Concurrent Computing Laboratory
for Materials Simulations

INTRODUCTION TO PARALLEL COMPUTING

AIICHIRO NAKANO
Department of Computer Science
Concurrent Computing Laboratory for Materials Simulations
Louisiana State University - Baton Rouge

NSF Supported Workshop to Enhance Minority Undergraduate Faculty Education in Robotics and Machine Vision
June 19-30, Baton Rouge, LA

Louisiana State University

OUTLINE

- Introduction
  - > Parallel applications
  - > Parallel architectures
  - > Parallel programming styles

- Experience on various platforms
  - > MasPar SIMD architecture
  - > Intel iPSC/860 distributed-memory MIMD architecture
  - > PVM on a Digital Alpha farm
  - > Intel iWarp systolic array

LEARNING PARALLEL COMPUTING ON WORLD WIDE WEB

- **On-line book**
  "Designing and Building Parallel Programs"
  by Ian Foster
  `http://www.mcs.anl.gov/dbpp/`

- **e-book**
  "Computational Science Education Project"
  `http://csep1.phy.ornl.gov/csep.html`

COMPUTATIONAL SYNERGETICS

- Synergetics between analytical & computational methods in nonlinear, complex problems
  "High-speed computing devices may provide us with those heuristic hints which are needed in all parts of mathematics for genuine progress. "
  (John von Neumann)

- Emerging Grand-Challenge applications
  > Computer-aided materials design
  > Computational electronics
  > Computational fluid dynamics

