# SCI Summer School
## Trinity College Dublin
## October 2000

Håkon O. Bugge
Scali AS
mailto: hob@scali.no
http://www.scali.com

# Outline

Scalable Linux Systems

- Who’s Scali?
- Scalability issues with Shared Address Space cluster architectures
- Cons and Pros of a direct SCI network
- Fault tolerant routing in a 2D SCI Torus
- Low level SCI programming using ScaMPI
- Node level parallelism. Would that be pthreads, OpenMP, or MPI?
- Cluster Management through Scali's *Universe*

# Scali's Mission:

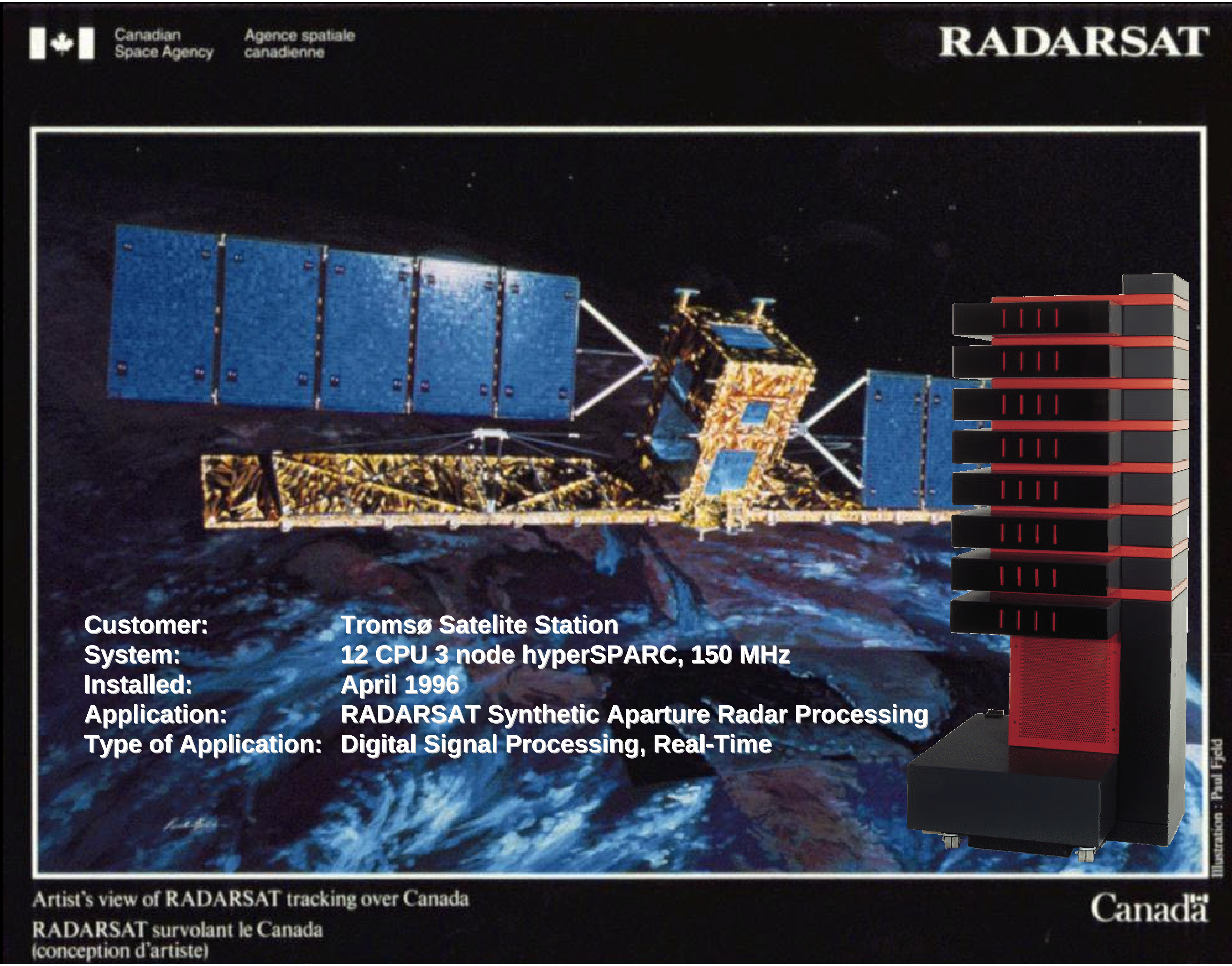
SCALI
Scalable Linux Systems

**Dedicated to provide**
**state-of-the-art middleware**
**and**
**system management software;**
**the key enabling technologies**
**for building**
***scalable systems!***

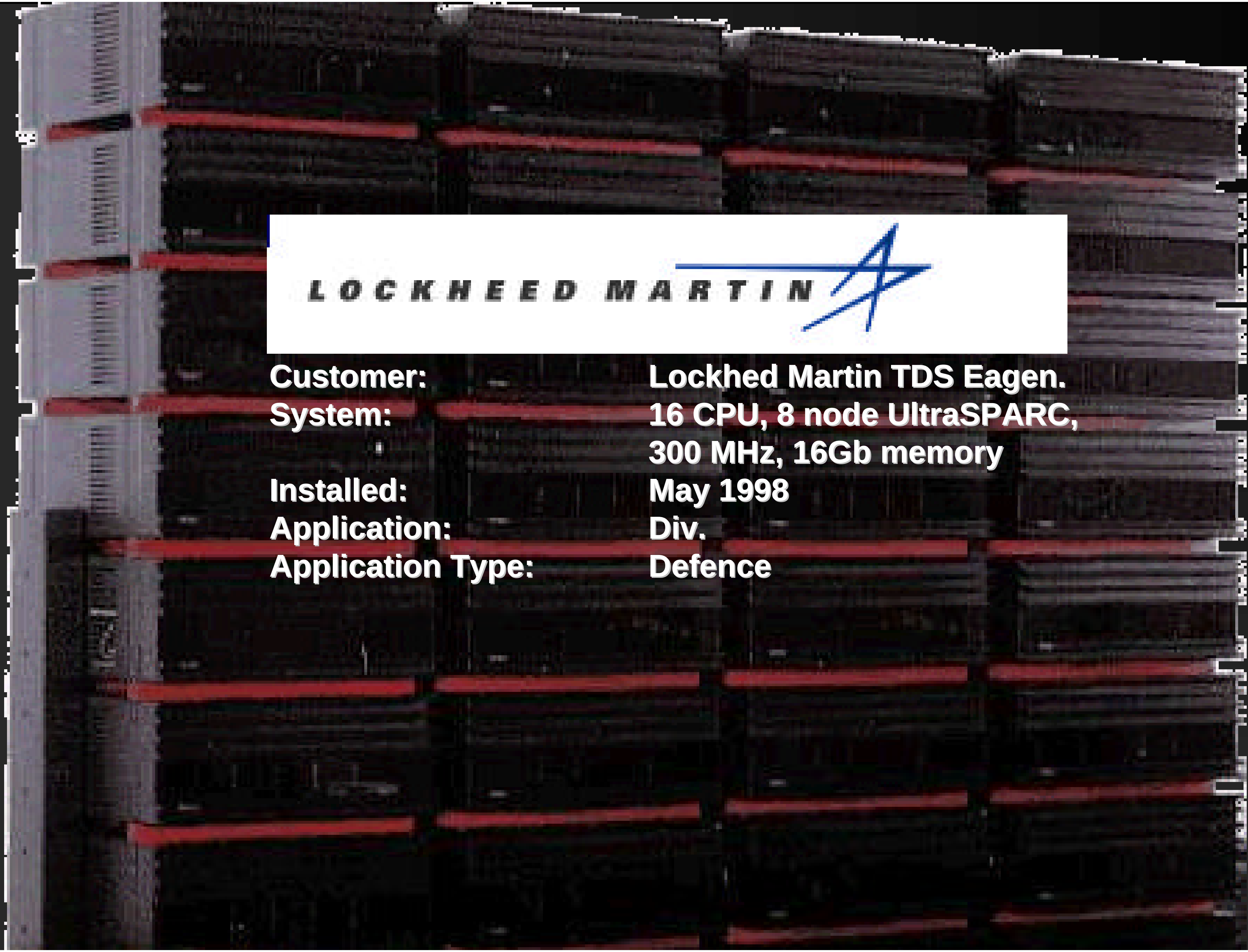
# Reference Installations

SCALI
Scalable Linux Systems

- Spacetec/Tromsø Satellite Station, Norway
- Norwegian Defense Research Establishment
- Parallab, Norway
- Paderborn Parallel Computing Center, Germany
- Spacebel, Belgium
- Aerospatiale, France
- Fraunhofer Gesellschaft, Germany
- Lockheed Martin Tactical Defense Systems, USA
- University of Geneva, Switzerland
- University of Oslo, Norway
- Uni-C. Denmark
- Paderborn Parallel Computing Center ”Phase-2”, Germany
- University of Lund, Sweden
- University of Aachen, Germany
- DNV, Norway
- DaimlerChrysler, Germany
- DaimlerChrysler, Germany, 2nd order
- BMW, Germany
- BMW, Germany, 2nd order
- Voith-Siemens Hydro
- Max Planck Institute für Plasmaphysik, Germany
- University of New Mexico, USA
- University of Alberta, Canada
- University of Manitoba, Canada
- Etnus Software, USA
- HP labs, USA
- University of Florida, USA
- Northern Lights, Japan
- Uni-Heidelberg, Germany
- GMD, Germany
- Uni-Giessen, Germany
- Uni-Hannover, Germany
- Uni-Düsseldorf, Germany
- VA Linux Systems, USA
- Alta Technology, USA
- ASL Workstations, USA



Canadian Space Agency
Agence spatiale canadienne
RADARSAT
Customer: Tromsø Satelite Station
System: 12 CPU 3 node hyperSPARC, 150 MHz
Installed: April 1996
Application: RADARSAT Synthetic Aparture Radar Processing
Type of Application: Digital Signal Processing, Real-Time
Illustration : Paul Fjeld
Artist's view of RADARSAT tracking over Canada
RADARSAT survolant le Canada (conception d'artiste)
Canada



| | |
|---|---|
| Customer: | Lockhed Martin TDS Eagen. |
| System: | 16 CPU, 8 node UltraSPARC, 300 MHz, 16Gb memory |
| Installed: | May 1998 |
| Application: | Div. |
| Application Type: | Defence |

DAIMLERCHRYSLER

| | |
|---|---|
| **Customer:** | **Chrysler Daimler** |
| **System:** | **32 CPU, 16 node Pentium II, 500 MHz, 16Gb memory** |
| **Installed:** | **December 1999** |
| **Application:** | **FEKO** |
| **Application Type:** | **ElectroMagnetic Simulation** |

**Upgrade to 64 CPUs, November 2000**


Customer: Paderborn center for Parallel Computing
System: 192 CPU, 96 node Pentium II, 450 MHz
Installed: April 1999
Application: Research, Industry, Chess

