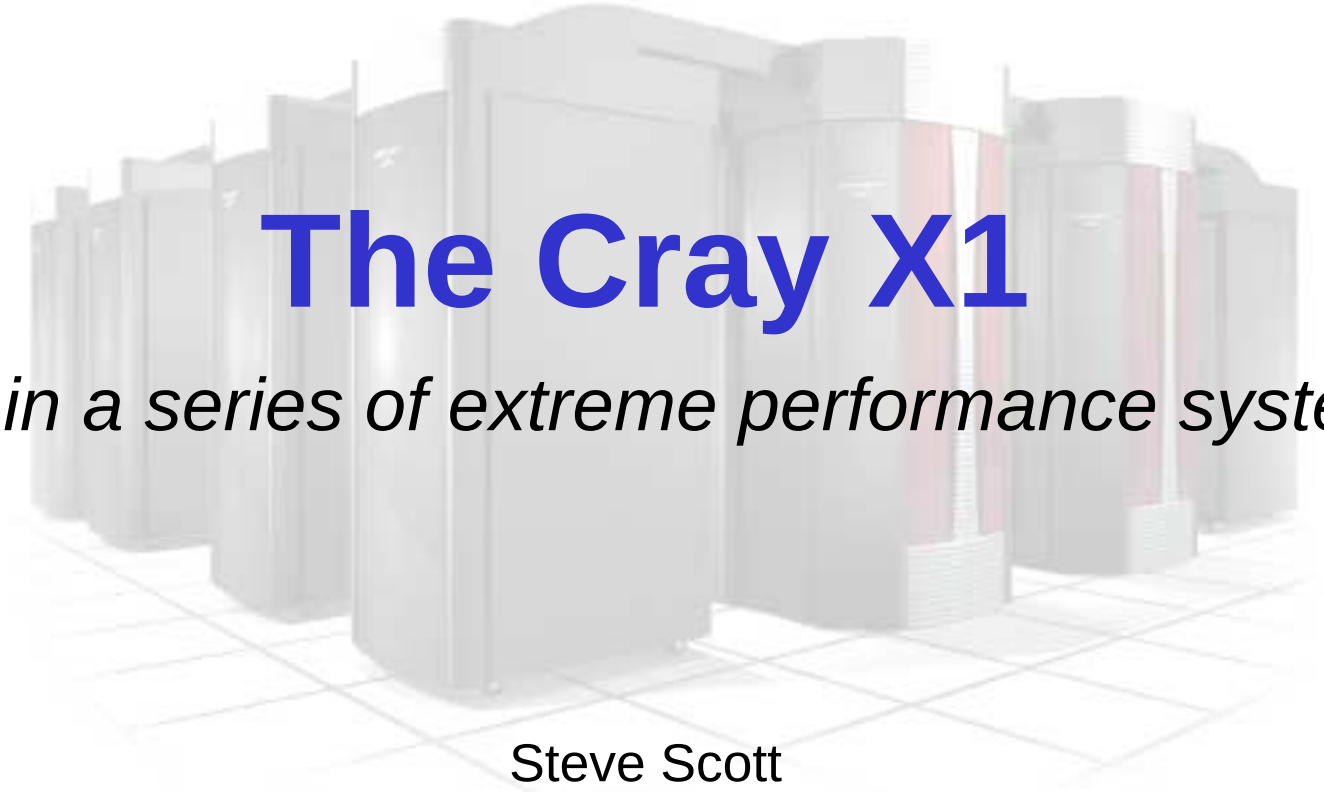
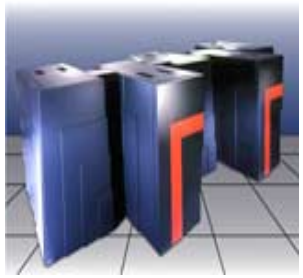


The Cray X1

First in a series of extreme performance systems

Steve Scott
Cray X1 Chief Architect
sscott@cray.com

Cray Supercomputer Evolution and Roadmap



sustained petaflops

High BW MPP

DARPA HPCS

BW++

BW+

BlackWidow

X1

X1e

Microprocessor-based MPP
1000's of processors
UNICOS/mk

1350

1200

T3E

T3D

SV1ex

SV1

J90

T90

C90

Y-MP

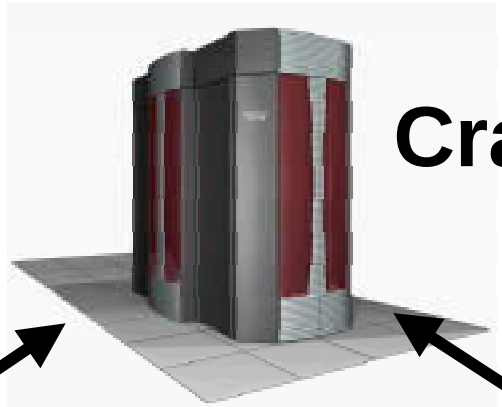
X-MP

Cray-1

High Bandwidth Vector
1-32 processors

Architecture

Cray X1



Cray PVP

- Powerful single processors
- Very high memory bandwidth
- Non-unit stride computation
- Special ISA features

- Modernized the ISA and microarchitecture

Cray T3E

- Distributed shared memory
- High BW scalable network
- Optimized communication and synchronization features

- Improved via custom processors

Cray X1 Instruction Set Architecture

New ISA

- Much larger register set (32x64 vector, 64+64 scalar)
- All operations performed under mask
- 64- and 32-bit memory and IEEE arithmetic
- Integrated synchronization features

Advantages of a vector ISA

- Compiler provides useful dependence information to hardware
- Very high single processor performance with low complexity

  ops/sec = (cycles/sec) * (instrs/cycle) * (ops/instr)
- Localized computation on processor chip
  - large register state with very regular access patterns
  - registers and functional units grouped into local clusters (pipes)

  ⇒ excellent fit with future IC technology
- Latency tolerance and pipelining to memory/network

  ⇒ very well suited for scalable systems
- Easy extensibility to next generation implementation