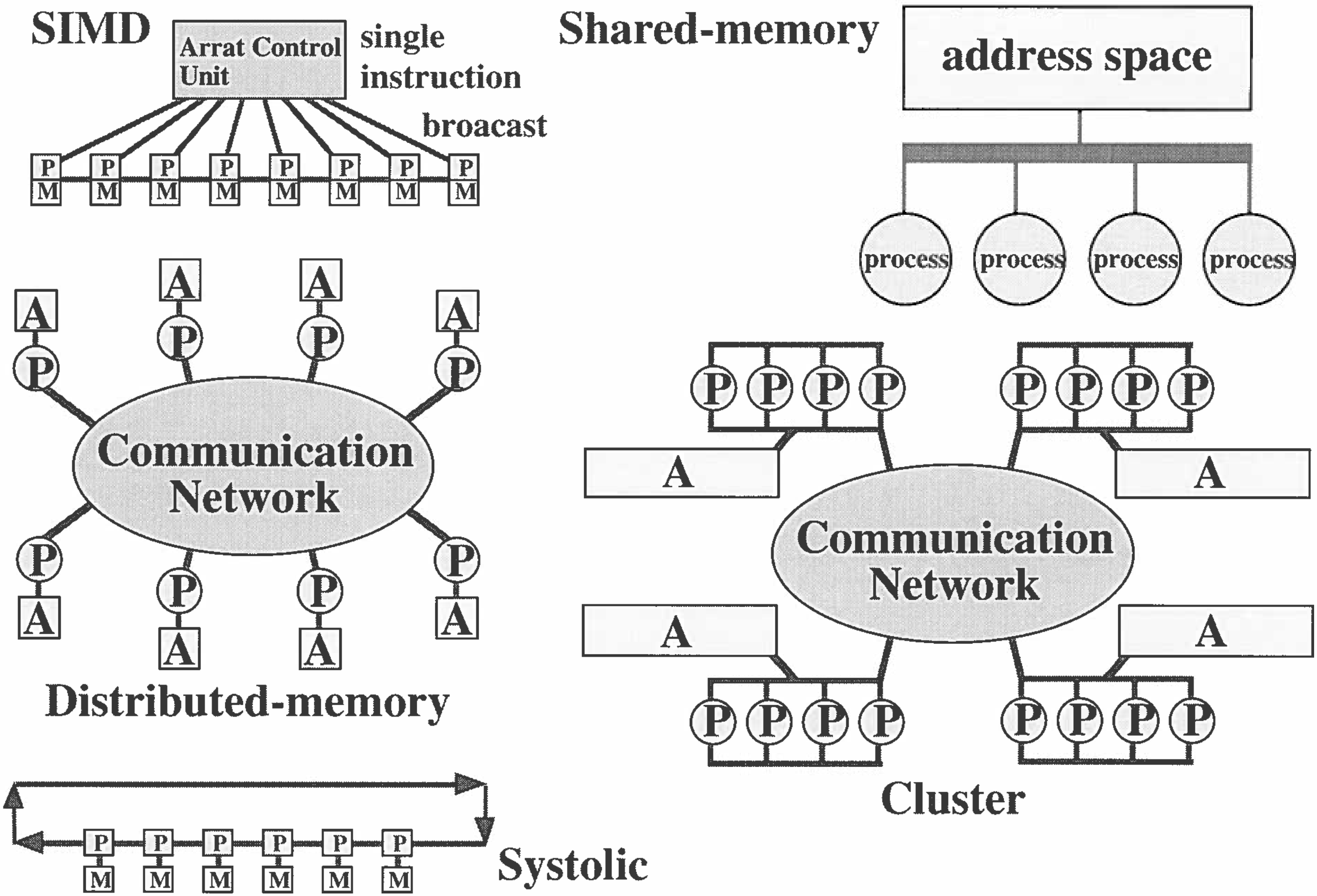
PARALLEL APPLICATIONS

- **SYNCHRONOUS**
  - > Finite difference simulations
  - > Field data type
  - > Static & regular
- **LOOSELY SYNCHRONOUS**
  - > Molecular dynamics
  - > Particle data type
  - > Dynamic & irregular
- **DOMAIN DECOMPOSITION PARALLELISM**

PARALLEL ARCHITECTURES

- **SIMD (single instruction multiple data)**
  - > MasPar
  - > Data parallelism; Static, regular data structures
- **Shared-memory MIMD (multiple instruction multiple data)**
  - > Silicon Graphics Power Center
  - > High-speed bus; Easy programming
- **Distributed-memory MIMD**
  - > Intel iPSC/860
  - > Data & function parallelisms; Dynamic, irregular problems
- **Workstation farm**
  - > Digital Alpha farm; IBM SP2
  - > Higher performance/cost ratio
- **Systolic architecture**
  - > Intel iWarp
  - > Special purpose: image processing

PARALLEL ARCHITECTURES



PARALLEL COMPUTING MODELS

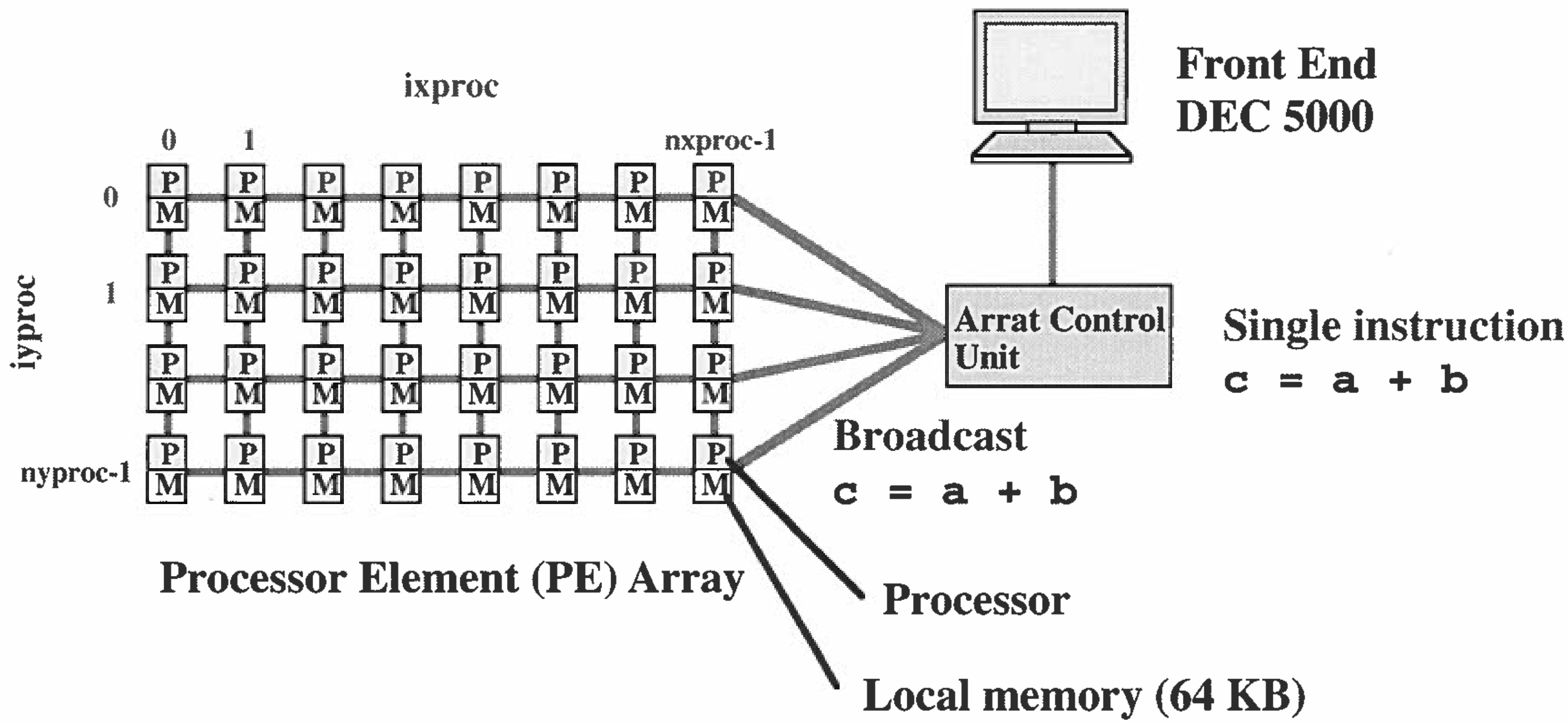
Workload Allocation
- Data parallelism
- Function parallelism
  - > Pipeline

