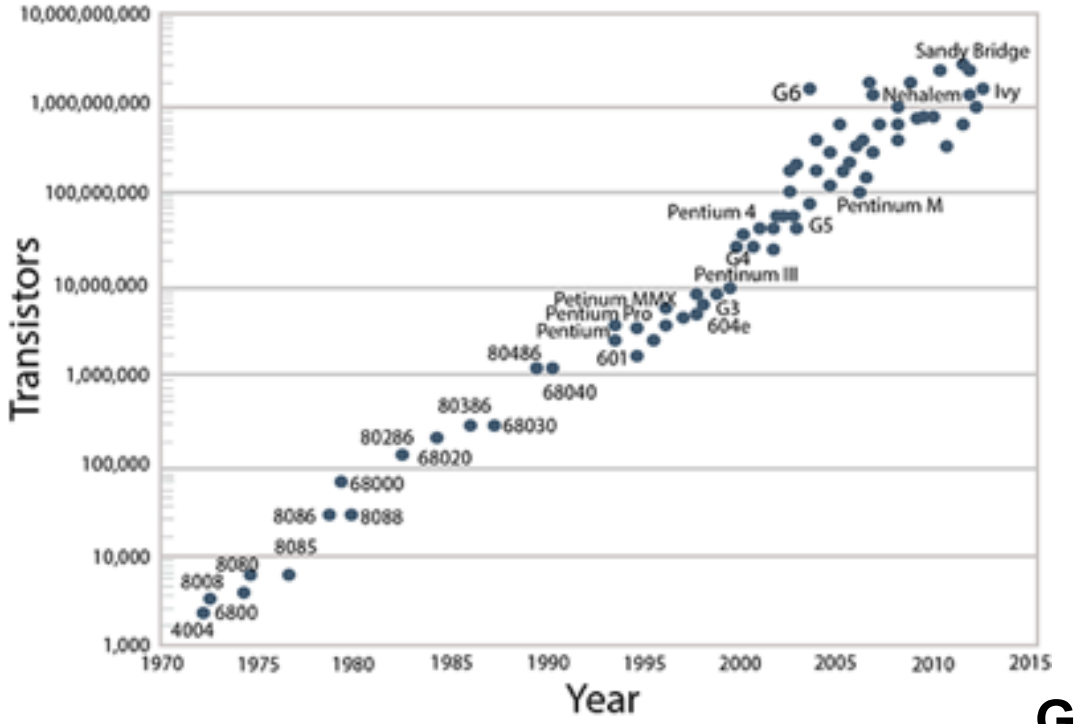
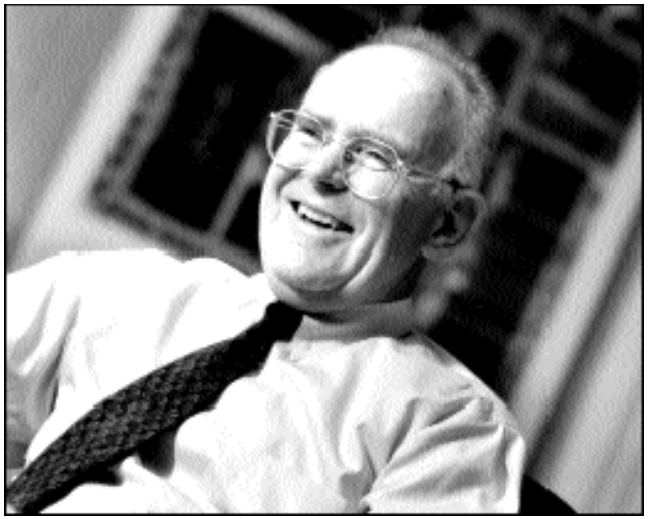
CS 140:

Models of parallel programming:

