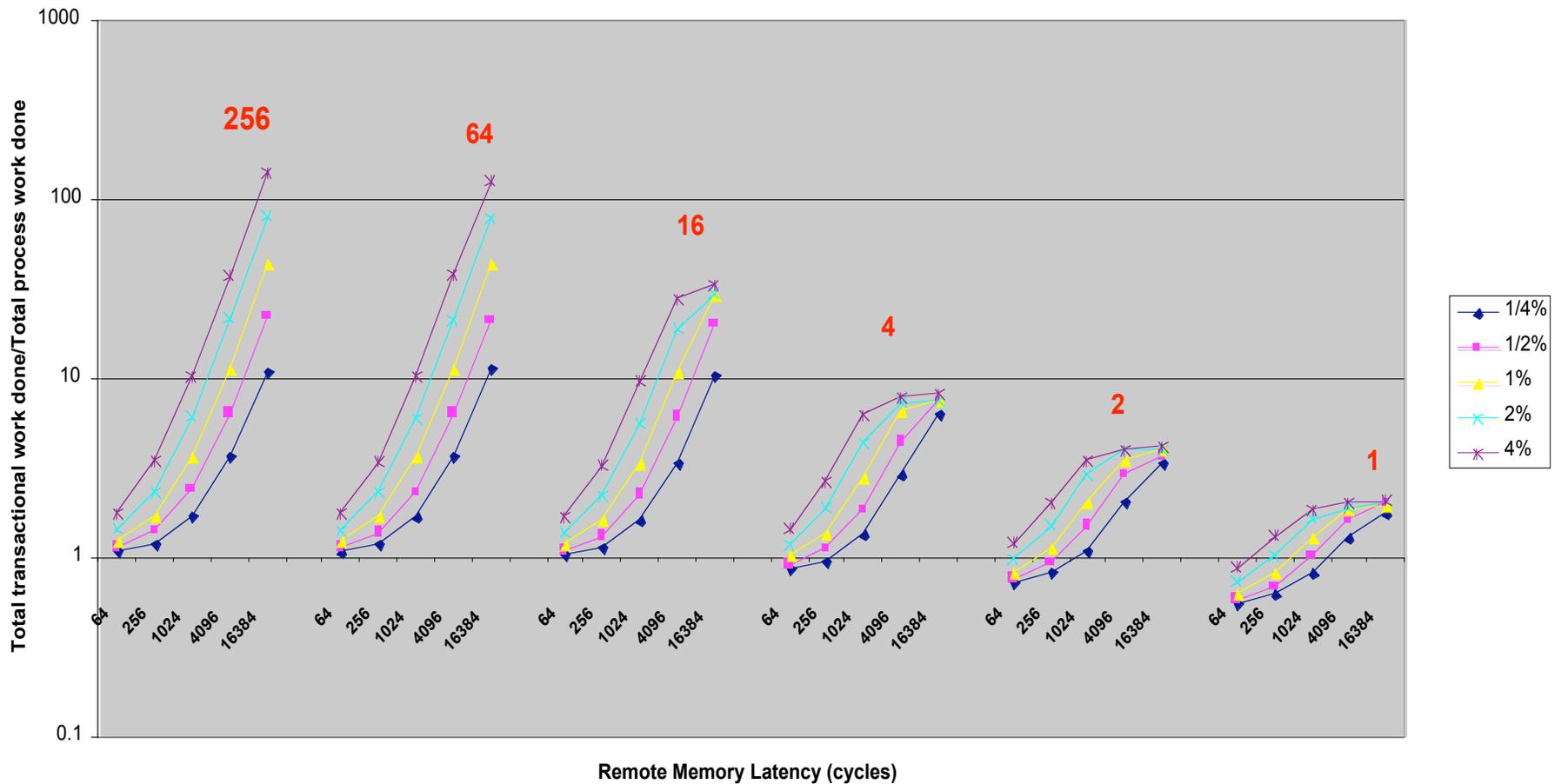
# Latency Hiding with Parcels

with respect to System Diameter in cycles

Sensitivity to Remote Latency