Programming Style
- Data parallel programming
  - > High Performance Fortran
    - - Array operations; FORALL statement
- Message-passing programming
  - > Message Passing Interface

Computing Paradigms
- SPMD (single program multiple data) paradigm
- Master-slave paradigm
  - > Work pool

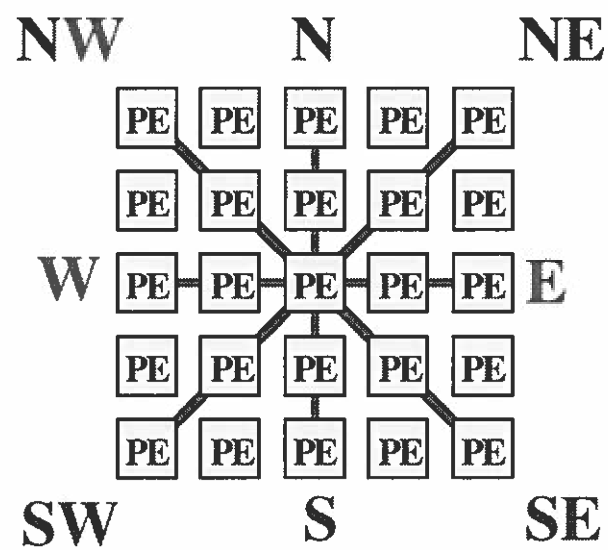
MASPAR ARCHITECTURE

- Single instruction multiple data (SIMD) architecture
- 2D mesh: $128 \times 64 = 8192$ nodes
- 750 MFlops (32 bit)
- XNet communication in the PE array - 12 GB/sec
- Global router: multistage crossbar switch