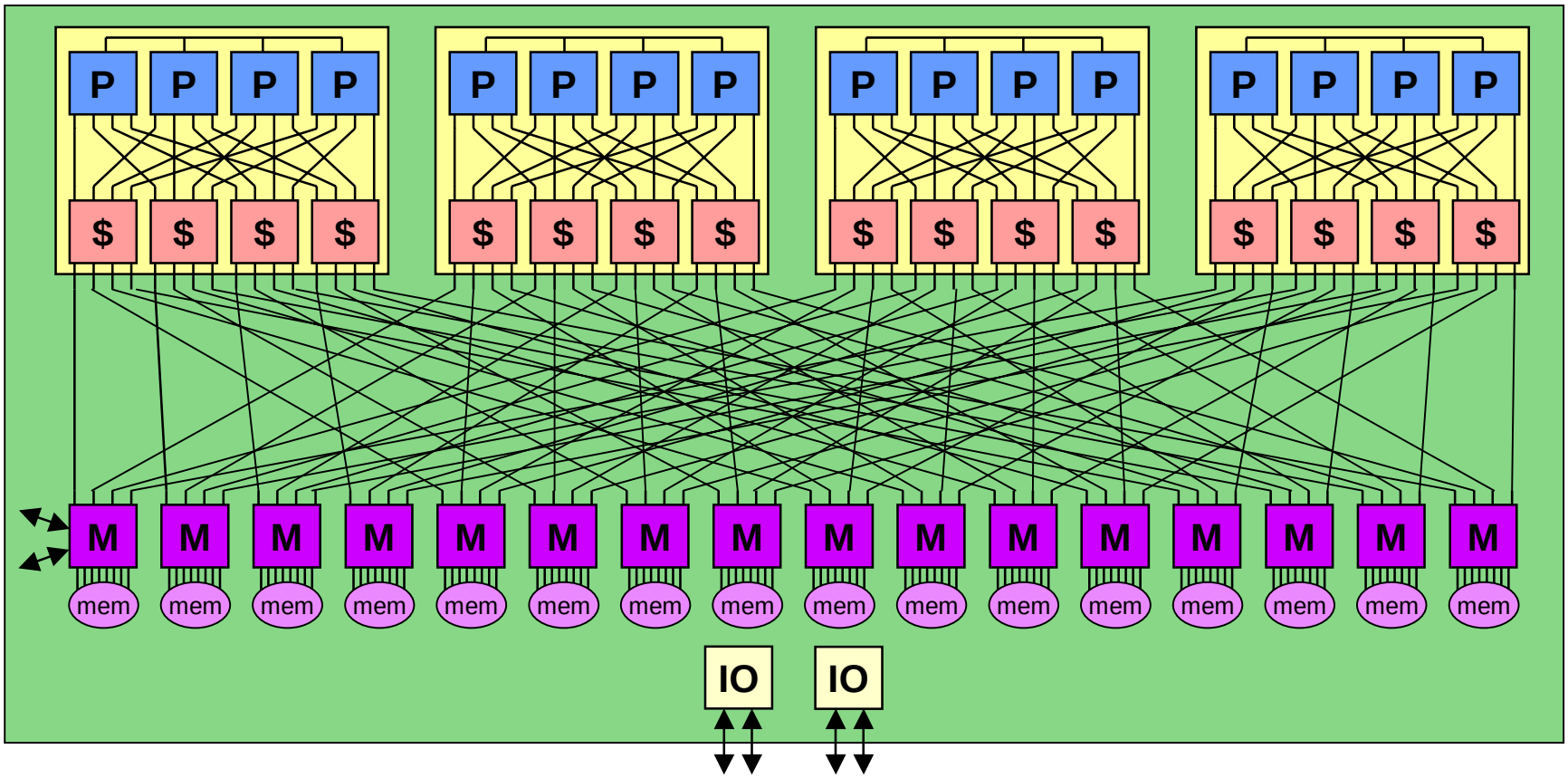
Not your father's vector machine

- New instruction set
- New system architecture
- New processor microarchitecture

- "Classic" vector machines were programmed *differently*
  - Classic vector: Optimize for loop length with little regard for locality
  - Scalable micro: Optimize for locality with little regard for loop length

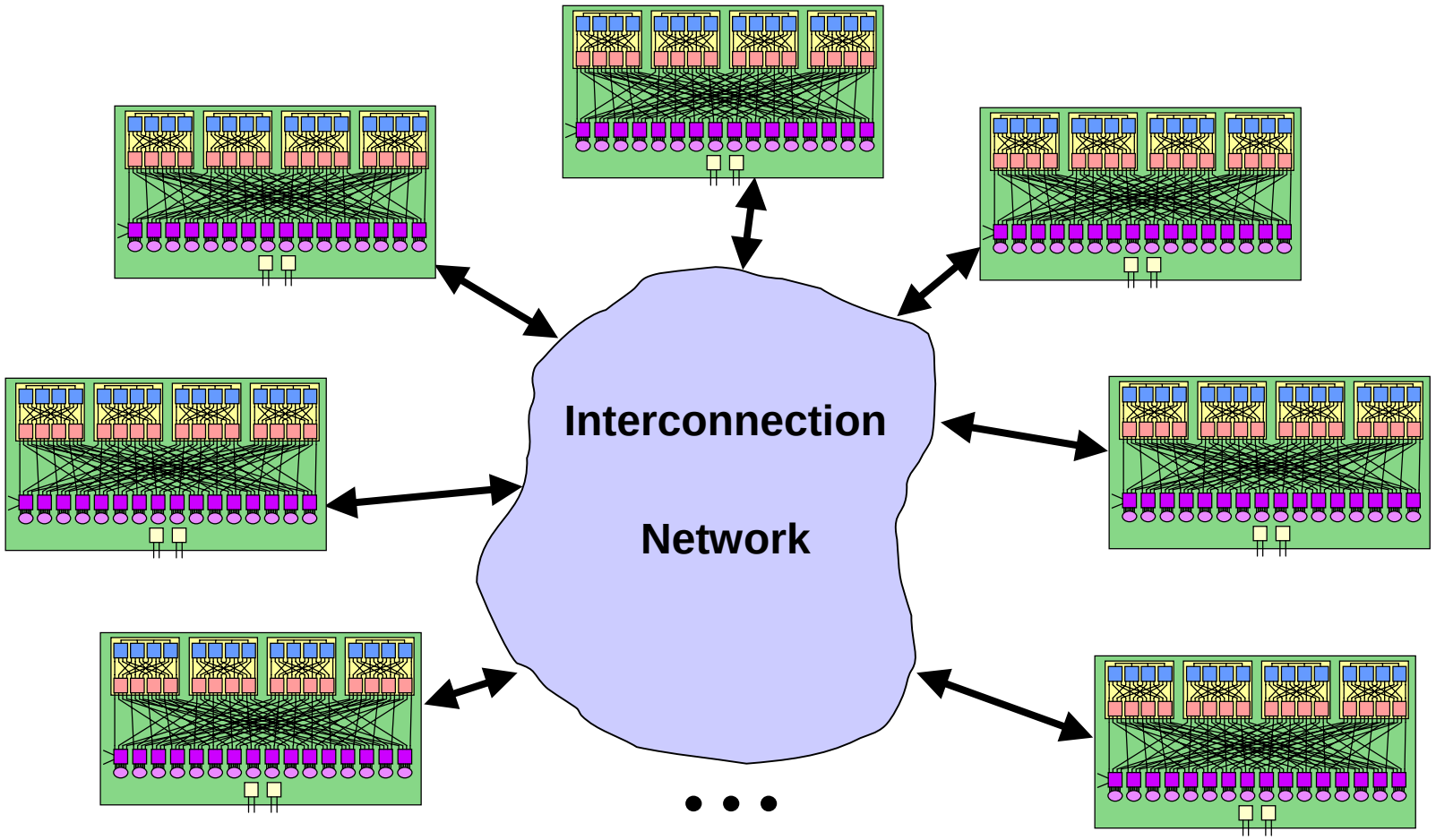
- The Cray X1 is programmed like a parallel micro-based machine
  - Rewards locality: register, cache, local memory, remote memory
  - Decoupled microarchitecture performs well on short loop nests
  - (however, *does* require vectorizable code)