and Remote Access Fraction

16 Nodes

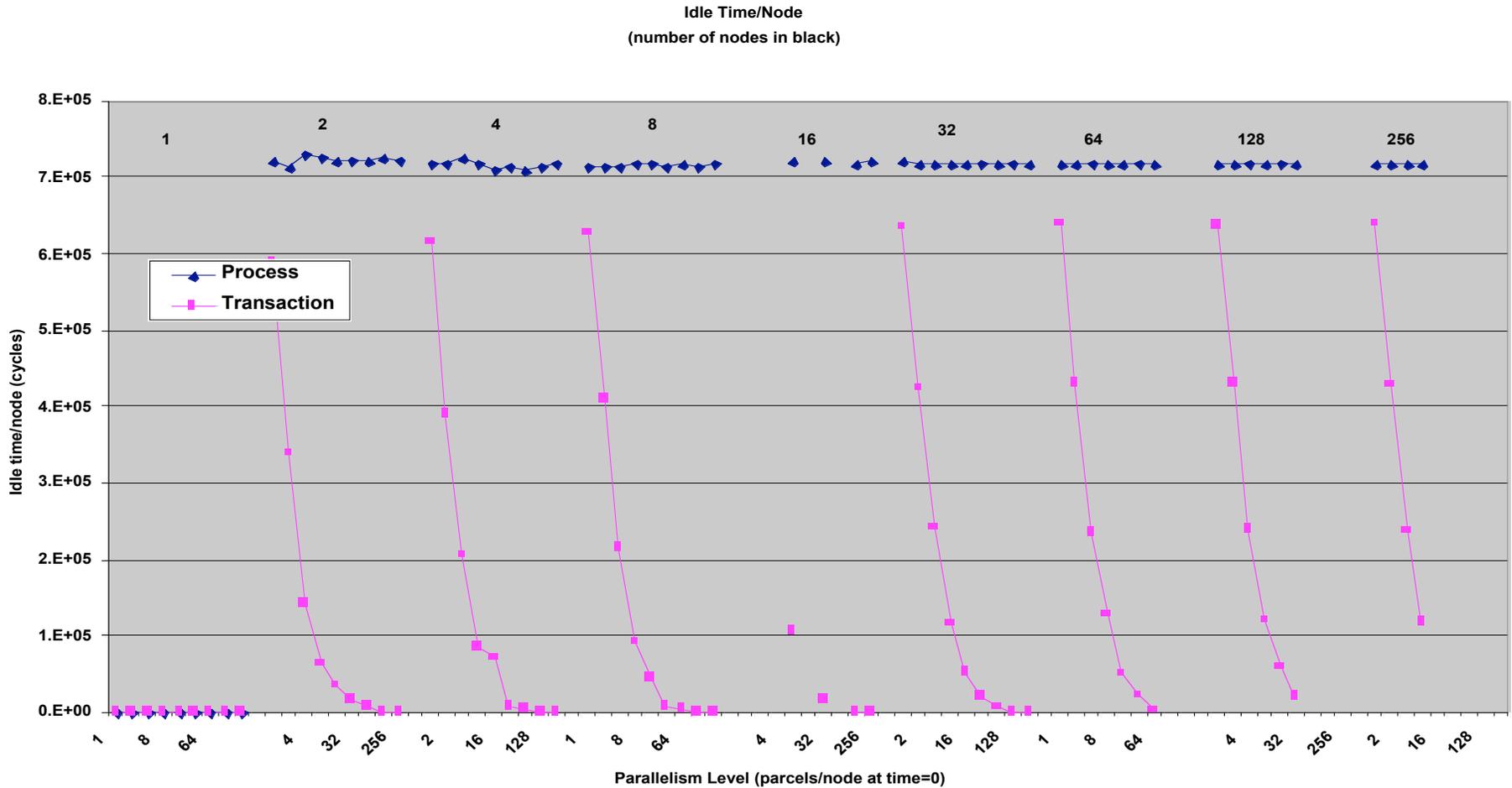
deg\_parallelism in RED (pending parcels @ t=0 per node)