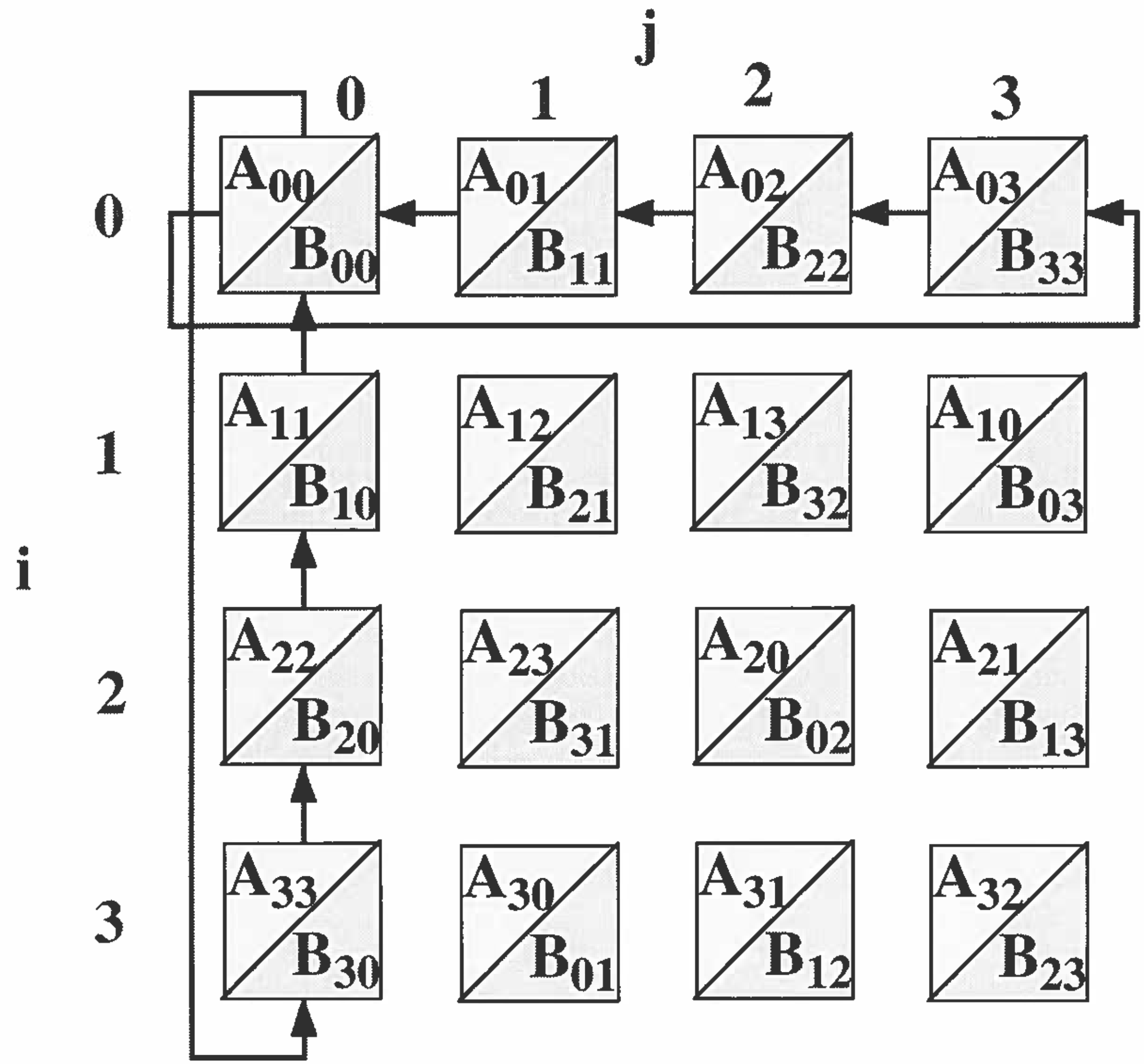
MPL PROGRAMMING

- **plural data type**
```
plural int a, b, c, d;
```
- **XNet constructs**
```
c = xnetE[1].a + xnetW[1].b;
```
- **Global operations**
```
int sum;
sum = reduceAdd32(a);
```
- **Plural indices (PE addresses)**
```
ixproc, iyproc
```
- **Active set**
```
if (a == 1) b = c + d;
```



SYSTOLIC MATRIX MULTIPLICATION

$$C(i,j) = \sum_{k=0}^{N-1} A(i, i+j+k \bmod N) \times B(i+j+k \bmod N, j)$$



RUNNING MPL APPLICATIONS

- **Log in to the DEC 5000 front end**

- **Compiling an MPL code**
  (Standalone MPL)