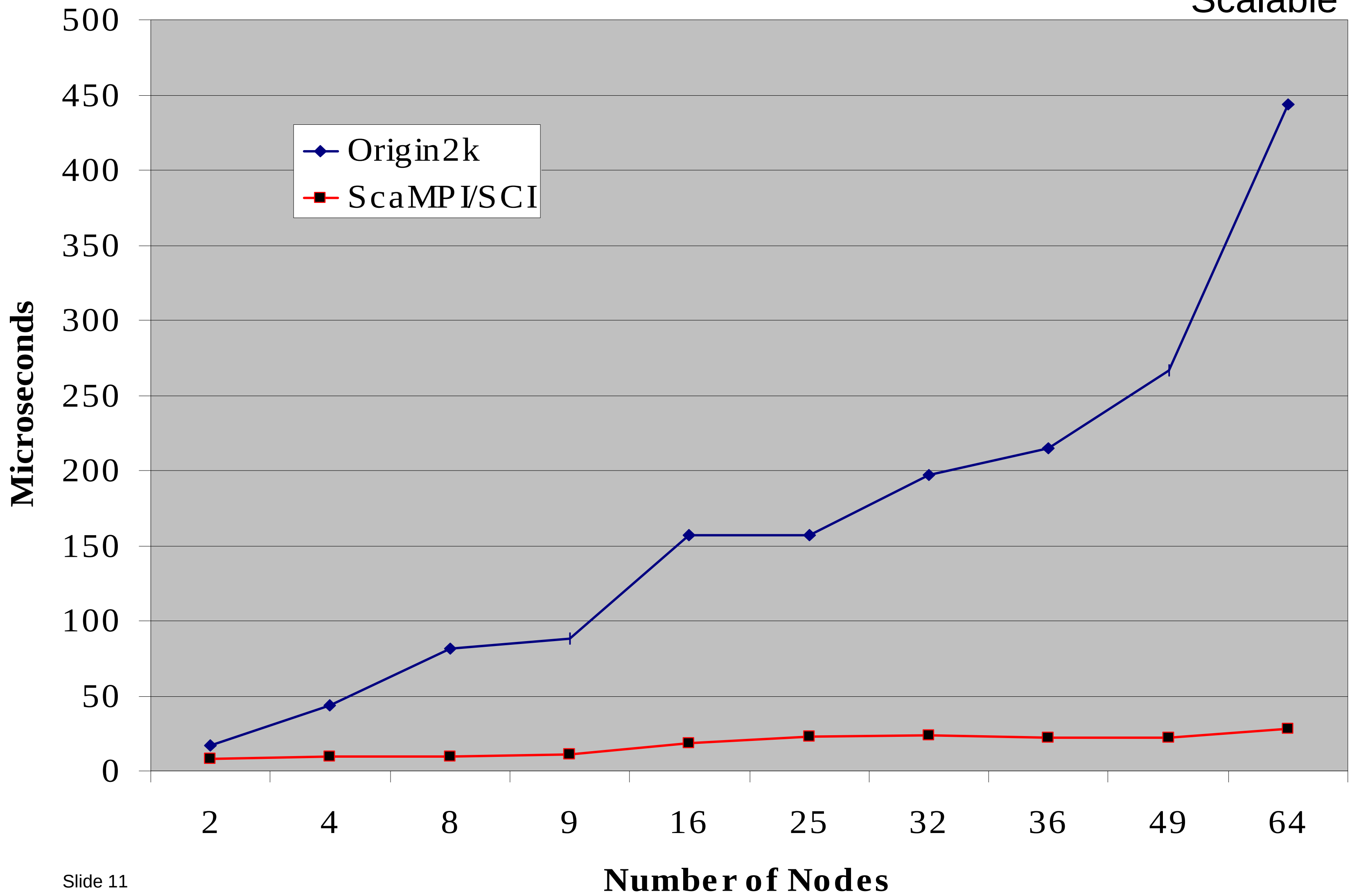
# Scalability (N is #nodes)

SCALI
Scalable Linux Systems

- **Latency**
  - Constant wrt. N (theory)
  - O(log N) (practise)
- **Bandwidth**
  - Constant per node
  - Accumulated proportional to N

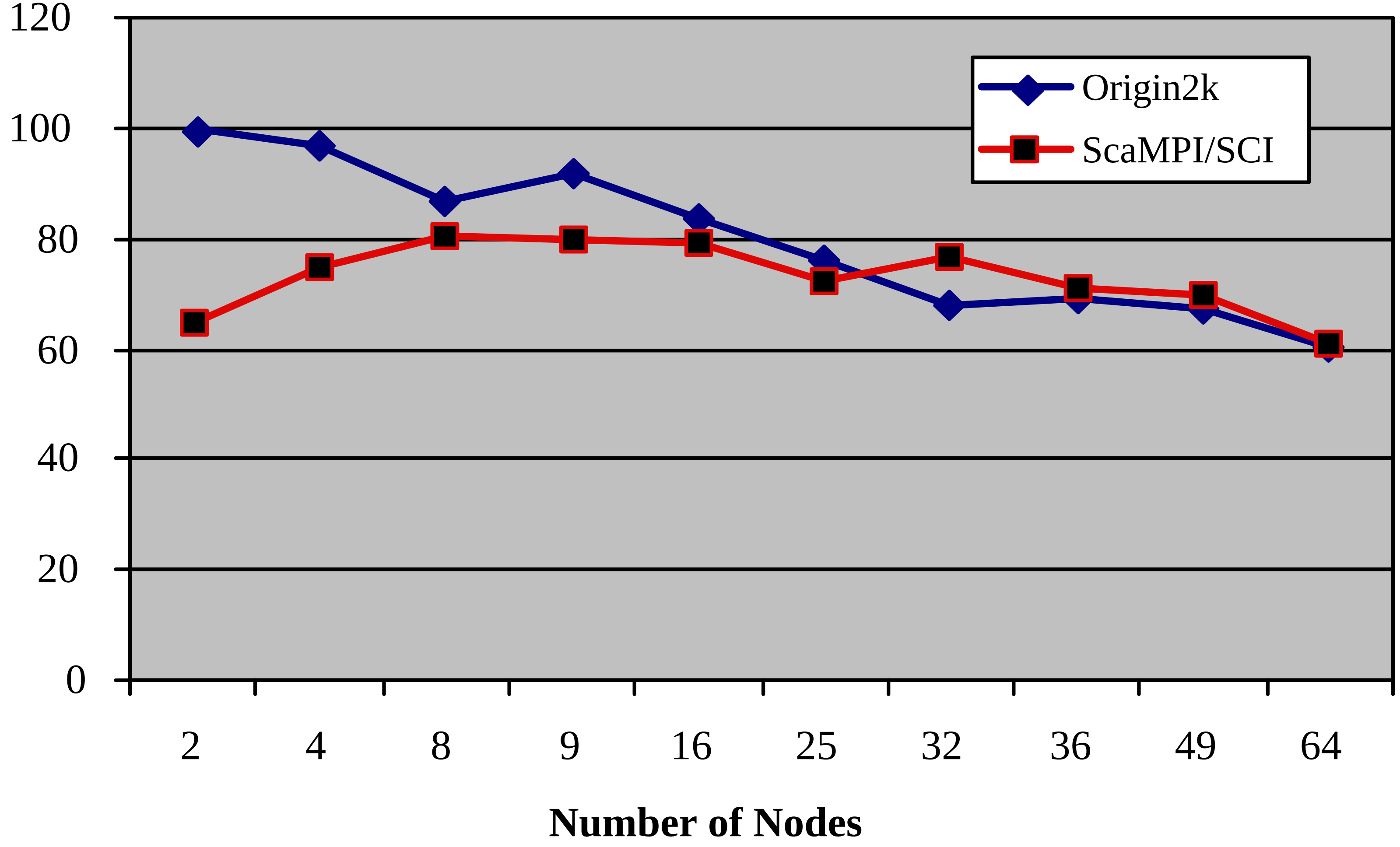
# MPI_Barrier() latency (smaller is better)

Scalable Linux Systems



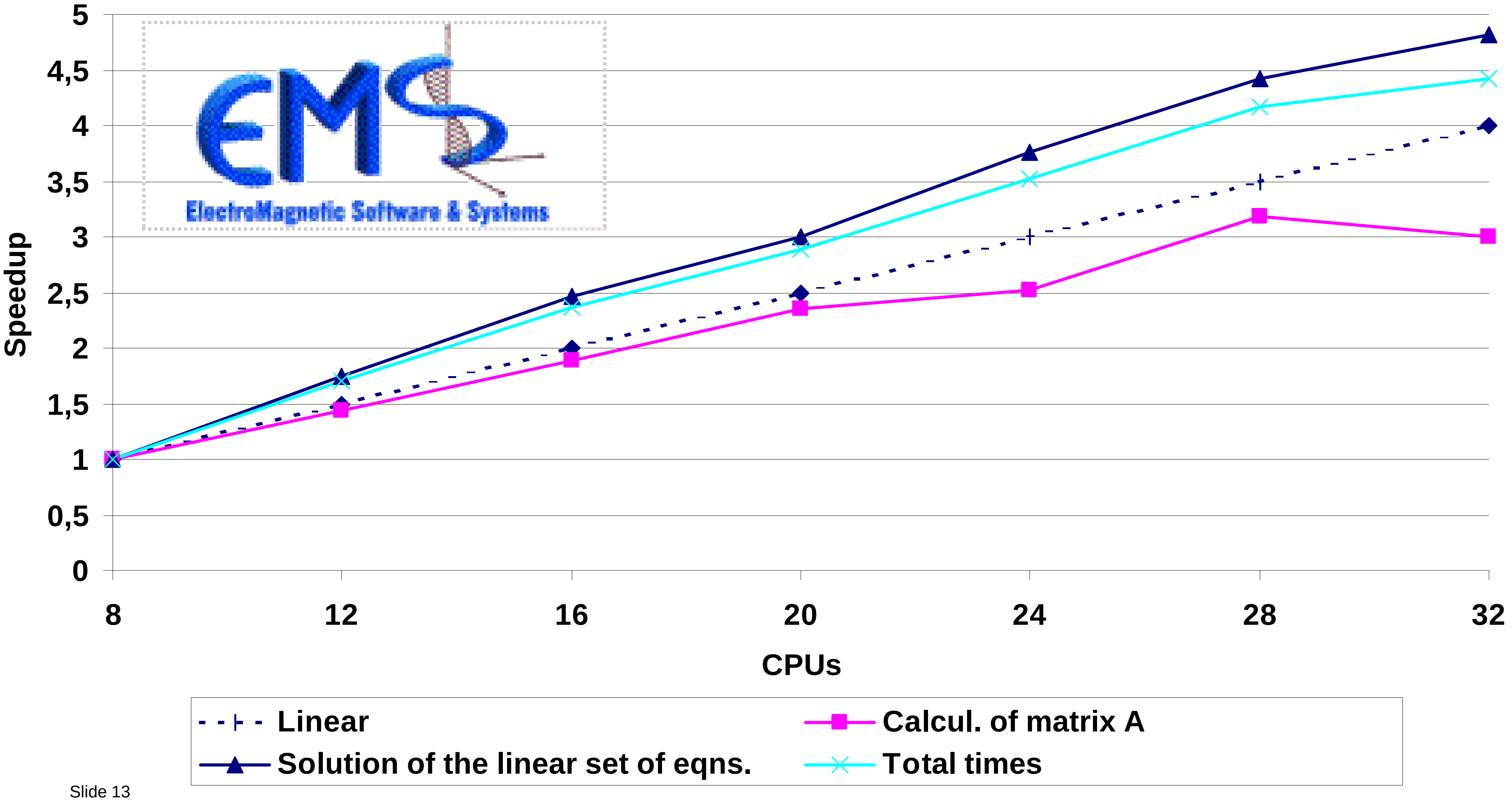
# MPI_Alltoall() bandwidth per compute node