Cray X1 Node



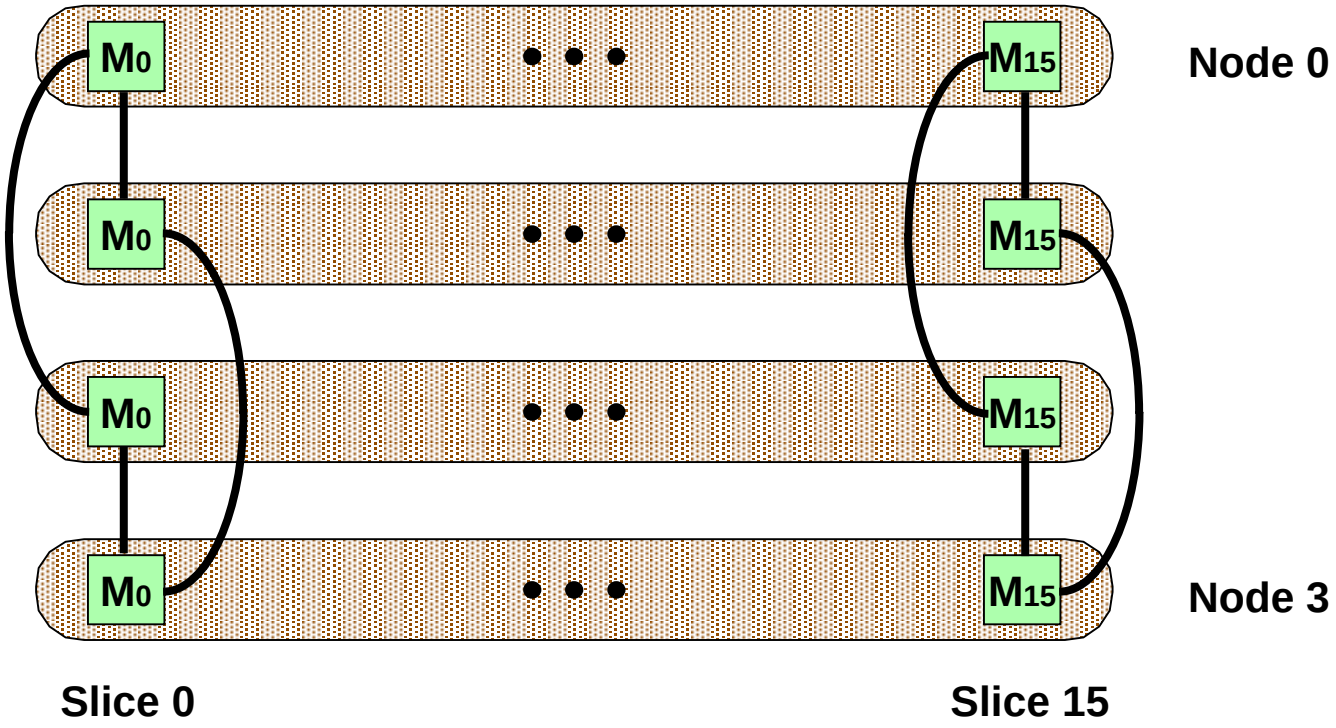
- Four multistream processors (MSPs), each 12.8 Gflops
- High bandwidth local shared memory (128 Direct Rambus channels)
- 32 network links and four I/O links per node

NUMA Scalable up to 1024 Nodes



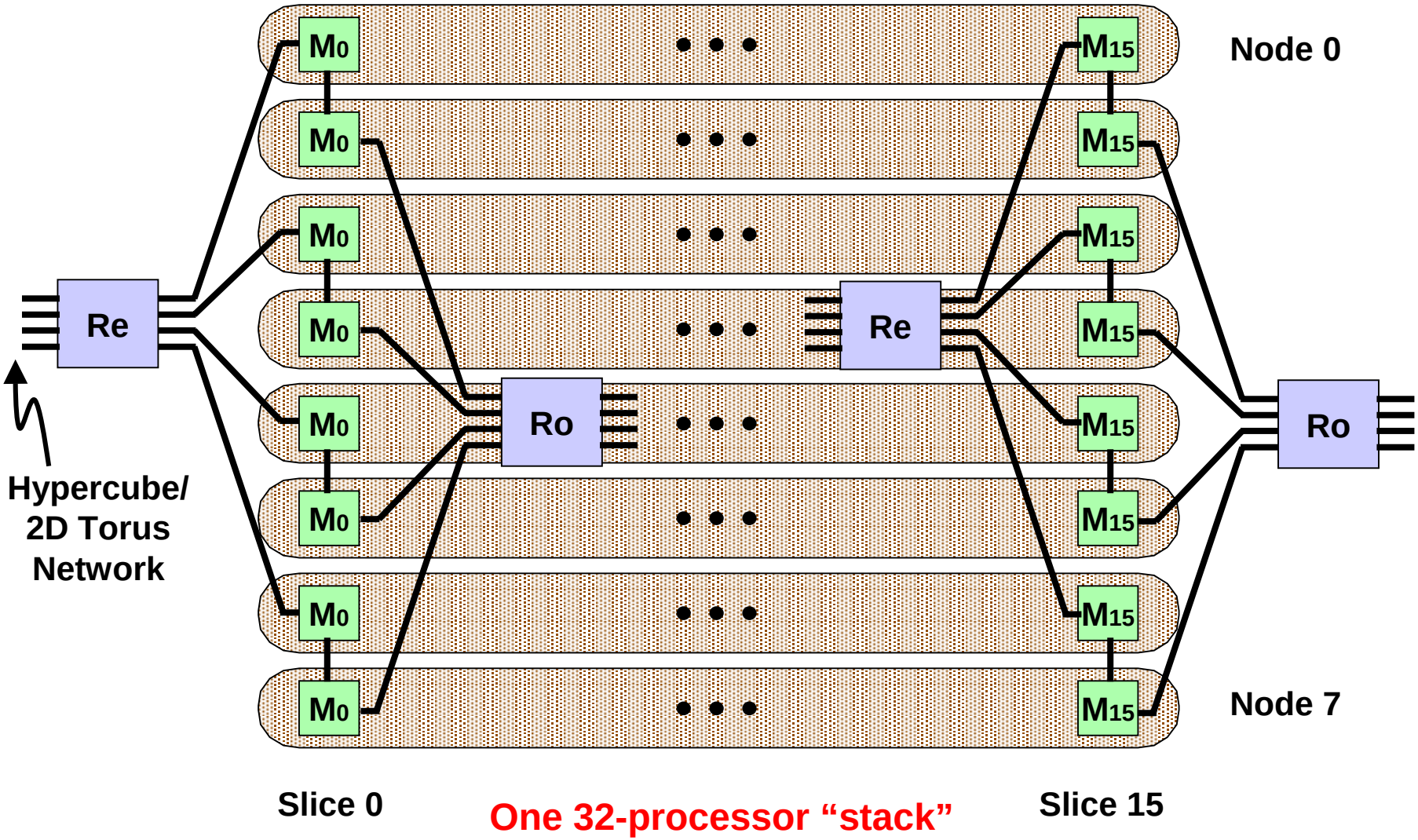
- 32 parallel networks for bandwidth
- Global shared memory across machine