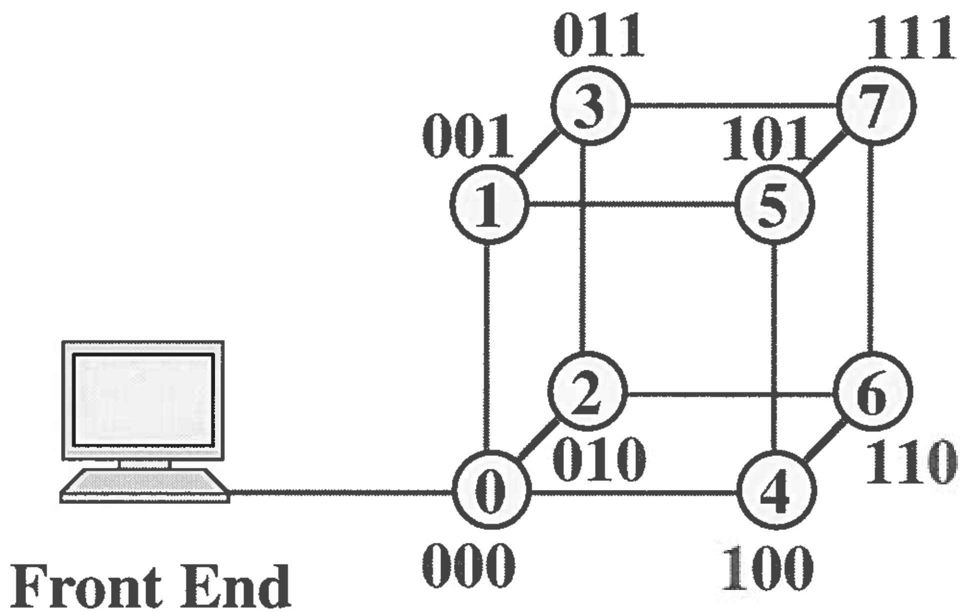
  ```
  hurricane> mpl_cc Matrix_mult.m
  ```

- **Running an MPL application**

  ```
  hurricane> mpl_cc Matrix_mult.m
  ```

iPSC/860 ARCHITECTURE

- Distributed-memory MIMD (multiple instruction multiple data) architecture
- Hypercube network: $2^3 = 8$ nodes
- 640 MFlops (32 bit)
- 2.5 MB/s node-to-node bandwidth



NX LIBRARY

- **Node ID**

```
long numnodes();
long mynode();
```

- **Point-to-point communication**

```
csend(long type, char *buffer, long length,
      long node, long process_id);
crecv(long type, char *buffer, long length);
```

- **Global operations**

```
gdsum(double *x, long n, double *work);
```

RUNNING NX APPLICATIONS

- **Log in to the front end**
- **Compiling a source program calling the NX library**
```
pearl> icc -o pi pi.c -node
```
- **Getting cube information**
```
pearl> cubeinfo -s
```
- **Getting a subcube**
```
pearl> getcube -c mycube -t4
```
- **Running an application on the subcube**
```
pearl> load pi
```
- **Releasing the subcube**
```
pearl> relcube -c mycube
```

NETWORK COMPUTING

- RISC (reduced instruction set computer) Processors
  > Speed demons (high clock rate)
    Digital Alpha (~ 300 MHz)
  > Brainiacs (superscalar)
    IBM Power2 (dual floating-point multiple-add pipelines)
- High-Speed Networks
  > FDDI (100 Mbit/sec)
  > HiPPI (800 Mbit/s or 1.6 Gbit/sec)
- Parallel Programming Systems
  > Message passing systems - PVM, MPI
  > Data parallel programming systems - HPF