SCALI
Scalable Linux Systems



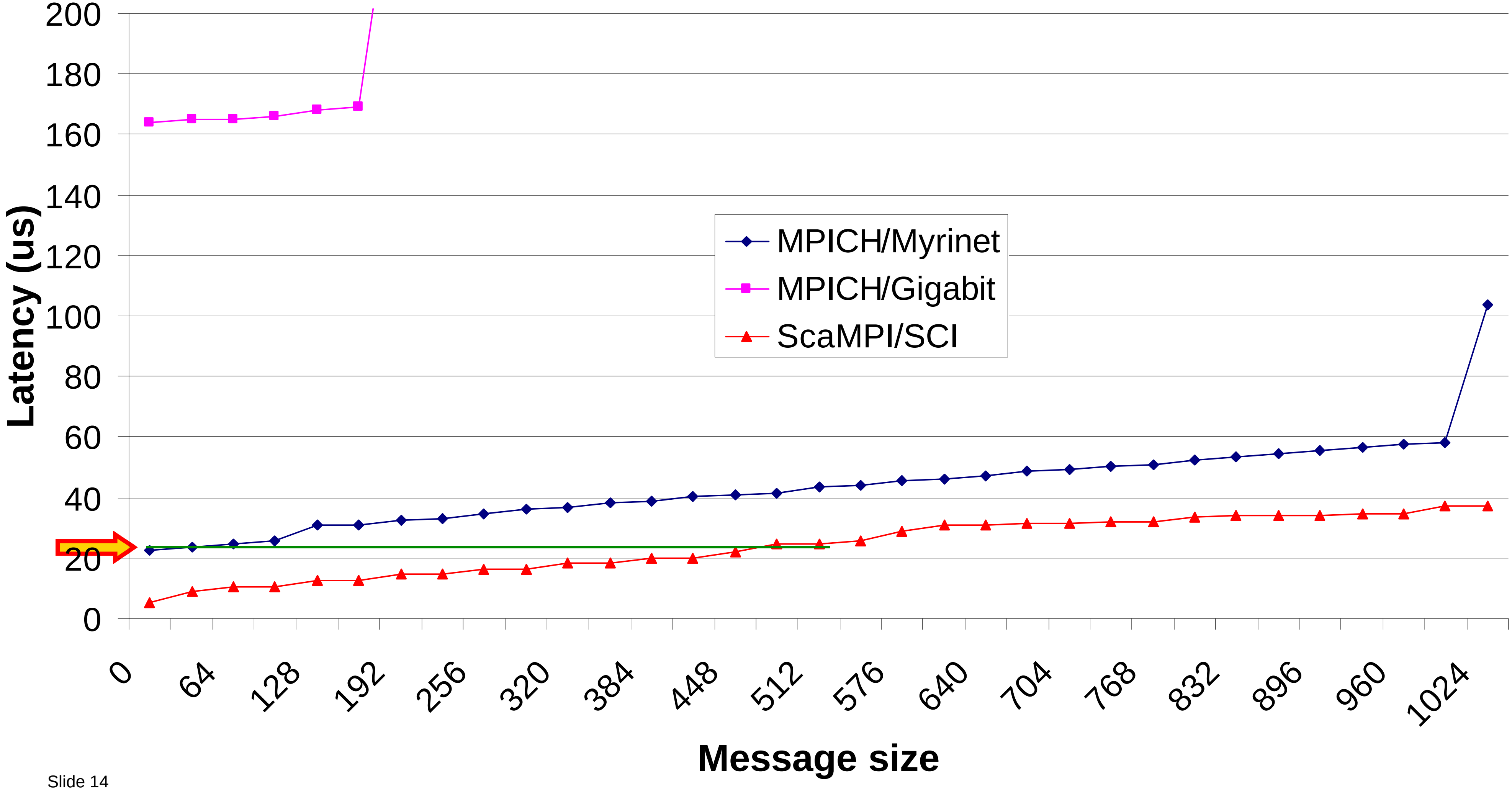
# FEKO: Parallel Speedup

SCALI
Scalable Linux Systems



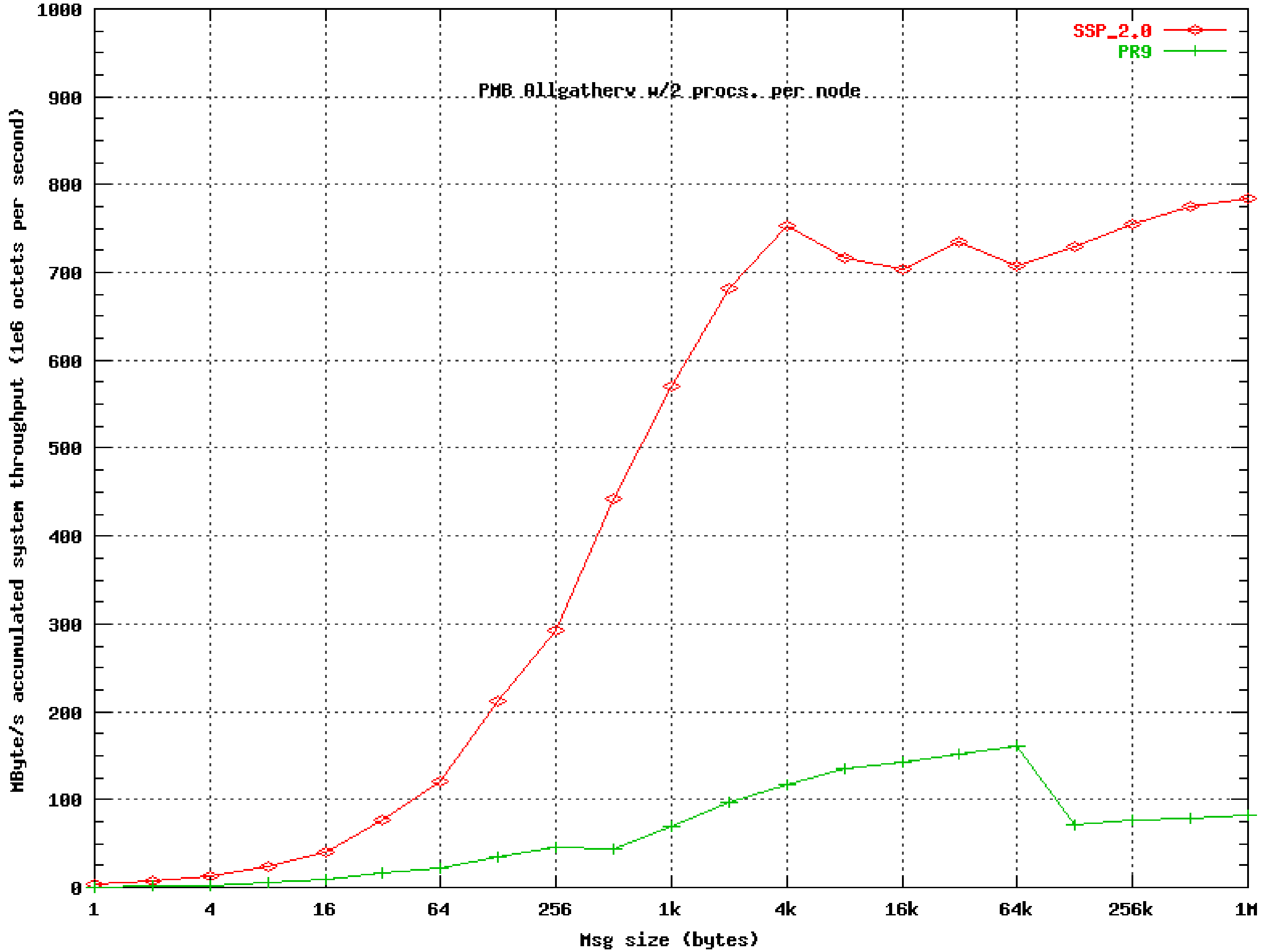
# Ping-pong latency

SCALI
Scalable Linux Systems



# 32 process Allgatherv

Scalable Linux Systems



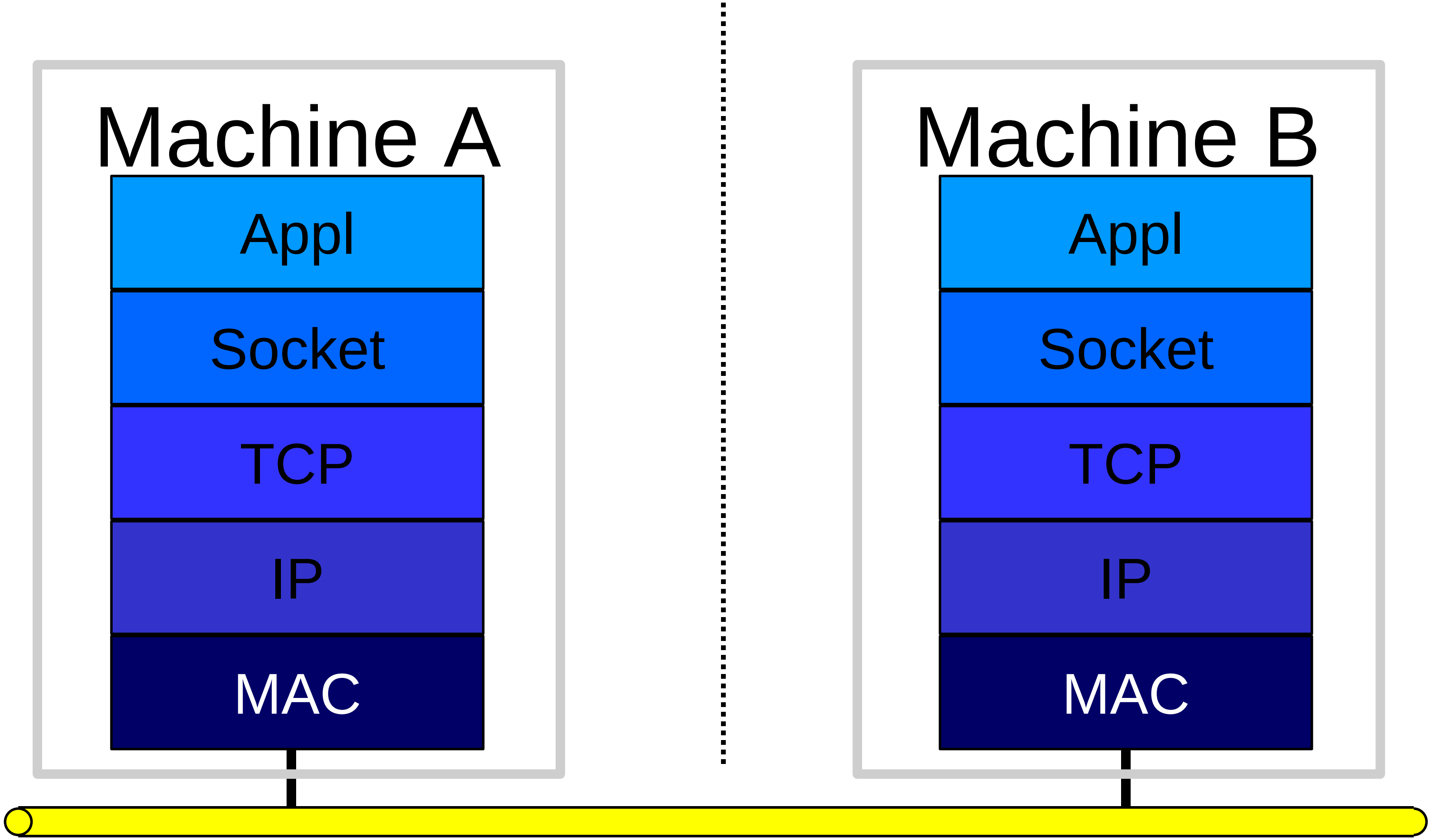
# Outline

Scalable Linux Systems

- ✓ Who's Scali?
- ✓ **Scalability issues with Shared Address Space cluster architectures**
- ✓ Cons and Pros of a direct SCI network
- ✓ Fault tolerant routing in a 2D SCI Torus
- ✓ Low level SCI programming using ScaMPI
- ✓ Node level parallelism. Would that be pthreads, OpenMP, or MPI?
- ✓ Cluster Management through Scali's *Universe*