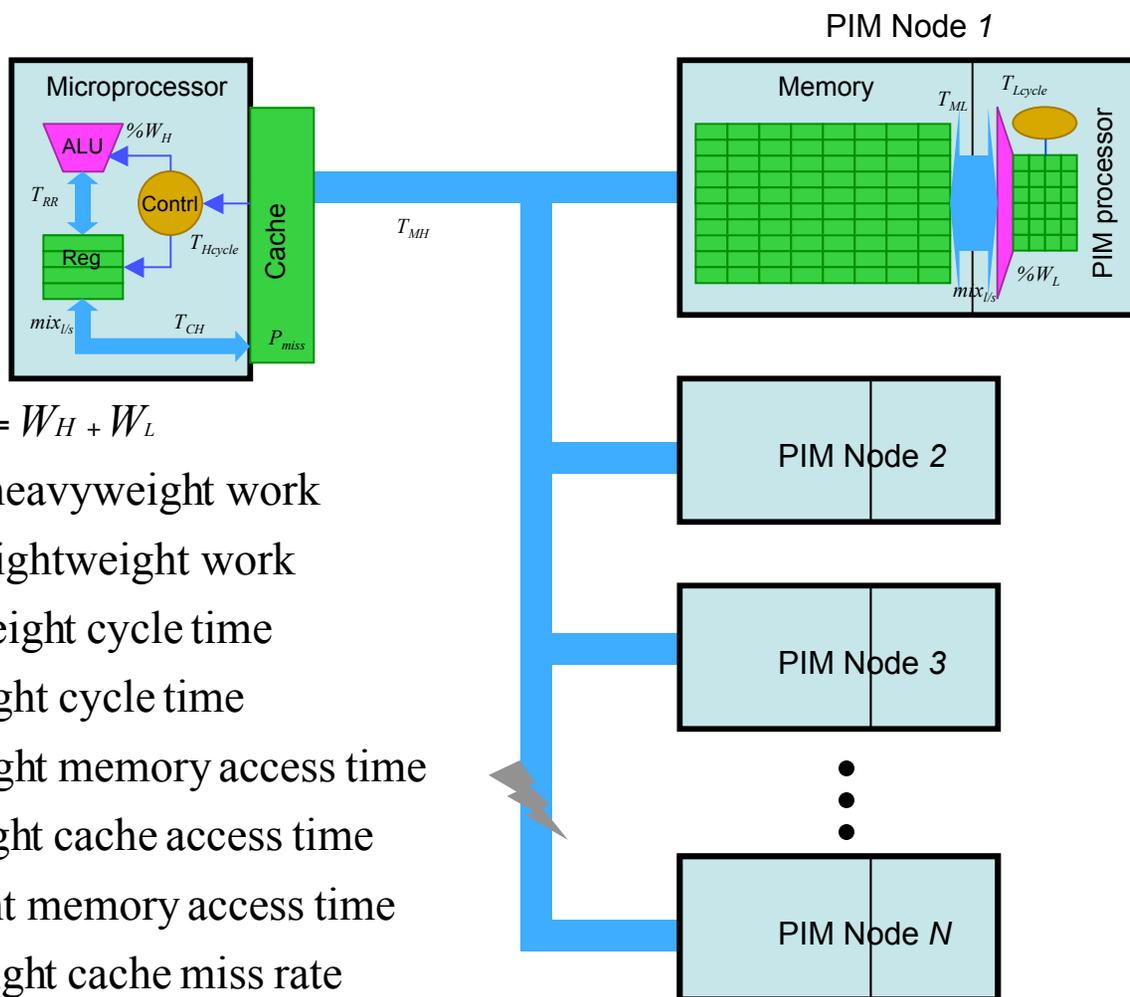
# Latency Hiding with Parcels

## Idle Time with respect to Degree of Parallelism





# Multi-Core Microprocessor with Memory Accelerator Co-processor



## Metrics

$W \equiv$  total work =  $W_H + W_L$

$\%W_H \equiv$  percent heavyweight work