DIGITAL ALPHA FARM

38 DEC 3000 Workstations

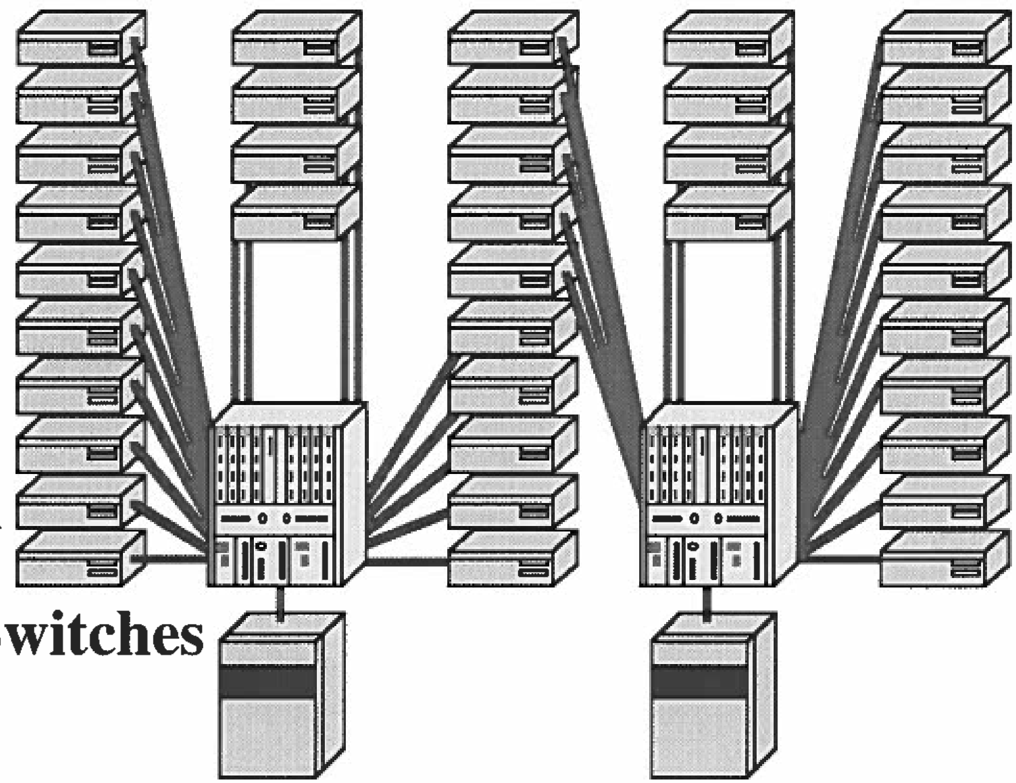
166MHz

128MB RAM

2GB disk

FDDI ports: 01ml.cclms.lsu.edu,...,38ml.cclms.lsu.edu

Ethernet: ml01.cclms.lsu.edu,...,ml38.cclms.lsu.edu



22×22 crossbar

12.5MB/s bandwidth

15μs latency

2 FDDI GIGASwitches

2 Alpha Servers 2100

mona.cclms.lsu.edu	lisa.cclms.lsu.edu
275MHz × 4	275MHz × 4
512MB RAM	1GB RAM
16GB disk	16GB disk

PARALLEL VIRTUAL MACHINE (PVM)

- **Heterogeneous distributed computing**
- **PVM components**
  - > **Daemon: process coordination, message routing**
  - > **Message-passing library**
- **Advanced features**
  - > **Dynamic group**
  - > **Global communications**
- **Implementation**
  - > **TCP-IP**
- **Oak Ridge National Lab.**
  - > **Anonymous ftp**

    `netlib2.cs.utk.edu`
  - > **World wide web**

    `http://www.netlib.org/pvm3/index.html`

PVM BASICS

- **PVM console**

```
ml01.cclms.lsu.edu> pvm
pvm> add ml02.cclms.lsu.edu
pvm> conf
2 hosts, 1 data format
                    HOST      DTID      ARCH    SPEED
      ml01.cclms.lsu.edu     40000     ALPHA     1000
      ml01.cclms.lsu.edu     80000     ALPHA     1000

pvm> quit
```