# Shared Nothing
# Communication Architecture

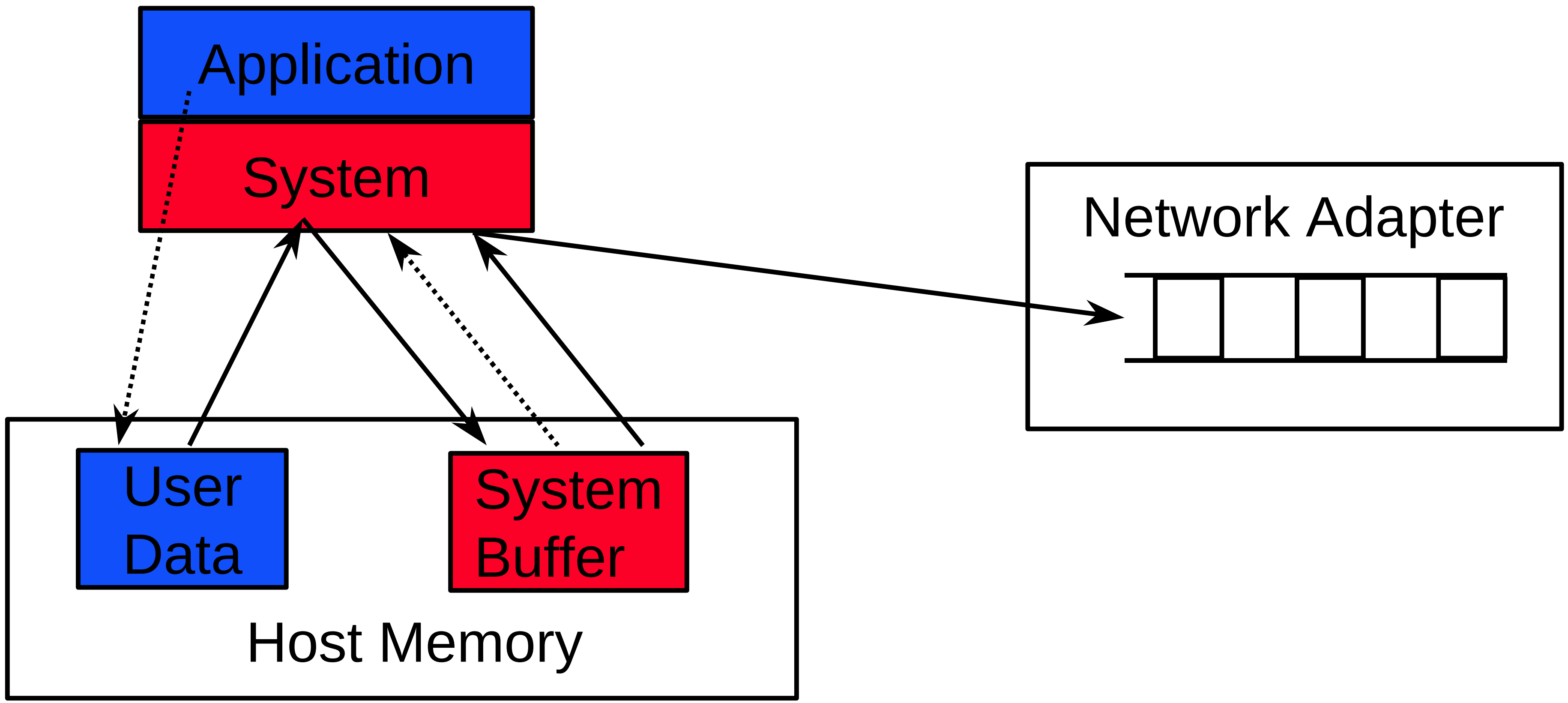
SCALI
Scalable Linux Systems




Protocol Entities embedded in Packets

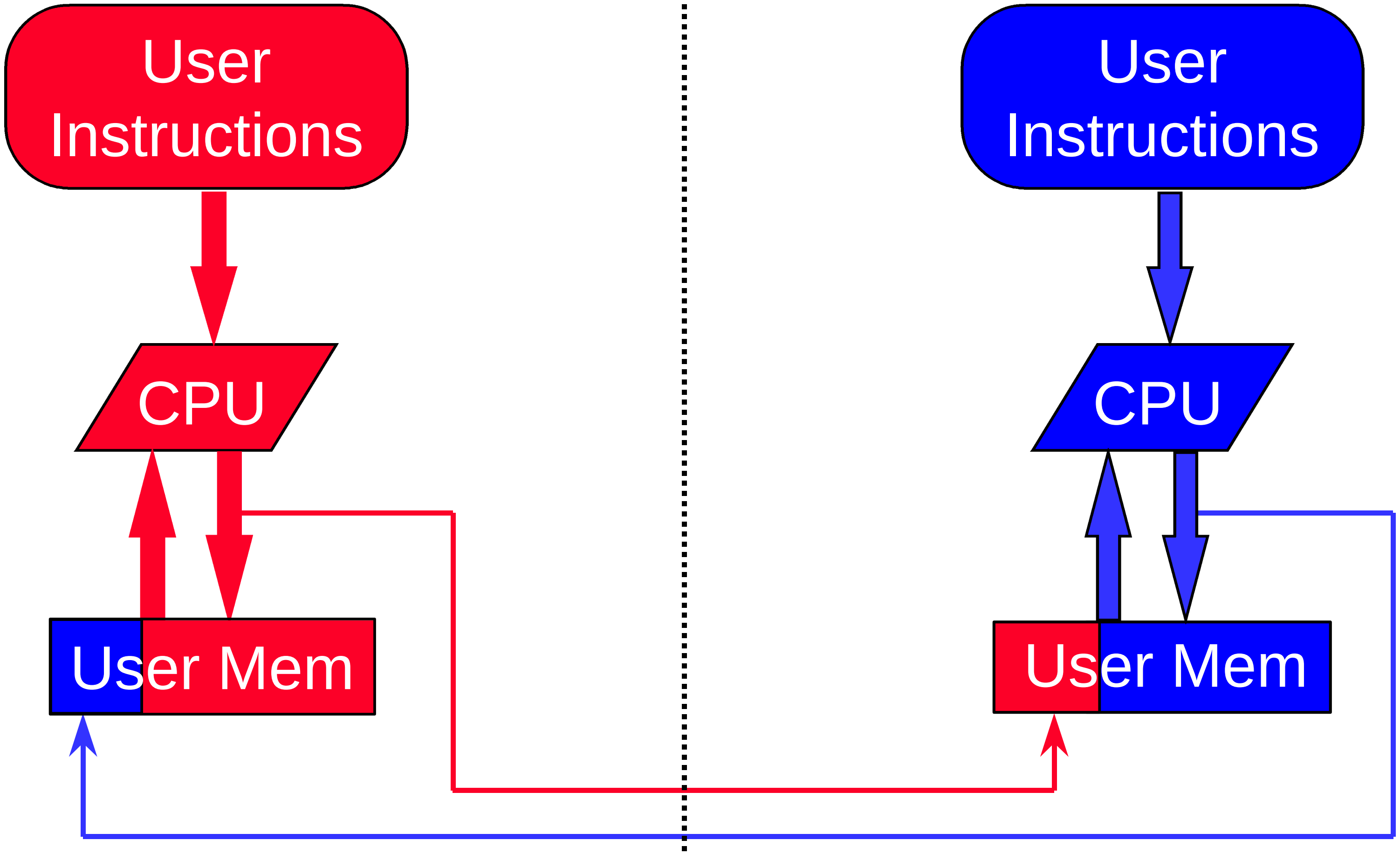
# Shared Nothing Data Transfers

SCALI
Scalable Linux Systems



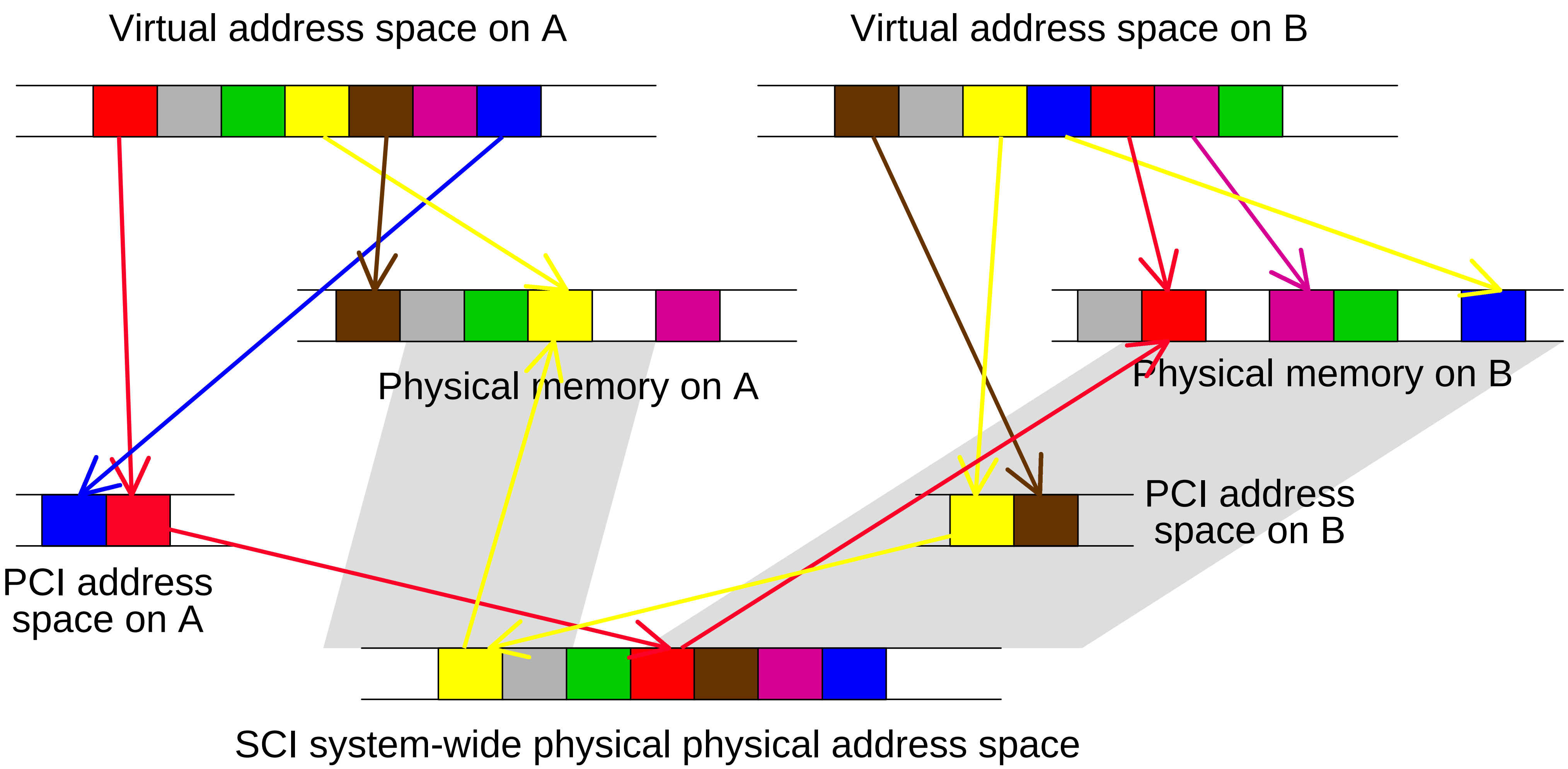
# Shared Address Space Architecture

Scalable Linux Systems



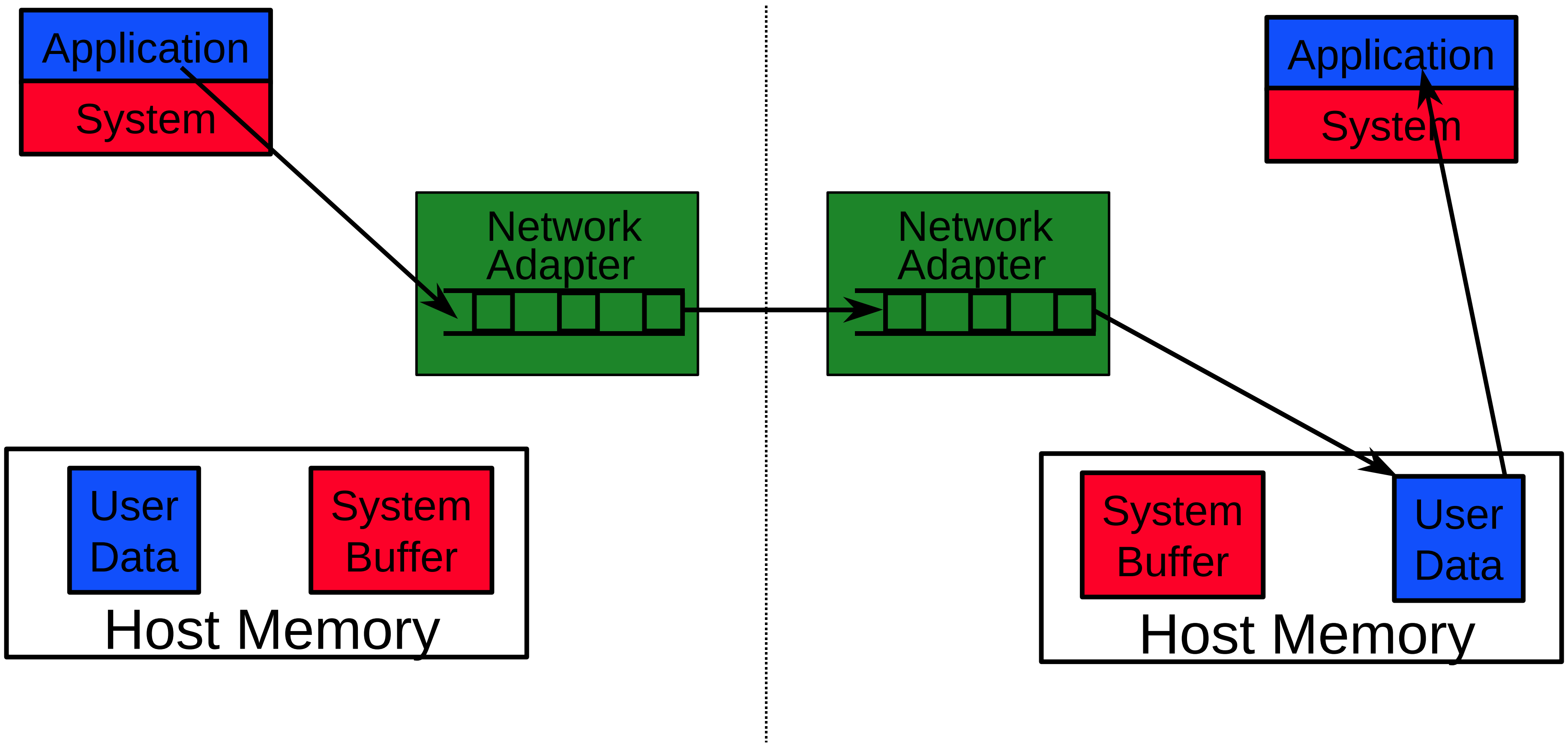

Memory Operations in a Packet Switched Network

# Shared Address Space from User Level



# Shared Address Space Data Transfers

SCALI
Scalable Linux Systems



# Atomic updates

SCALI
Scalable Linux Systems

- An update of a multi-byte entity is **atomic** if its side-effect is never made **partly** visible. That is, the update has either not (yet) occurred or it has already occurred.
- Memory consistency impacts the picture.
- Example (*p* points to a shared variable):

```
Producer:       for (i=0;; ++i) *p=i;
Consumer:       for (old=*p;;) if (old!=*p) {
                        printf("*p = %d\n", *p); old = *p;
                }
```

Result 1: 1,2,3,4,…,0xFE, 0xFF, 0x1FF, 0x100, 0x101, ...