$\%W_L \equiv$  percent lightweight work

$T_{Hcycle} \equiv$  heavyweight cycle time

$T_{Lcycle} \equiv$  lightweight cycle time

$T_{MH} \equiv$  heavyweight memory access time

$T_{CH} \equiv$  heavyweight cache access time

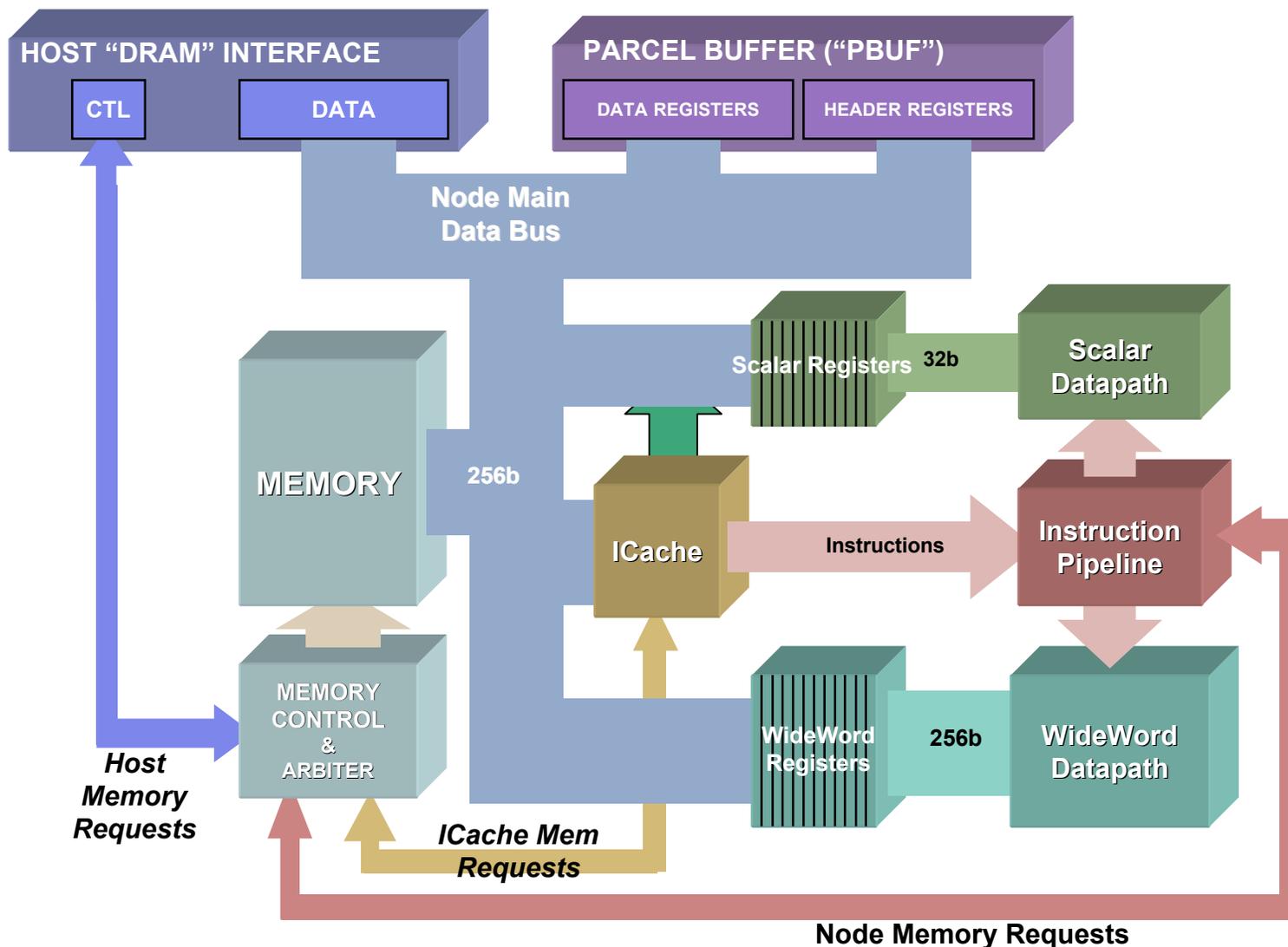
$T_{ML} \equiv$  lightweight memory access time