- **PVM task ID**

```
task_id = pvm_mytid(void);
```

- **Spawning PVM tasks**

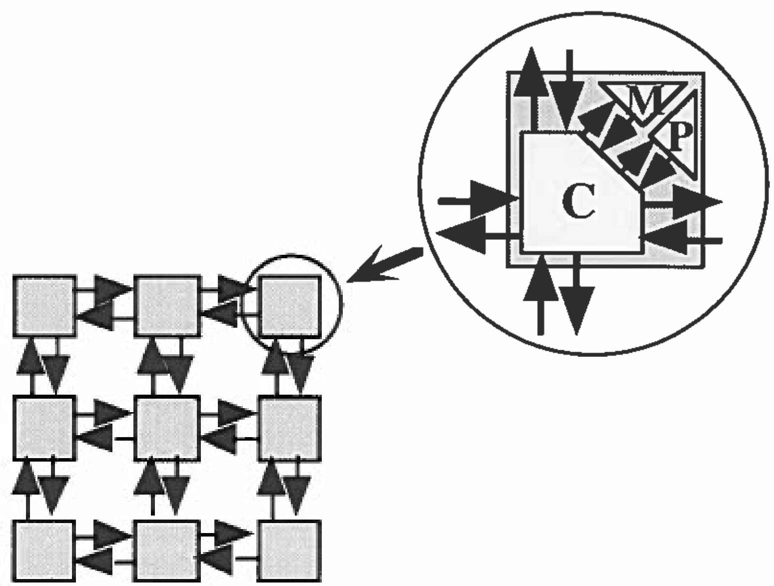
```
num_tasks = pvm_spawn(task, argv, flag, where, ntask, tids);
```

- **Point-to-point communication**

```
info = pvm_psend(task_id,message_tag,buffer,length,datatype);
info = pvm_precv(task_id,message_tag,buffer,length,datatype,
       &source,&actual_tag,&actual_length);
```

INTEL iWarp ARCHITECTURE

- Systolic array: Data is pumped through a network of processors
- 2D mesh: $8 \times 8 = 64$ cells
- iWarp cell
  > Computation agent: 20MFlops (32 bit)
  > Communication agent: 320 MB/s bandwidth, 100ns latency
  > Memory unit: 2 MB



PathLib PROGRAMMING

- **Cell ID**

```
struct iwcfg cfg;
getcfg(&cfg);
myid = cfg.cellid;
```

- **Logical channel**

```
pl_rpe_configure(0x1, 0, 0x2, 0, 0xC);
```

- **Open a connection**

```
o_chan = pl_send_oc(PL_GATE0,PL_DIR_XL,route_info, route_len);
i_chan = pl_send_oc(PL_GATE0,&header);
```

- **Send & receive a message header**

```
pl_send_om(PL_GATE0, PL_NOMATCH);
pl_recv_om(PL_GATE0, &message_header);
```

- **Send & receive data**

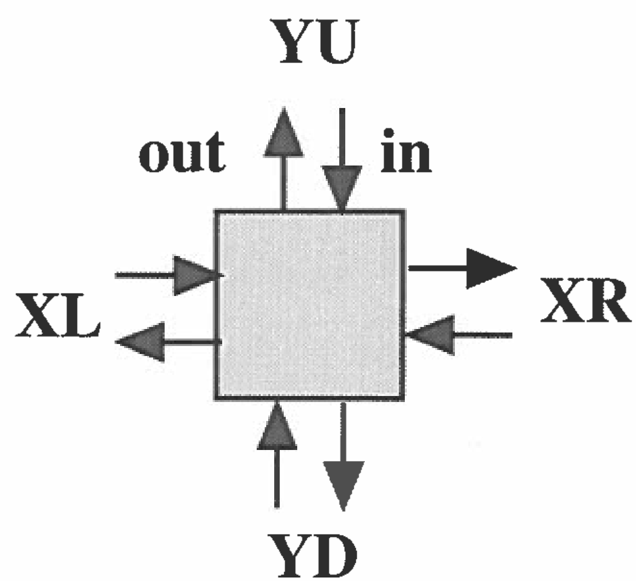
```
pl_sendi(PL_GATE0, a);
c += pl_recvi(PL_GATE0);
```

- **Send & receive a message trailer**

```
pl_send_cm(); pl_recv_cm();
```

- **Close a connection**

```
pl_send_cc(); pl_recv_cc();
```



RUNNING iWarp APPLICATIONS

- **Log in to the front end**
- **Compiling a source code**

  ```
  sunmp> iwcc -o mat2d mat2d.c -lpl -lrts
  ```
- **Executing an application**

  ```
  sunmp> lgo -r mat2d 0..63
  ```

MESSAGE PASSING INTERFACE (MPI)

- Message passing library
- User-defined message types
- Message contexts & process groups
- Collective communications

HIGH PERFORMANCE FORTRAN (HPF)

- **Array operations**
- **Data-parallel constructs**
  > FORALL statement
- **Data allignment directives**
- **Virtual processor array**