Result 2: 1,2,3,4,…,0xFE, 0xFF, 0x000, 0x100, 0x101, ...

# Atomic updates (cont'd)

SCALI
Scalable Linux Systems

```
Typedef struct {
   char *buffer;
   int  valid;
} t_msg;
```

## Wrong:

```
Producer:   msg->buffer = source; msg.valid = TRUE;
Consumer: while (!msg->valid); consume(msg->buffer);
```

## Correct:

```
Producer:   msg->buffer = source; membar(); msg.valid = TRUE;
Consumer: while (!msg->valid); consume(msg->buffer);
```

# Idempotent Datastucture

SCALI
Scalable Linux Systems

- **A datastructure is idempotent if it is consistent after at least one update, as opposed to only one update**
- **Consumer data structures are write-only, it is disjunct wrt. write (i.e. the consumer does not update it, and is private to one producer**
- **Important in situations where a remote update might give failure indication and has to be re-issued**

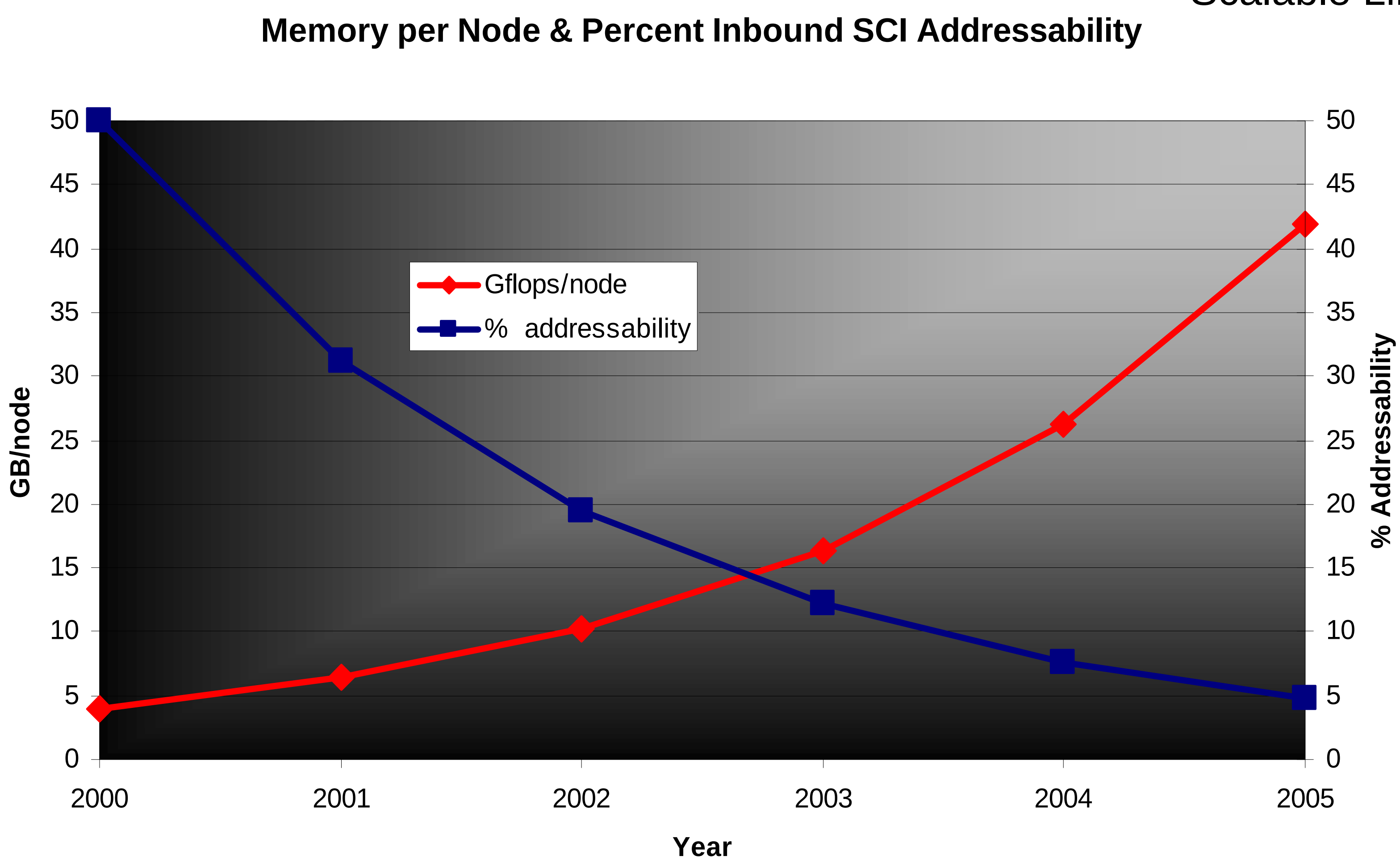
# Scalability issues of Shared Address Space Communication

SCALI
Scalable Linux Systems

- **Ideally, one like zero-copy methodology**
- **However, input addressability of current generation Dolphin PCI/SCI adapters is limited to 2GB**
  - **1 byte per flops rule**
  - **Today, close to 2Gflops/CPU ⇒ 4Gflops/node**
  - **FP performance increasing ~60% per year (Moore's law)**
  - **… and don't forget locality of user level pages**

# Scalability issues of Shared Address Space Communication

SCALI
Scalable Linux Systems



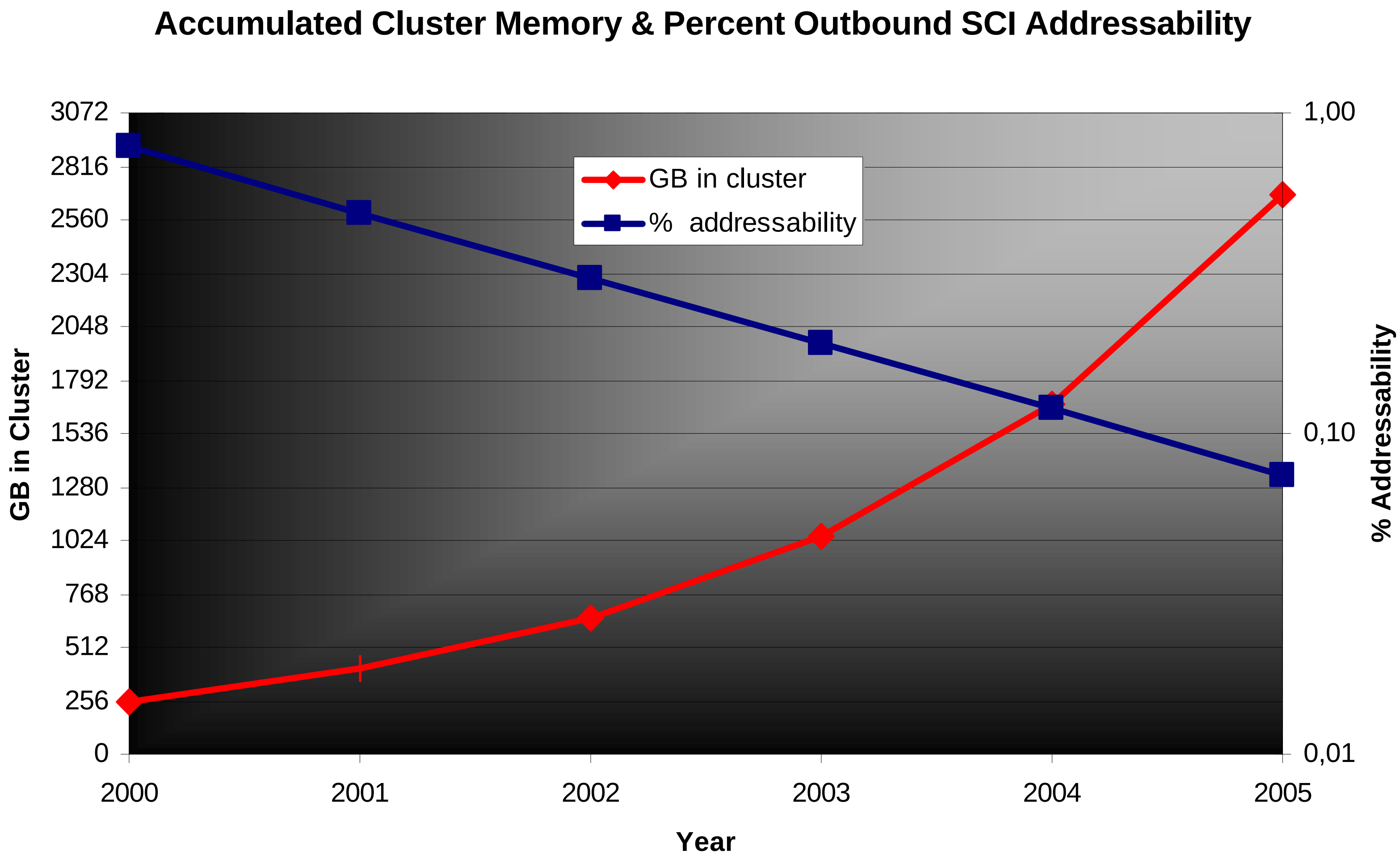
# Scalability issues of Shared Address Space Comm. (cont'd)

SCALI
Scalable Linux Systems

- **Outbound addressing is an even more severe problem:**
  - **PCI chip-sets have no demand for supporting large address space PCI targets, and will not get it in the foreseeable future**
  - **Hence, we are limited to max. 2GB outbound addressing**
  - **64 nodes, else same as previous example:**

# Scalability issues of Shared Address Space Comm. (cont'd)

SCALI
Scalable Linux Systems



# Scalability issues of Shared Address Space Comm. (cont'd)

SCALI
Scalable Linux Systems

- **Zero-copy, Remote Memory Access**
  - **associated with severe, over time increasing, limitations**
- **Alternatives:**
  - **Use DMA**
    - **No direct user-to-user level communication**
    - **Has the SCI architecture in general and Dolphin's products specifically an edge here?**
  - **Hybrid solution, i.e. both DMA and RMA**
    - **Good for specific problems, for example DSM**
  - **Develop a new host adapter architecture using *residual address control*. Example, Cray E-register file used in T3{DE}**

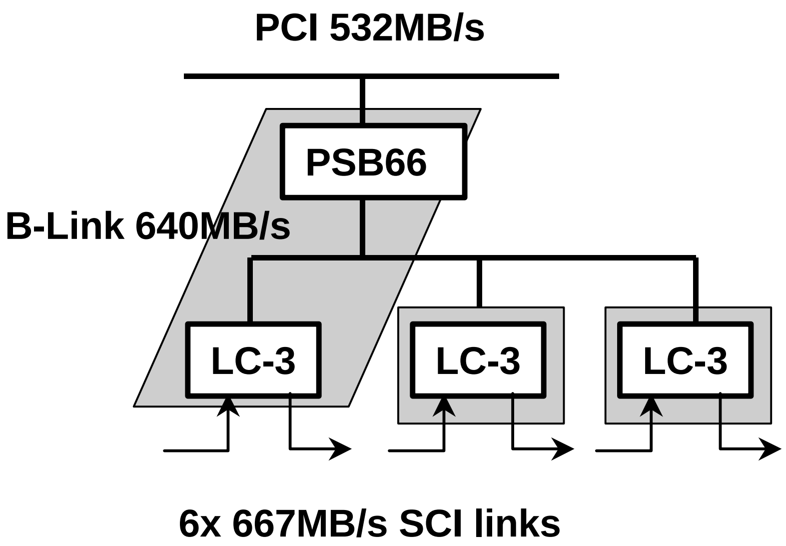
# Outline

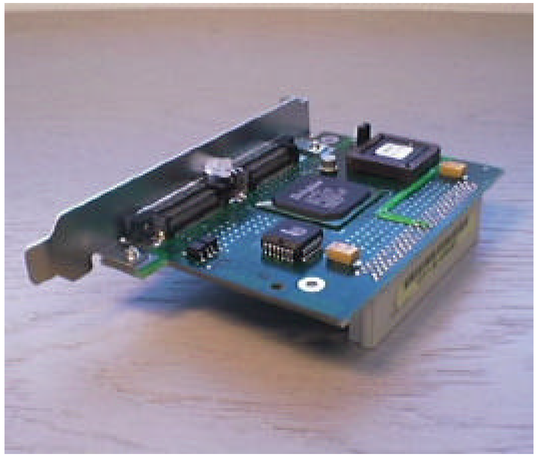
Scalable Linux Systems

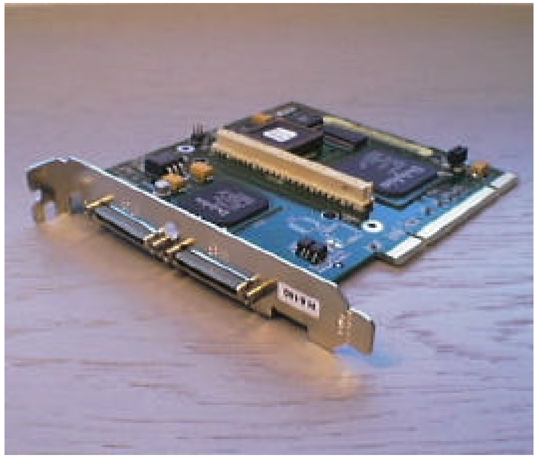
- ✓ Who's Scali?
- ✓ Scalability issues with Shared Address Space cluster architectures
- ✓ **Cons and Pros of a direct SCI network**
- ✓ Fault tolerant routing in a 2D SCI Torus
- ✓ Low level SCI programming using ScaMPI
- ✓ Node level parallelism. Would that be pthreads, OpenMP, or MPI?
- ✓ Cluster Management through Scali's *Universe*