$P_{miss} \equiv$  heavyweight cache miss rate

$mix_{l/s} \equiv$  instruction mix for load and store ops

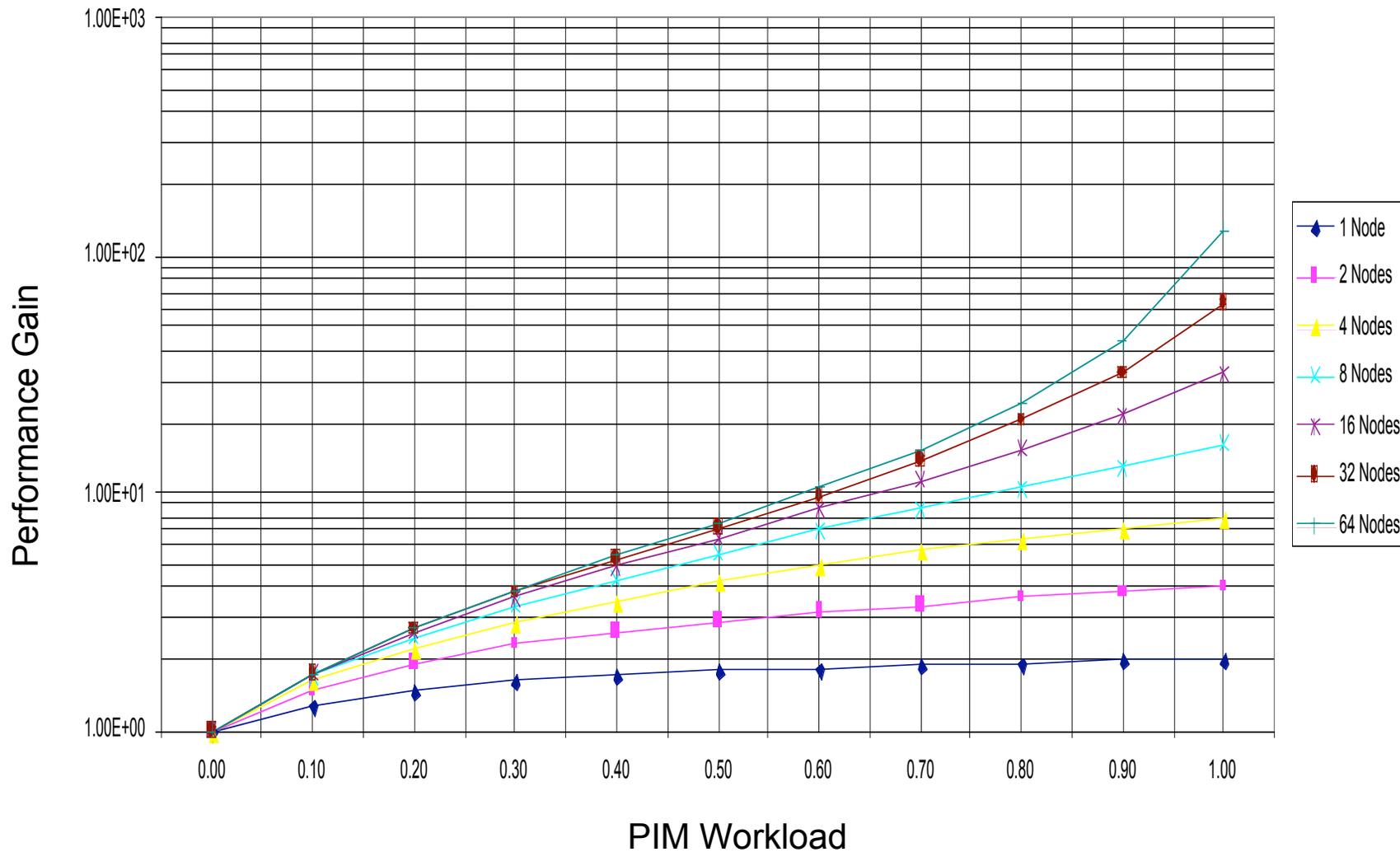


# DIVA USC ISI





# Simulation of Performance Gain



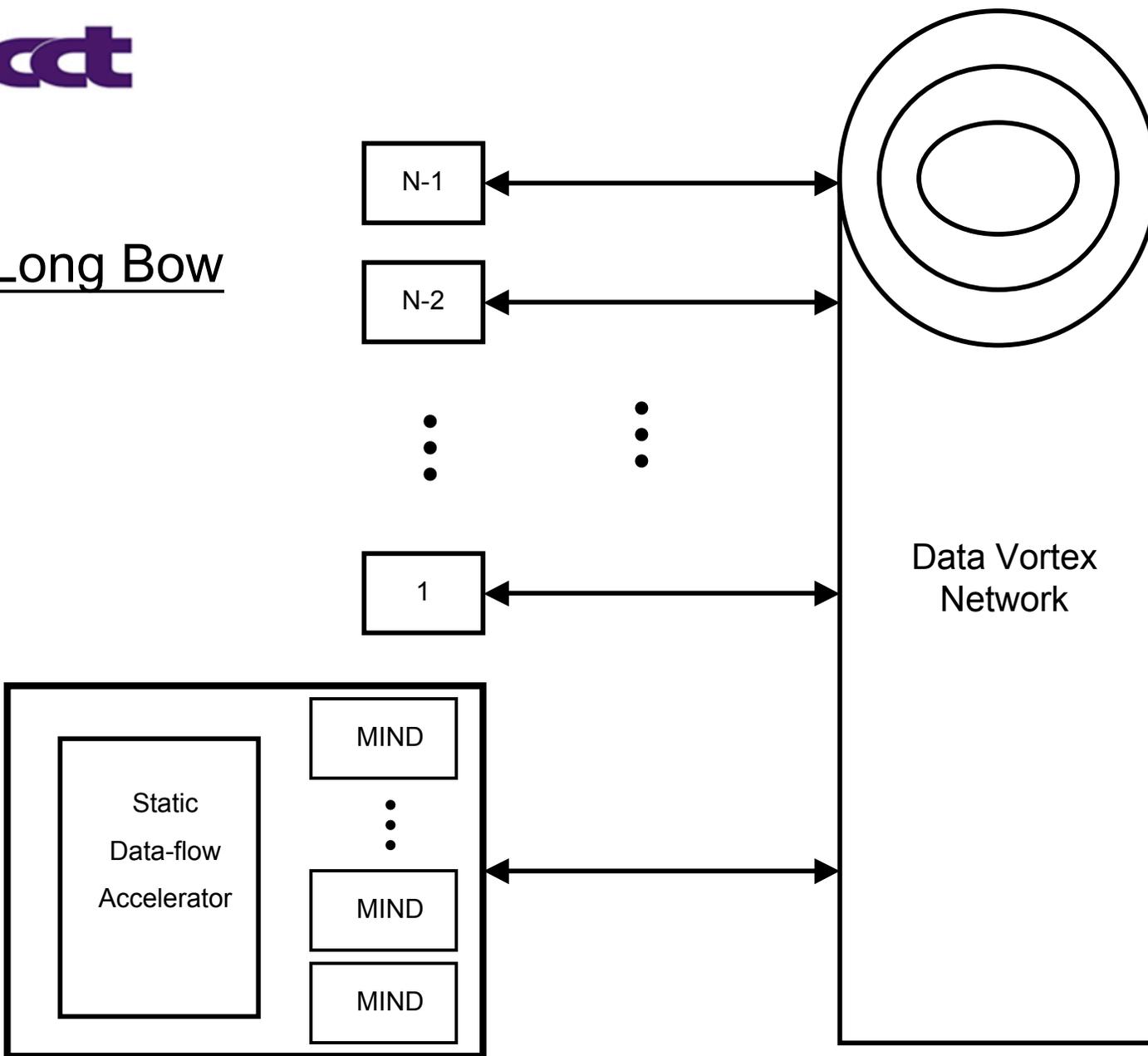


## ParalleX: A Latency Tolerant Parallel Computing Strategy

- split-transaction programming model (Dally, Culler)
- Distributed shared memory – not cache coherent (Scott, UPC)
- Embodies copy semantics in the form of location consistency (Gao)
- Message-driven (Hewitt)
- Multithreaded (Smith/Callahan)
- Futures synchronization (Halstead)
- Local Control Objects (e.g. dataflow synchronization)
- In-memory synchronization for producer-consumer



Long Bow





# Long Bow Architecture Attributes

- Latency Hiding Architecture
- Heterogeneous based on temporal locality
  - Low/no temporal locality high memory bandwidth logic
  - Hi temporal locality high clock rate ALU intensive structures
- Asynchronous
  - Remote message-driven
  - Local multithreaded, dataflow
  - Rich array of synchronization primitives including in-memory and in-register
- Global shared memory
  - Not cache coherence
  - Location consistency
- Percolation for prestaging
  - Flow control managed by MIND array
  - Processors are dumb, memory is smart
- Graceful Degradation
  - Isolation of replicated structures



# CCT High Speed Computing Element

- A kind of streaming architecture
  - Merrimac
  - Trips
- Employs array of ALUs in static data flow structure
- Temporary values passed directly between successive ALUs
- Data flow synchronization for asymmetric flow graphs
- Packet switched (rather than line switched)
- More general than pure SIMD
- Initialized (and take down) by MIND via percolation
- Multiple coarse-grain threads