Network for Small Systems



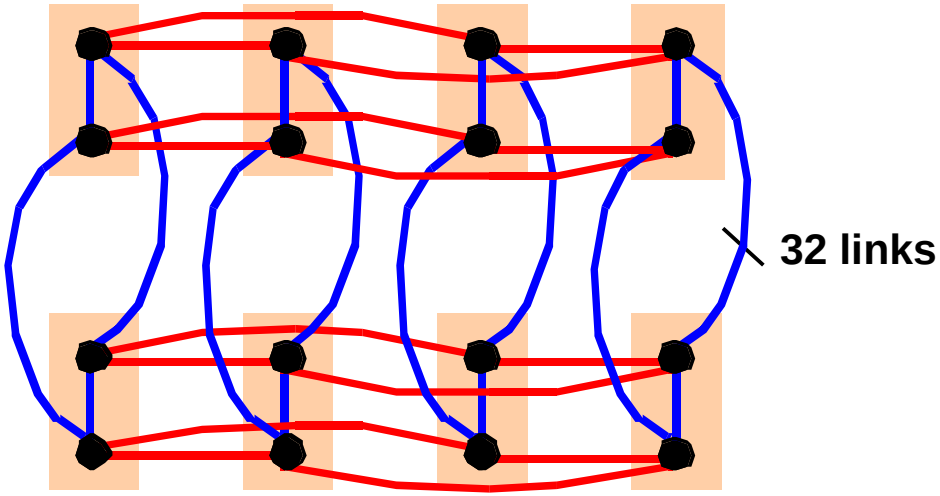
- Up to 16 CPUs, can connect directly via M chip ports

Scalable Network Building Block



Multicabinet Configuration

- Scales as a 2D torus of “stacks”
  - ~third dimension within a stack (32 links between stacks)
  - two stacks per cabinet