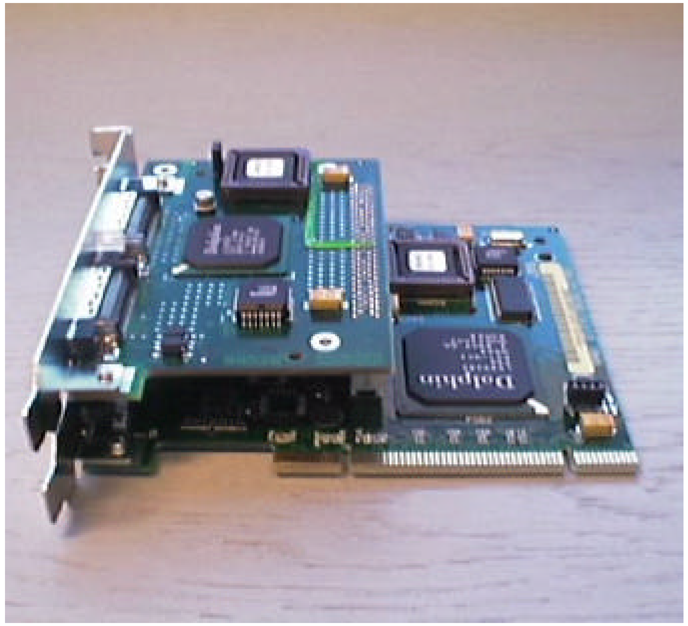
# 2D/3D Torus (D33X)

Scalable Linux Systems



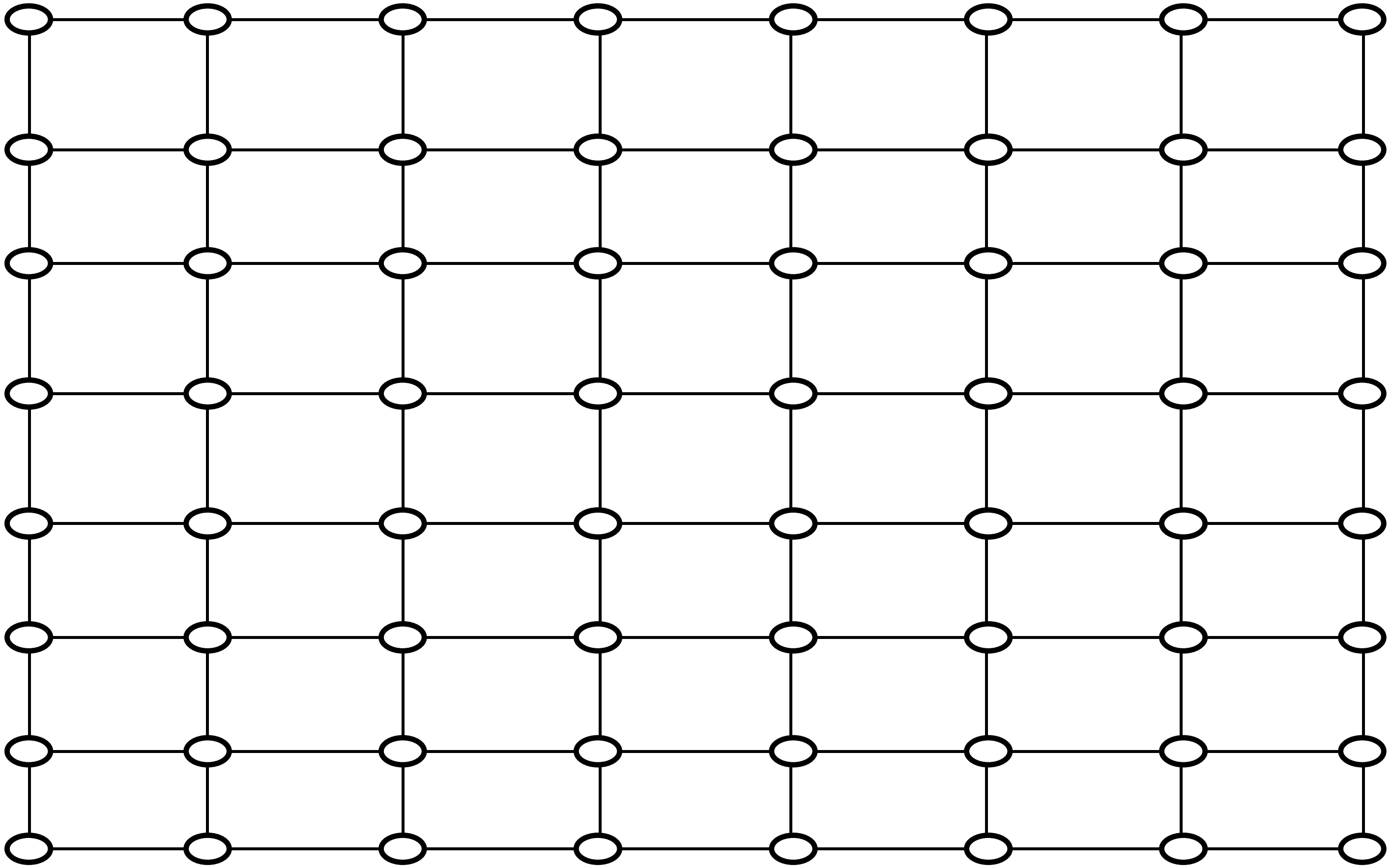



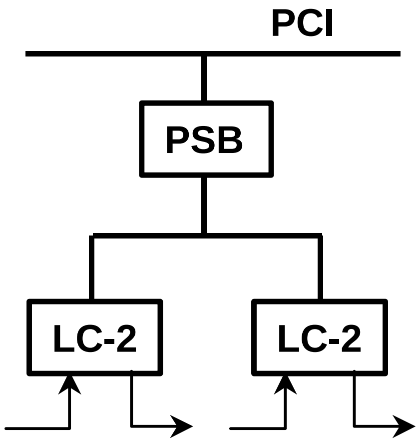




# 2D-Torus (64 nodes)

SCALI
Scalable Linux Systems

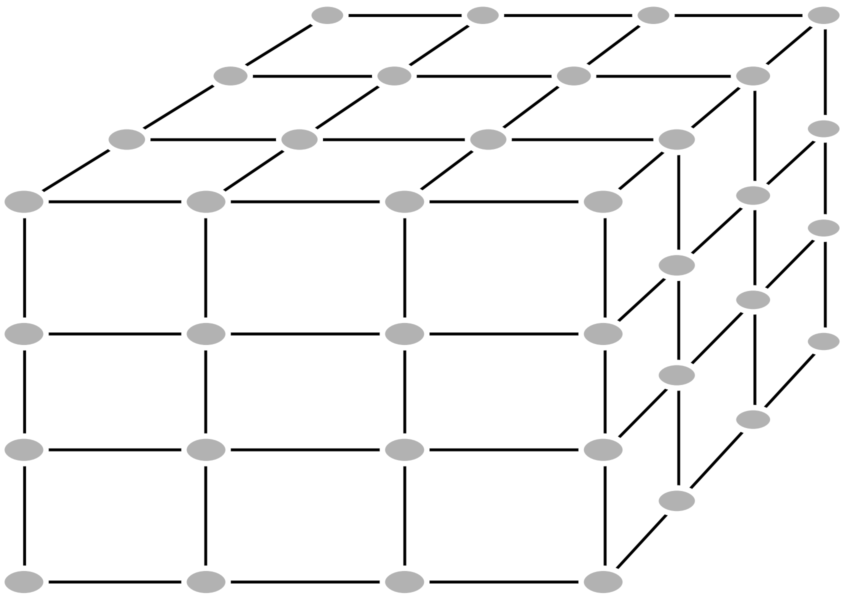


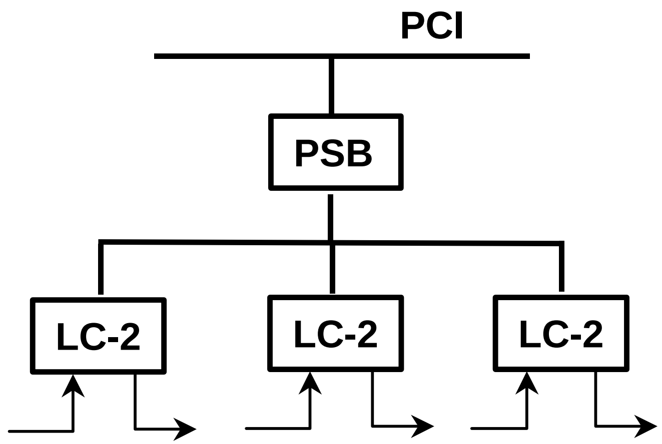



**Bi-section bandwidth: 14Gbyte/s**
**Longest Latency: 1.85 μsec**

# 3D-Torus, *4-ary 3-cube (64 nodes)*

SCALI
Scalable Linux Systems



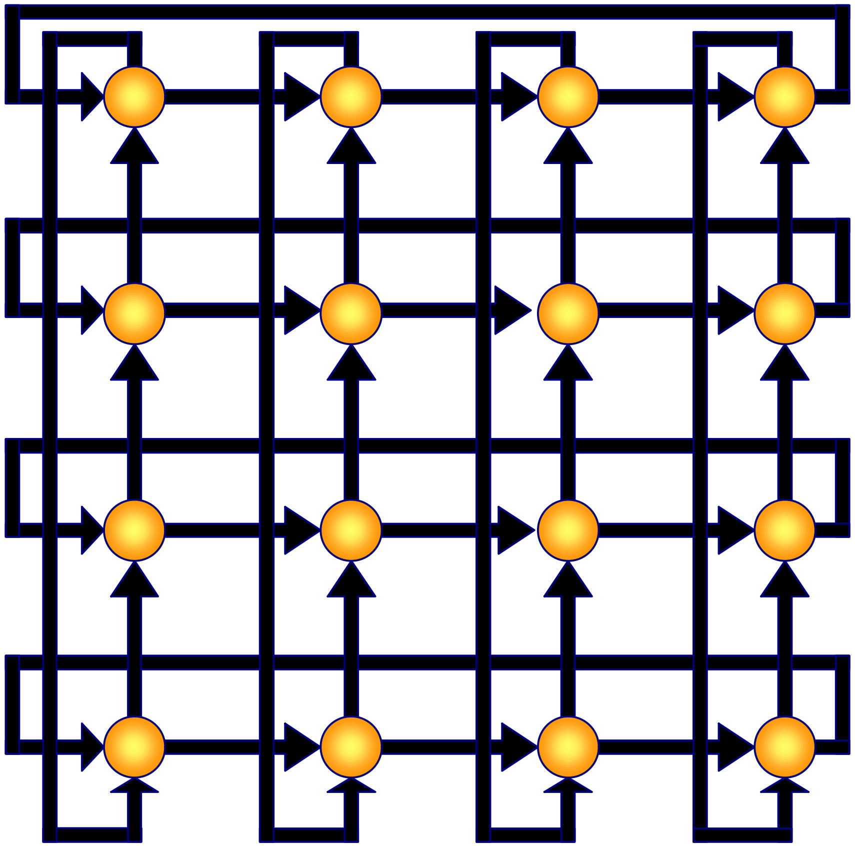



**Bi-section bandwidth: 24Gbyte/s**
**Longest Latency: 2.3 μsec**

# Switch-less topology

SCALI
Scalable Linux Systems

- **Distributed switching**
  - No single point of failure
  - Automatic re-routing
  - Simplified logistics
- **Low latencies**
  - Each node has direct access to the network
- **Cost-effective usage of excess SCI bandwidth vs. PCI bandwidth**