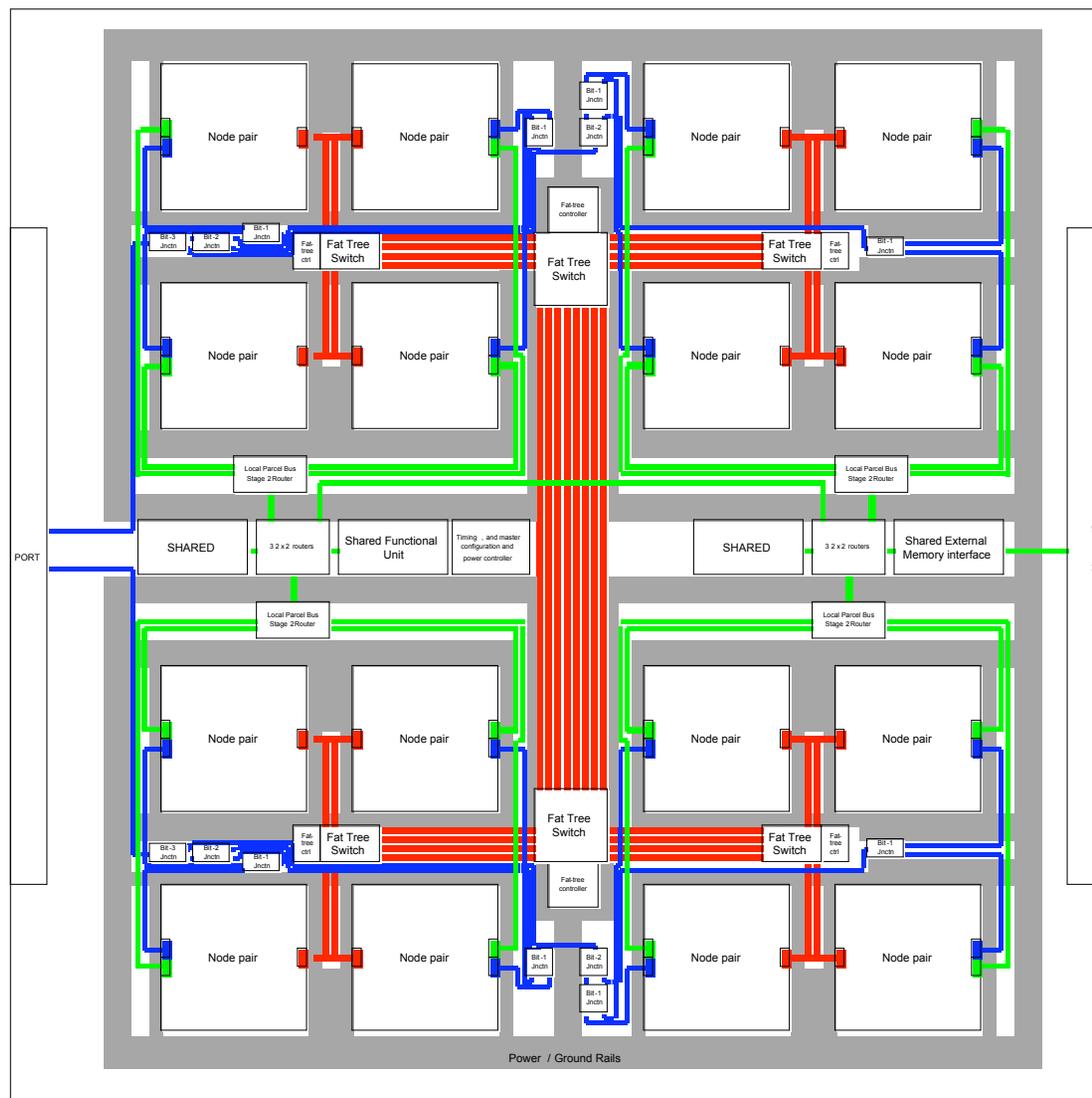
# Concepts of the MIND Architecture

- Virtual to physical address translation in memory
  - Global distributed shared memory thru distributed directory table
  - Dynamic page migration
  - Wide registers serve as context sensitive TLB
- Multithreaded control
  - Unified dynamic mechanism for resource management
  - Latency hiding
  - Real time response
- Parcel active message-driven computing
  - Decoupled split-transaction execution
  - System wide latency hiding
  - Move work to data instead of data to work
- Parallel atomic struct processing
  - Exploits direct access to wide rows of memory banks for fine grain parallelism and guarded compound operations
  - Exploits parallelism for better performance
  - Enables very efficient mechanisms for synchronization
- Fault tolerance through graceful degradation
- Active power management