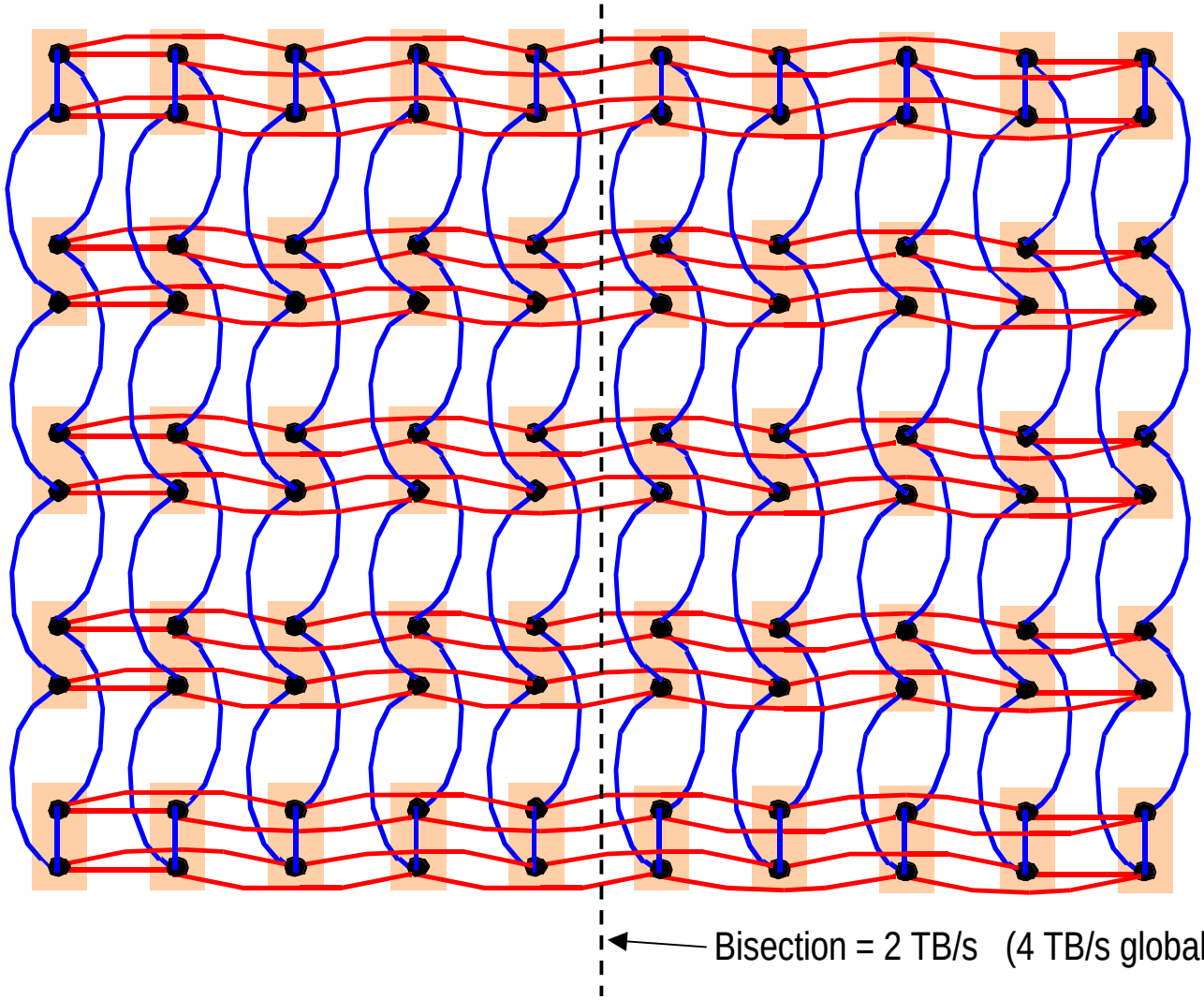


Bird’s Eye View

8 cabinets, 512 processors, 6.5 Tflops

A 40 Tflop Configuration

50 cabinets, 3200 processors



Designed for Scalability
T3E Heritage

- Distributed shared memory (DSM) architecture
  - Low latency, load/store access to entire machine (tens of TBs)
- Decoupled vector memory architecture for latency tolerance
  - Thousands of outstanding references, flexible addressing
- Very high performance network
  - High bandwidth, fine-grained transfers
  - Same router as Origin 3000, but 32 parallel copies of the network
- Architectural features for scalability
  - Remote address translation
  - Global coherence protocol optimized for distributed memory
  - Fast synchronization
- Parallel I/O scales with system size

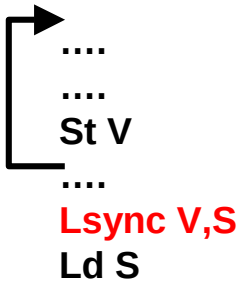
Decoupled Microarchitecture

- Decoupled access/execute *and* decoupled scalar/vector
- Scalar unit runs ahead, doing addressing and control
  - Scalar and vector loads issued early
  - Store addresses computed and saved for later use
  - Operations queued and executed later when data arrives
- Hardware dynamically unrolls loops
  - Scalar unit goes on to issue next loop before current loop has completed
  - Full scalar register renaming through 8-deep shadow registers
  - Vector loads renamed through load buffers
  - Special sync operations keep pipeline full, even across barriers

This is *key* to making the system *like* short-VL code

Maintaining Decoupling Past Synchronization Points

Lsyncs ensure ordering within a processing unit



Msyncs ensure ordering among the processing units of an MSP

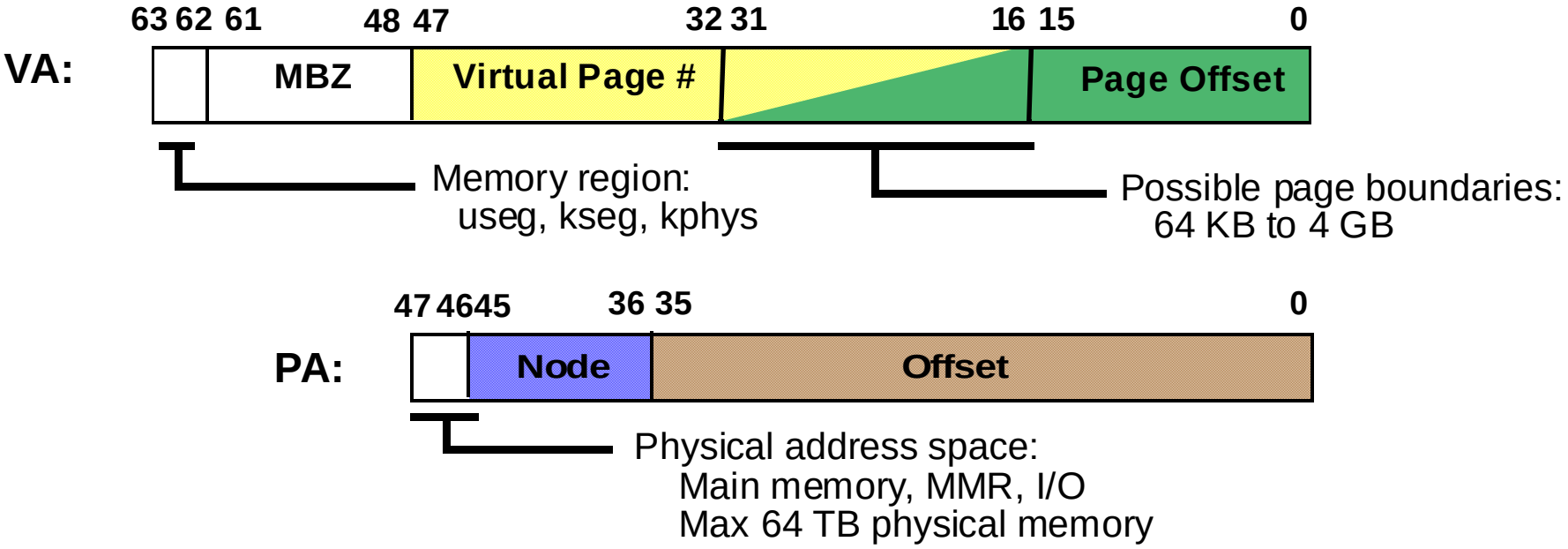
P0	P1	P2	P3
....	....	....	....
St Vx	St Vx	St Vx	St Vx
....	....	....	....
Msync	Msync	Msync	Msync
....	....	....	....
Ld	Ld	Ld	Ld

Want to protect against hazards, but *not* drain memory pipeline.