# Outline

SCALI
Scalable Linux Systems

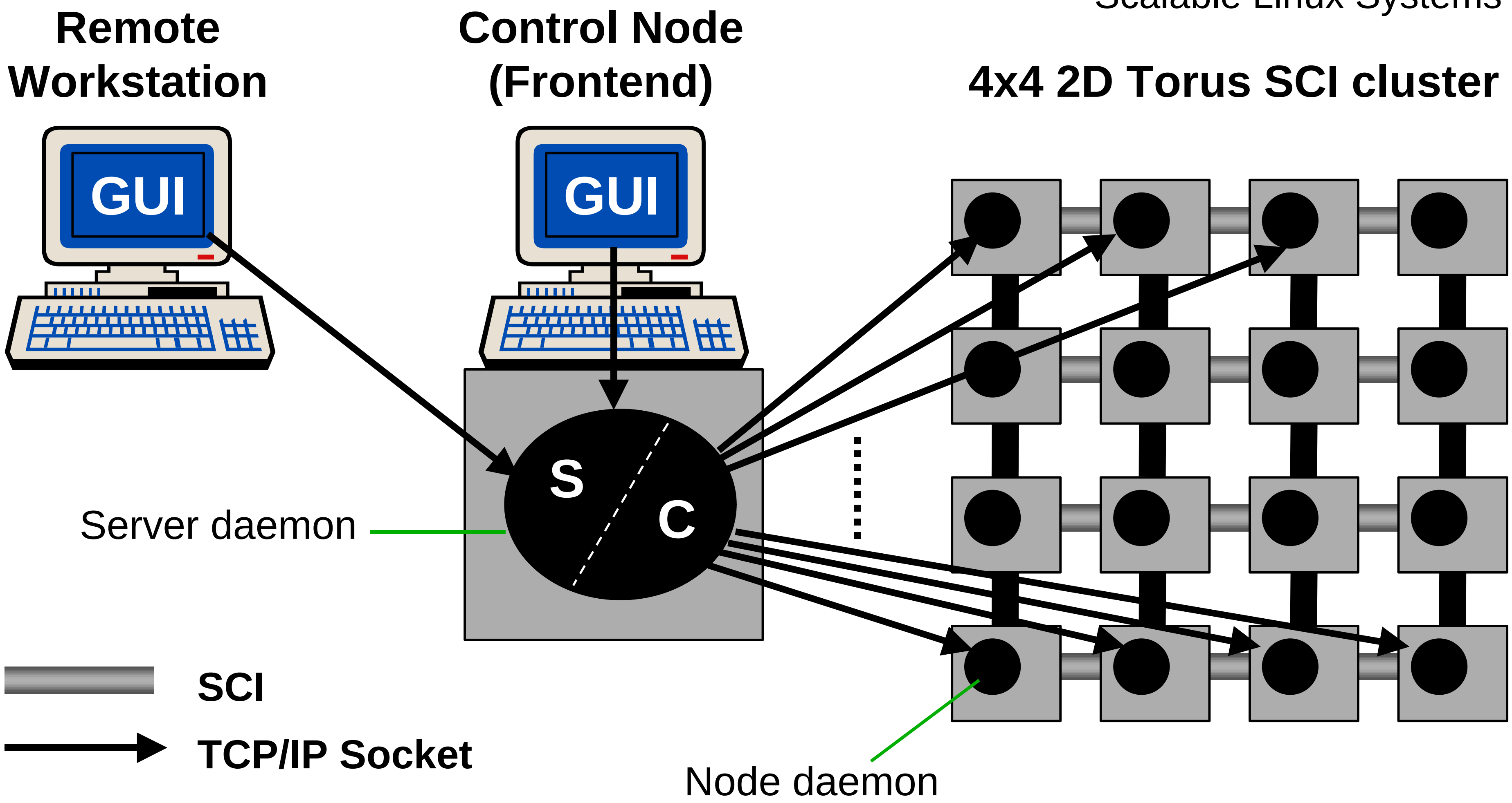
- ✓ Who's Scali?
- ✓ Scalability issues with Shared Address Space cluster architectures
- ✓ Cons and Pros of a direct SCI network
- ✓ **Fault tolerant routing in a 2D SCI Torus**
- ✓ Low level SCI programming using ScaMPI
- ✓ Node level parallelism. Would that be pthreads, OpenMP, or MPI?
- ✓ Cluster Management through Scali's *Universe*

# Scali Configuration System (Universe)

SCALI
Scalable Linux Systems

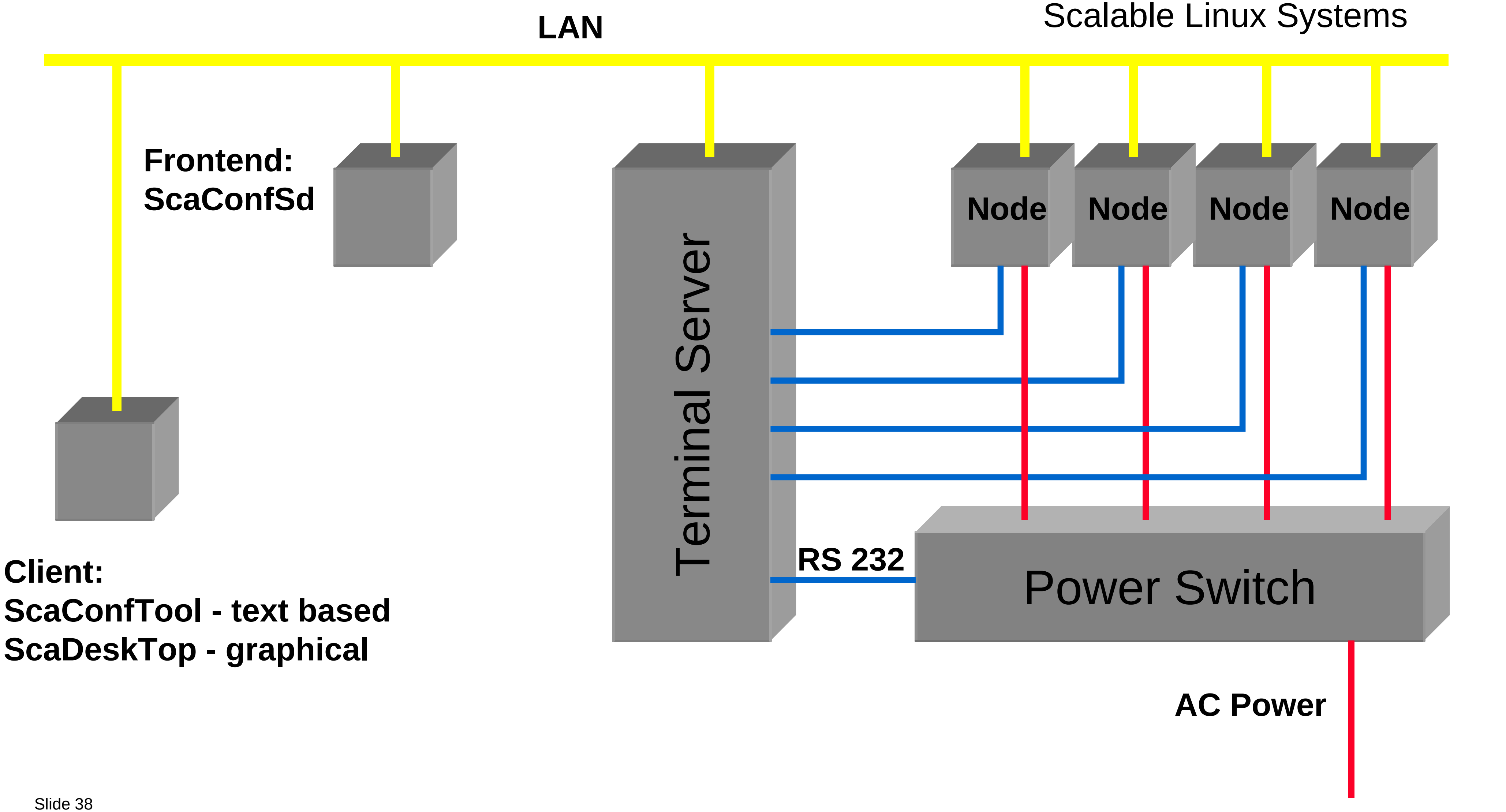
- Single point for:
  - System configuration
  - System management
  - System observability
  - Software installation
  - Software update
- Heterogeneous systems:
  - Operating Systems
  - HW Architecture
- Manages:
  - Nodes
  - Console ports
  - Power switches
  - Interconnect
- Uses:
  - SNMP
  - rsh/ssh
  - telnet
  - ScaSH

# *Universe*: System Architecture



# *Universe*: Physical Connectivity

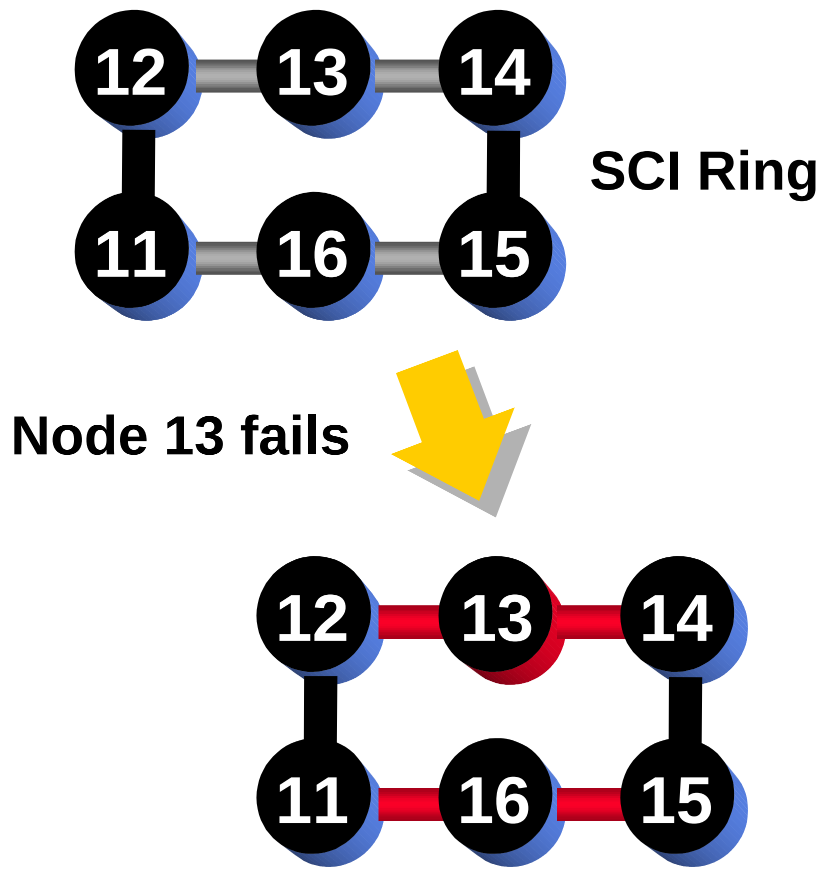
SCALI
Scalable Linux Systems



# *Universe*: Fault Tolerance

SCALI
Scalable Linux Systems

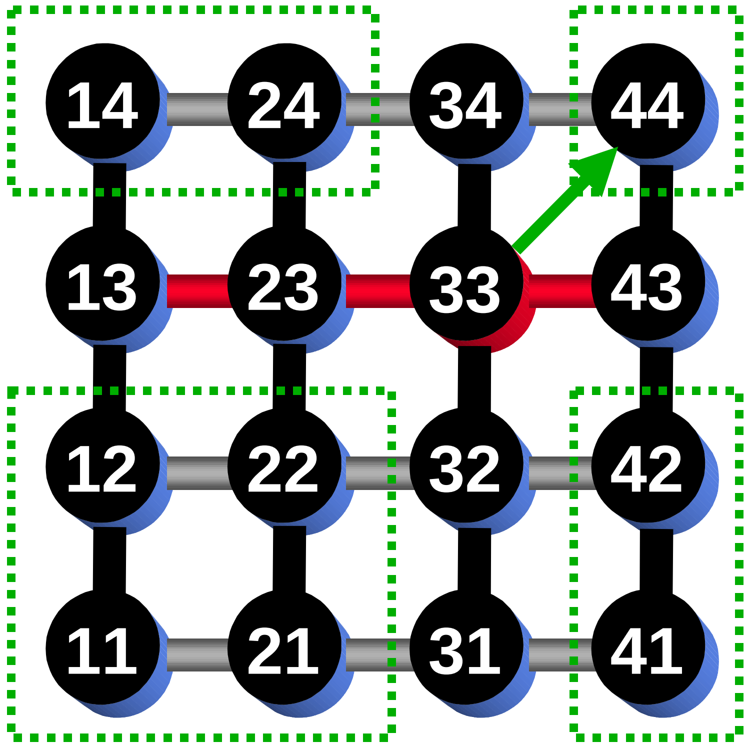
- Graceful degradation
  - Maximises connectivity of *alive* nodes
  - *Partitions* the system if necessary
- Fail State Categories
  - Reachable (1)
  - Unreachable (2)
  - Power Off (3)
- Single Ring Topology
  - Limited routing options