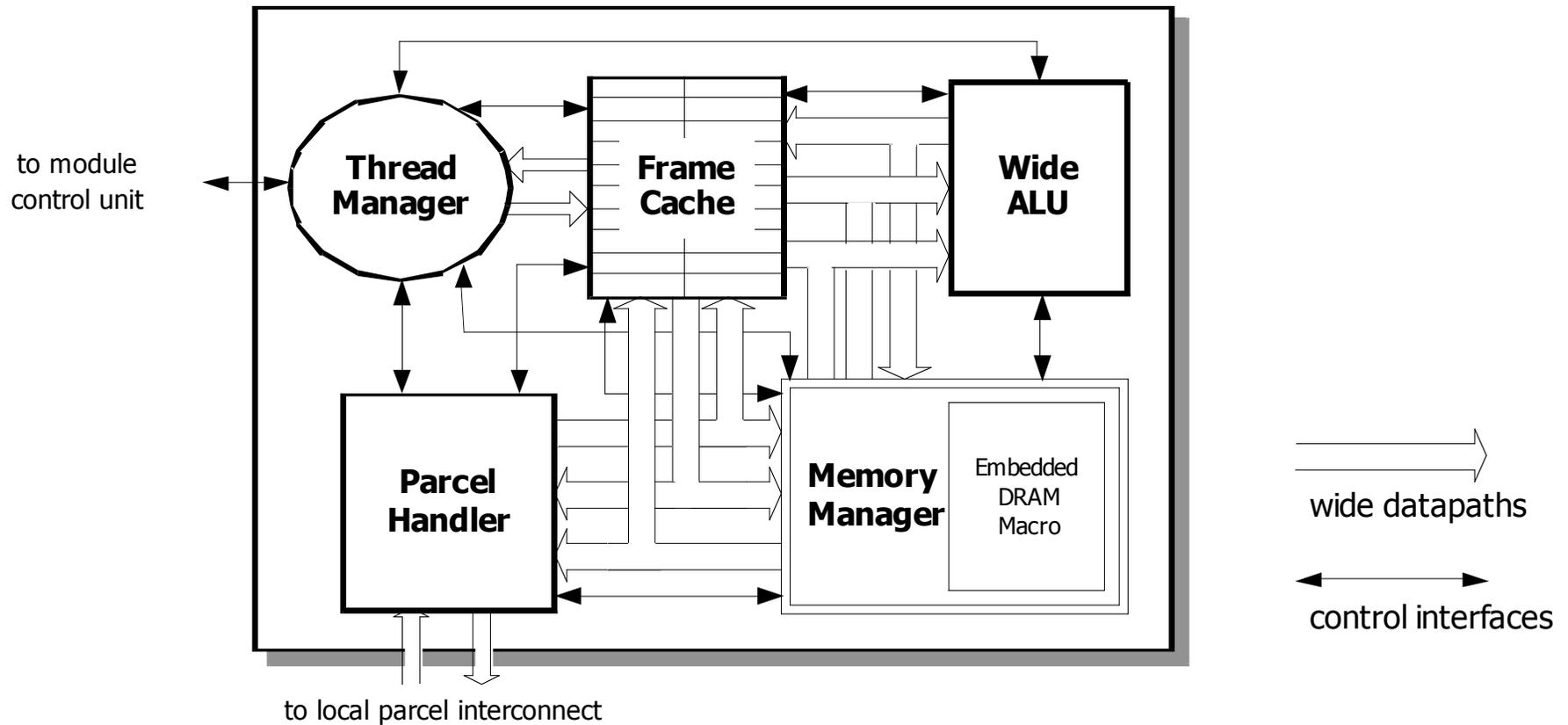
# Chip Layout





**cct**

# MIND Memory Accelerator





# Top 10 Challenges in HPC Architecture

*concepts, semantics, mechanisms, and structures*

- 10: Inter-chip high bandwidth data interconnect
- 9: Scalability through locality-oriented asynchronous control
- 8: Global name space translation including first-class processes for direct resource management
- 7: Multithreaded intra process flow control
- 6: Graceful degradation for reliability and high yield
- 5: Accelerators for lightweight-object synchronization including *futures* and other in-memory synchronization for mutual exclusion
- 4: Merger of data-links and go-to flow control semantics for directed graph traversal
- 3: In memory logic for memory accelerator for effective no-locality execution
- 2: Message-driven split-transaction processing
- 1: New execution model governing parallel computing



# Some Revolutionary Changes Won't Come from Architecture

- Debugging
  - Can be eased with hardware diagnostics
- Problem set up
  - e.g., mesh generation
- Data analysis
  - Science understanding
  - Computer visualization – an oxymoron
    - Computers don't know how to abstract knowledge from data
    - Once people visualize, the insight can't be computerized
- Problem specification – easier to program
  - Better architectures are will be easier to program
  - But problem representation can be intrinsically hard
- New models for advanced phenomenology
  - New algorithms
  - New physics