Vector store addresses computed early (before data is available):

- run past scalar Dcache to invalidate any matching entries
- sent out to the Ecache
- provides ordering with later scalar load or loads from *other* P chips

Address Translation



- High translation bandwidth: scalar + four vector translations per cycle per P
- Source translation using 256 entry TLBs with support for multiple page sizes
- Remote (hierarchical) translation:
  - allows each node to manage its own memory (eases memory mgmt.)
  - TLB only needs to hold translations for one node ⇒ *scales*

Cache Coherence

Global coherence, but only cache memory from local node (8-32 GB)

- Supports SMP-style codes up to 4 MSP (4-way sharing)
- References outside this domain converted to non-allocate
- Scalable codes use explicit communication anyway
- Keeps directory entry and protocol simple

Explicit cache allocation control

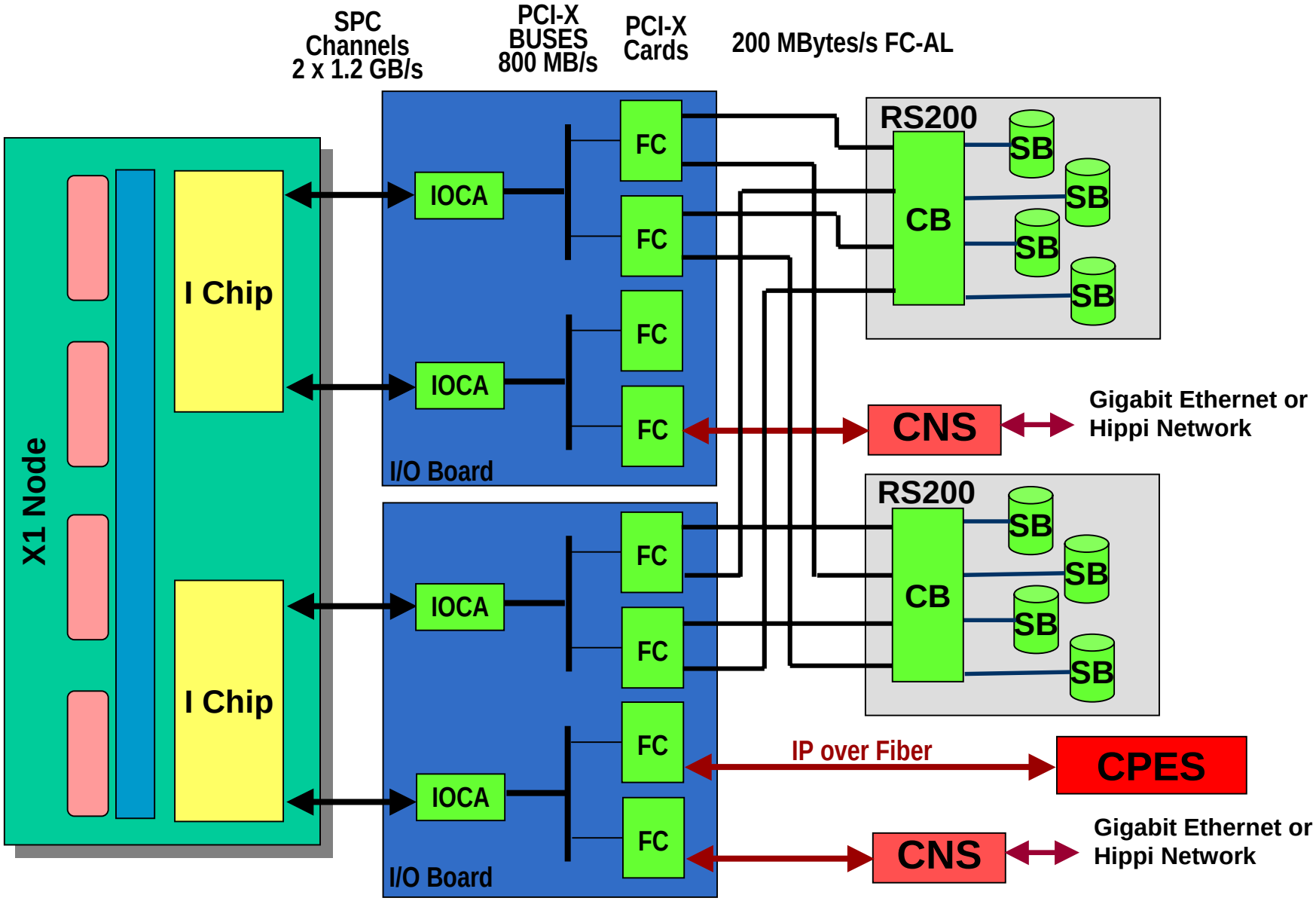
- Per instruction hint for vector references
- Per page hint for scalar references
- Use non-allocating refs for explicit communication
  or to avoid cache pollution

Coherence directory stored on the M chips (rather than in DRAM)

- Low latency and *really* high bandwidth to support vectors
  - Typical CC system: 1 directory update per proc per 100 or 200 ns
  - Cray X1: 3.2 dir updates per MSP per ns (factor of several hundred!)

System Software

Cray X1 I/O Subsystem

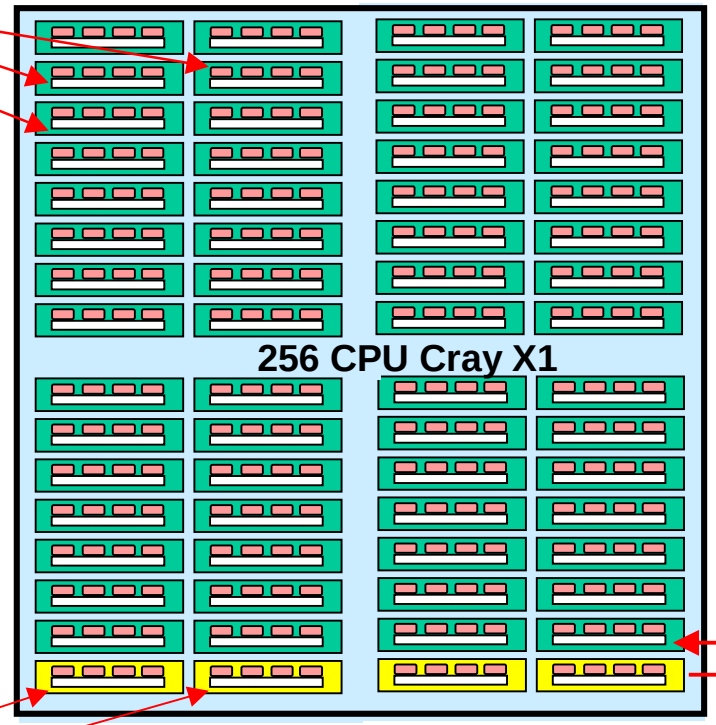


Cray X1 UNICOS/mp

Single System Image

UNIX kernel executes on each *Application* node (somewhat like Chorus™ microkernel on UNICOS/mk)

Provides: SSI, scaling & resiliency



UNICOS/mp Global resource manager (Psched) schedules applications on Cray X1 nodes

System Service nodes provide file services, process management and other basic UNIX functionality (like /mk servers). User commands execute on *System Service* nodes.