# *Universe*: Fault Tolerance

SCALI
Scalable Linux Systems

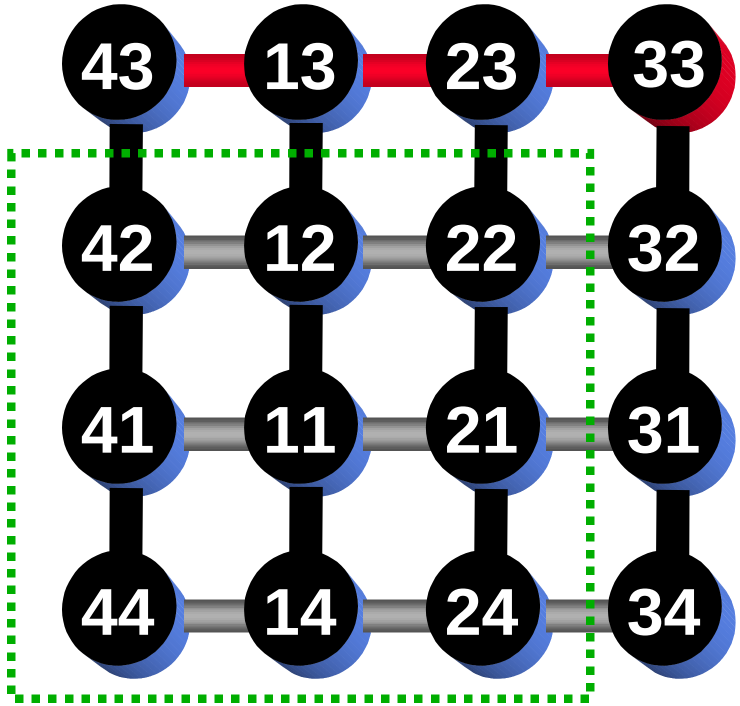
- **2D Torus topology**
  - **more routing options**
- **XY routing algorithm example:**
  - **Node 33 fails (3)**
  - **Nodes on 33's ringlets becomes unavailable**
  - **Cluster fractured with current routing setting**



# *Universe*: Fault Tolerance

SCALI
Scalable Linux Systems

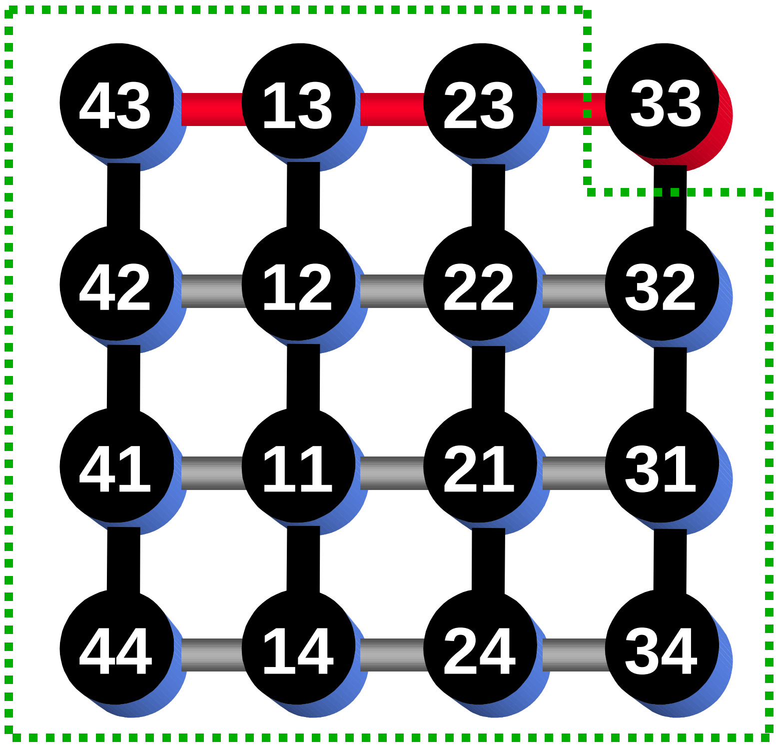
- **Rerouting with XY**
  - Failed node logically remapped to a corner
  - End-point NodeID's unchanged
  - Applications can continue
- **Problem:**
  - To many working nodes unused



# *Universe*:
# Fault Tolerance

SCALI
Scalable Linux Systems

- **Solution: Apply the advanced algorithm *“Scali Routing”***
  - **Scali routing maintains connectivity between all nodes with access to just one working ringlet**
- **All nodes but the failed one can be utilised as one big partition**
- **Exploits the *register-insertion-ring* property of SCI, i.e buffer dependency graph does <u>not</u> contained the bypassed nodes**
- **Calculation of optimum routing tables is handled by *ScaConfSd* automatically**



# Outline

Scalable Linux Systems

- ✓ Who's Scali?
- ✓ Scalability issues with Shared Address Space cluster architectures
- ✓ Cons and Pros of a direct SCI network
- ✓ Fault tolerant routing in a 2D SCI Torus
- ✓ **Low level SCI programming using ScaMPI**
- ✓ Node level parallelism. Would that be pthreads, OpenMP, or MPI?
- ✓ Cluster Management through Scali's *Universe*

# Low-level SCI programming using ScaMPI

SCALI
Scalable Linux Systems

- **ScaMPI has a lot of useful features:**
  - **Launching of applications**
  - **Abstraction of SCI *nodeIds***
  - **Debugging windows (gdb, TotalView or other)**
  - **Manual launch windows (strace, ltrace, LD_LIBRARY_PATH etc.)**
  - ***stdin* redirection, collecting *std{out,err}***
- **MPI has a rich set of features:**
  - **Point-to-Point communication**
  - **Communicators**
  - **Collective operations**
  - **MPI_Barrier()**
  - **MPI_Wtime()**

# Low-level SCI programming using ScaMPI (cont'd)

SCALI
Scalable Linux Systems

**These features can be combined with SCI level programming, through Scali's extension to MPI:**

```
void * p; int me; unsigned sz;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &me);

if (me) {
    p  = PMPI_TbInitRead(MPI_COMM_WORLD, 0);
    sz = PMPI_TbGetSizeRead(MPI_COMM_WORLD, 0);
} else {
    p  = PMPI_TbInitWrite(MPI_COMM_WORLD, 1);
    sz = PMPI_TbGetSizeWrite(MPI_COMM_WORLD, 1);
}
```

# Outline

SCALI
Scalable Linux Systems

- ✓ Who's Scali?
- ✓ Scalability issues with Shared Address Space cluster architectures
- ✓ Cons and Pros of a direct SCI network
- ✓ Fault tolerant routing in a 2D SCI Torus
- ✓ Low level SCI programming using ScaMPI
- ✓ **Node level parallelism. Would that be pthreads, OpenMP, or MPI?**
- ✓ Cluster Management through Scali's *Universe*

# Node level parallelism

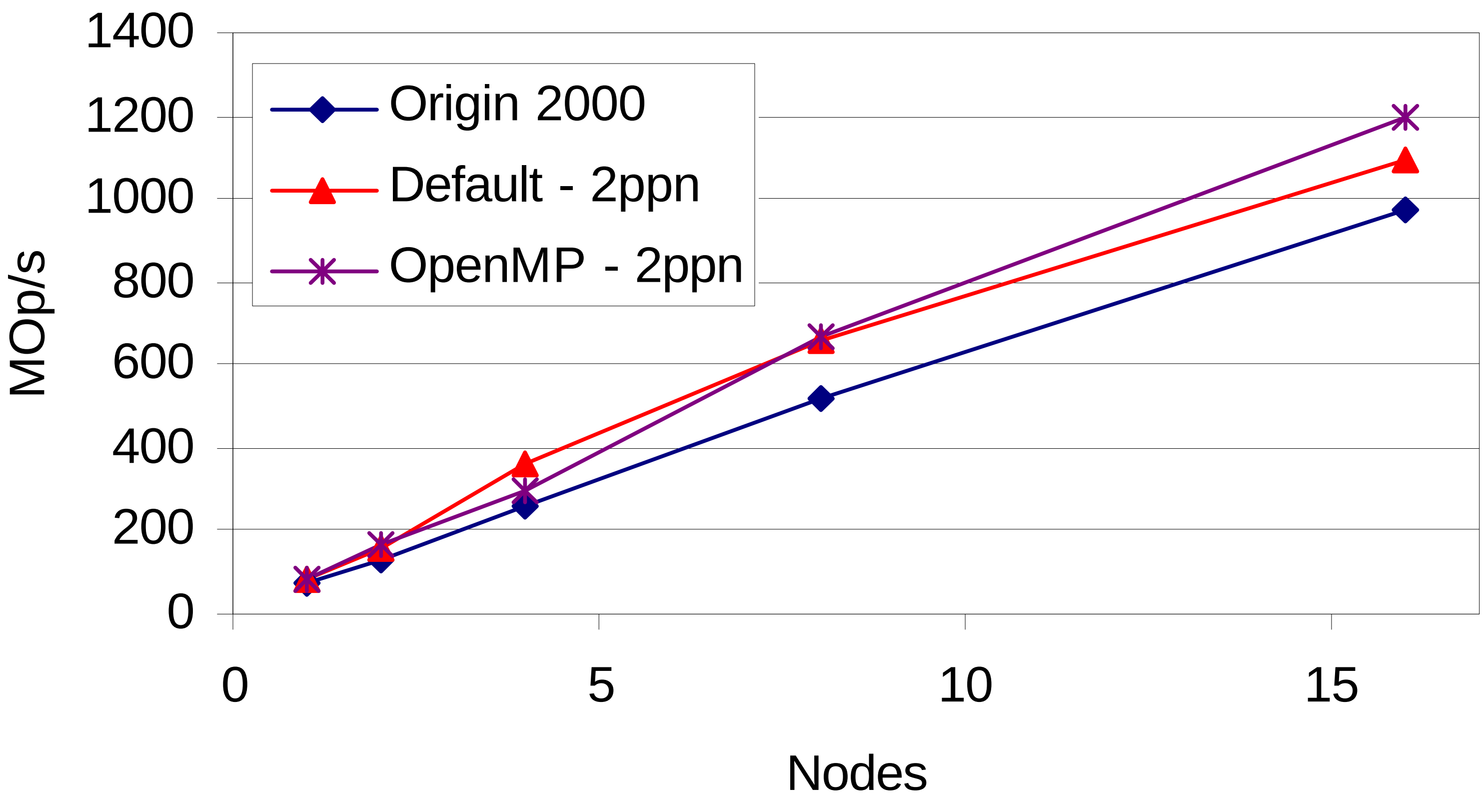
SCALI
Scalable Linux Systems

- **Straight or 1:1**
  - **launch one MPI process per CPU in the system**
- **SMP-ish or 1:N**
  - **Utilize OpenMP on the node level**
  - **Use multitreaded libraries (e.g. ATLAS BLAS, NAG, etc.)**
  - **Use PTHREADS**

# Node level parallelism (cont'd)

SCALI
Scalable Linux Systems

- **MG is a simplified multigrid kernel.**
- **MG uses highly structured long distance communication**



# Node level parallelism

SCALI
Scalable Linux Systems

- **Examples using the 1:1 model:**
  - **CCM3 - Atmospheric Simulation (NCAR)**
  - **DALTON - Quantum Chemistry (UiO)**
  - **RADYN - Astro Physics (UiO)**

| | Elapsed (secs) | | |
|---|---|---|---|
| Benchmark | 2 nodes, 1 CPU per node | 1 node, 2 CPUs | Ratio |
| CCM3 | 162,00 | 172,00 | 1,06 |
| DALTON | 4266,07 | 4124,46 | 0,97 |
| RADYN | 59,53 | 59,83 | 1,01 |

# Outline

Scalable Linux Systems

- ✓ Who’s Scali?
- ✓ Scalability issues with Shared Address Space cluster architectures
- ✓ Cons and Pros of a direct SCI network
- ✓ Fault tolerant routing in a 2D SCI Torus
- ✓ Low level SCI programming using ScaMPI
- ✓ Node level parallelism. Would that be pthreads, OpenMP, or MPI?
- ✓ **Cluster Management through Scali's *Universe***