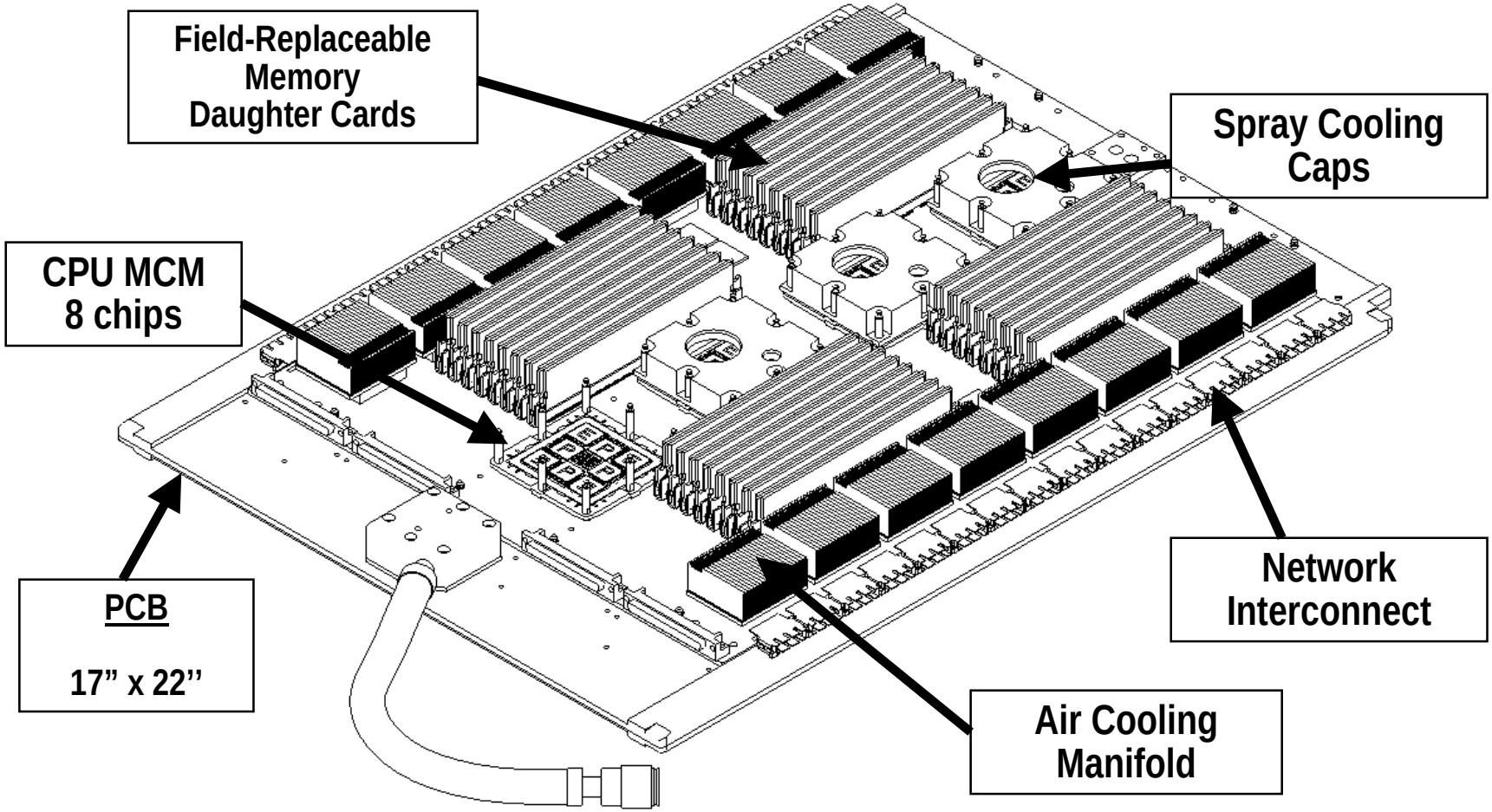
Commands launch applications like on T3E (mpprun)

Programming Models

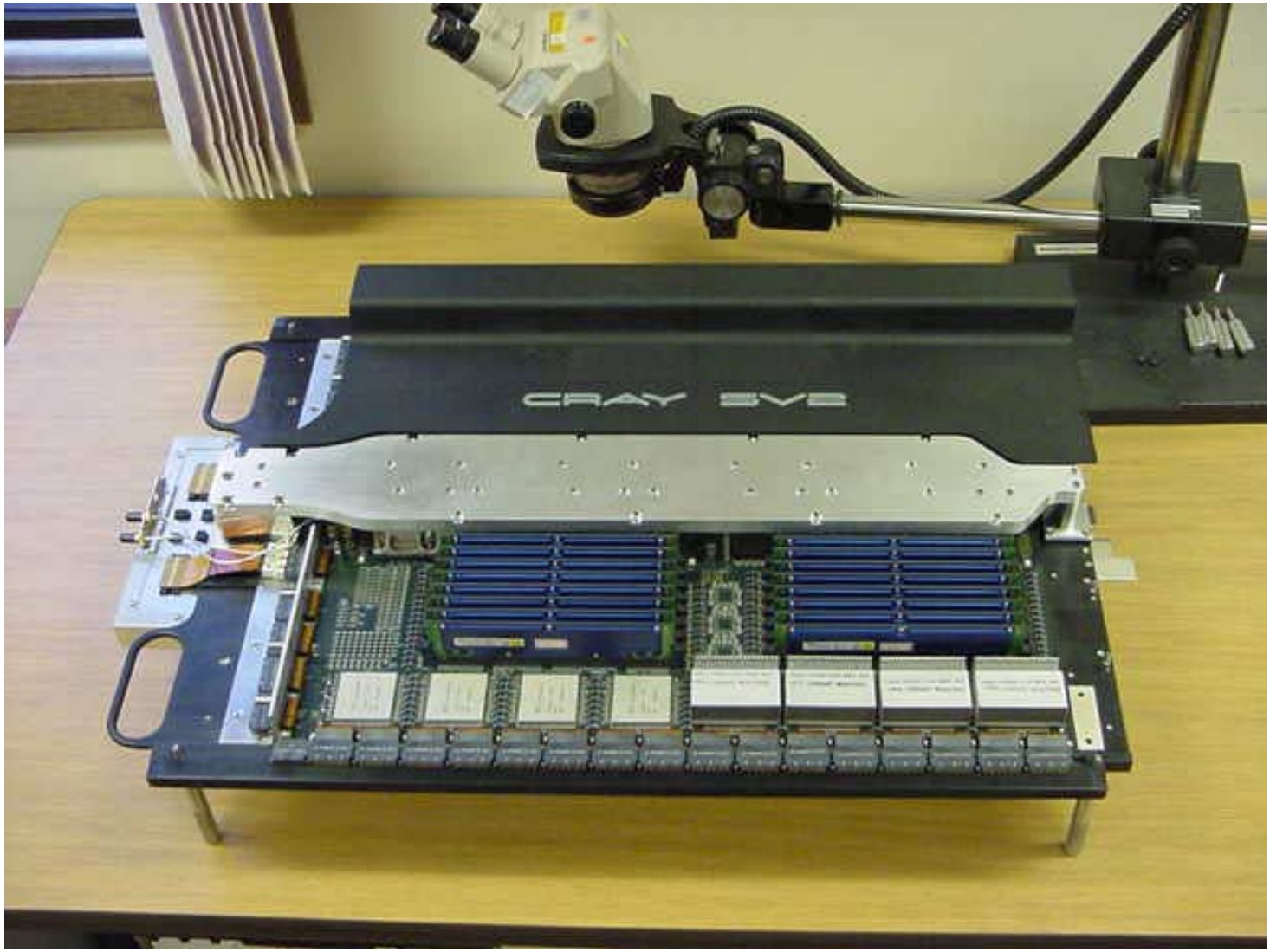
- Traditional shared memory applications
  - OpenMP, pthreads
  - 4 MSPs
  - Single node memory (8-32 GB)
  - Very high memory bandwidth
  - No data locality issues
- Distributed memory applications
  - MPI, shmem(), UPC, Co-array Fortran
  - Rewards optimizations for cluster/NUMA micro-based machines
    - work and data decomposition
    - cache blocking
  - But less worry about communication/computation ratio, strides and bandwidth
    - multiple GB/s network bandwidth between nodes
    - scatter/gather and large-stride support

Mechanical Design

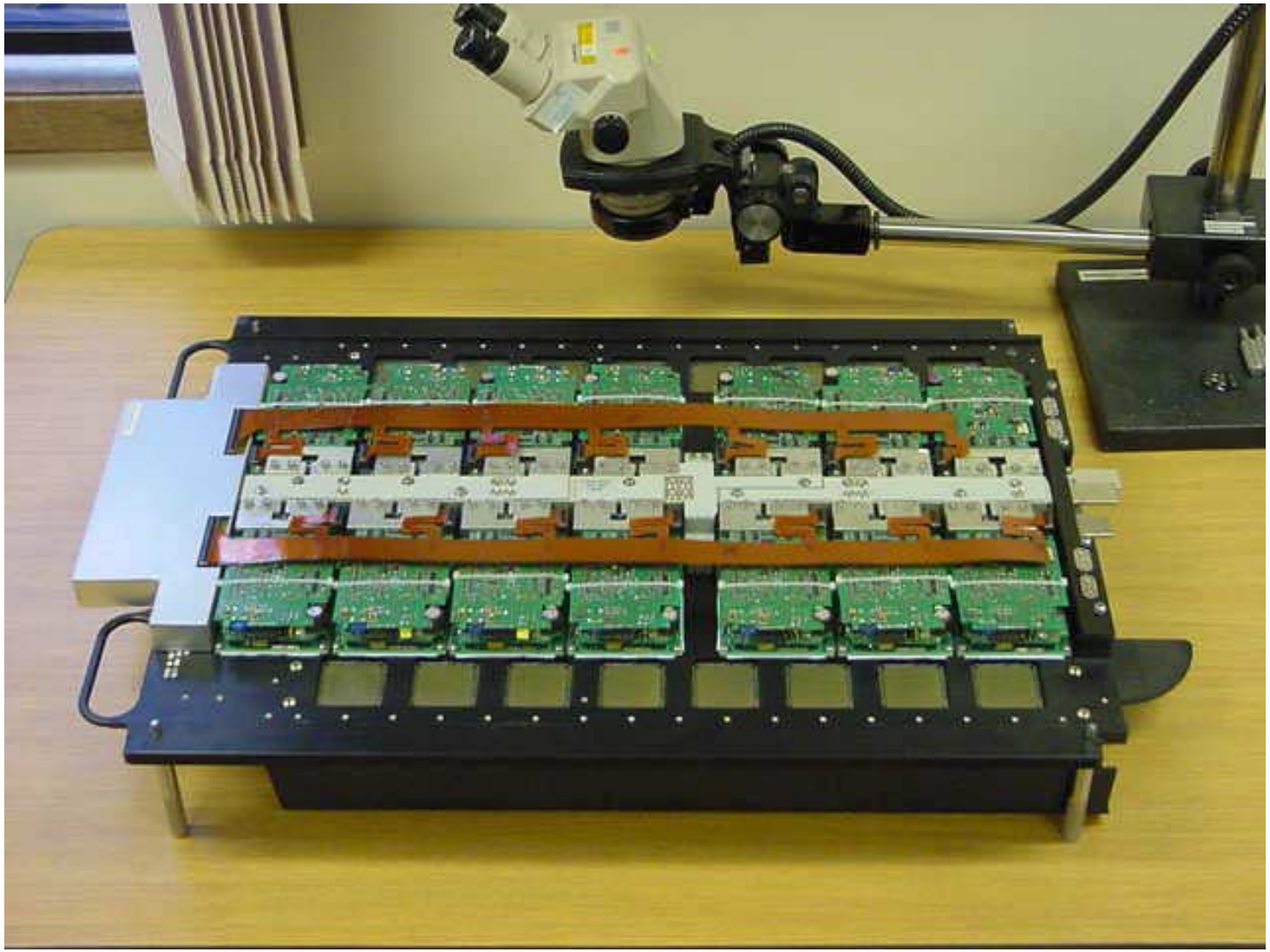
Packaging - node board lower side



Cray X1 Node Module



Node Module (Power Converter side)



Cray X1 Chassis

64 Processor Cray X1 System

~820 *Gflops*

256 Processor Cray X1 System

~ 3.3 Tflops

Cray X1 Summary

- Extreme system capability
  - Tens of TFLOPS in a Single System Image (SSI) MPP
  - Efficient operation due to balanced system design
  - Focus on sustained performance on the most challenging problems

Achieved via:

- Very powerful single processors
  - High instruction-level parallelism
  - High bandwidth memory system
- Best in the world scalability
  - Latency tolerance via decoupled vector processors
  - Very high-performance, tightly integrated network
  - Scalable address translation and communication protocols
  - Robust, highly scalable operating system (Unicos/mp)

Questions?