Distributed memory and MPI

Technology Trends: Microprocessor Capacity



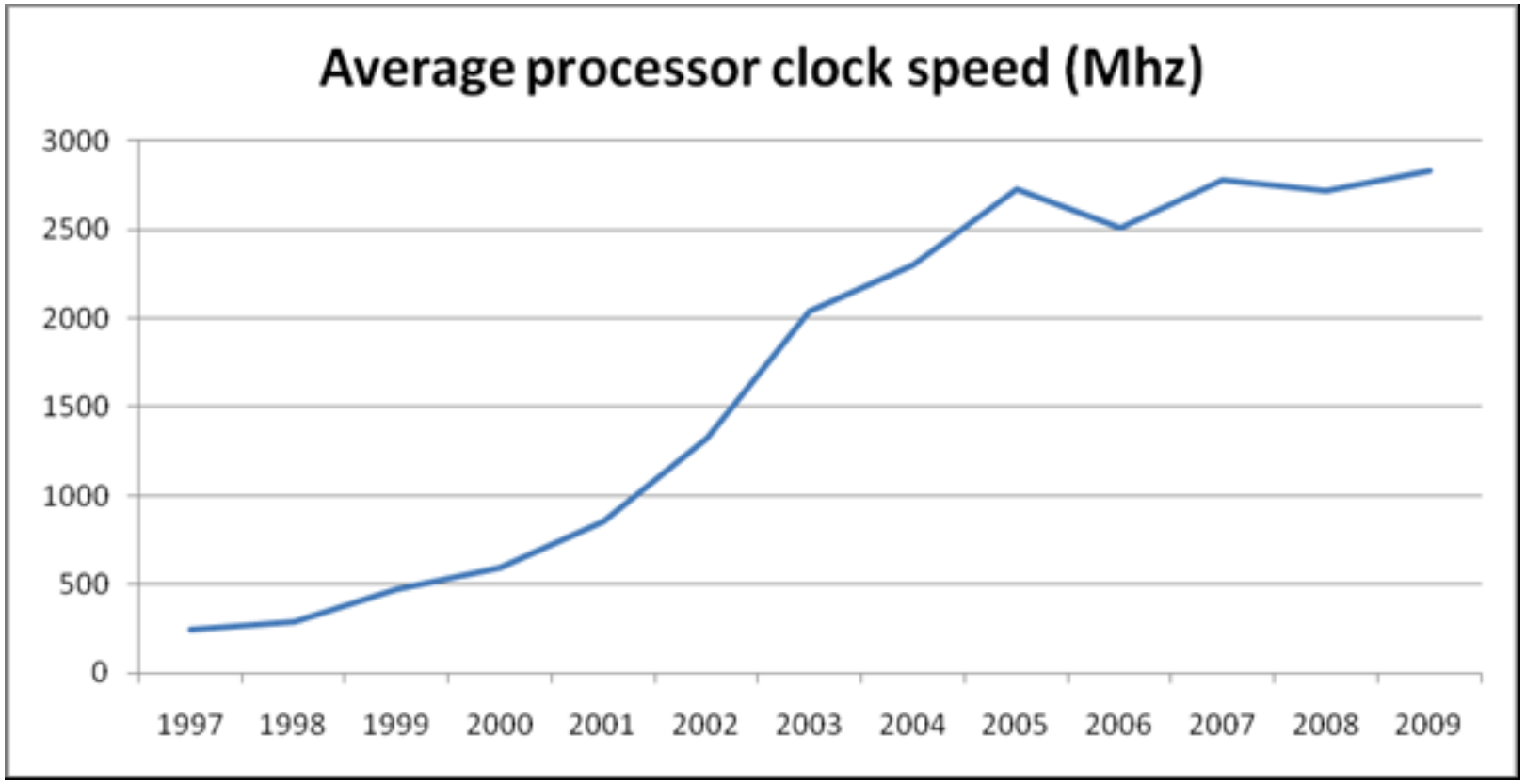


Moore's Law: # transistors / chip doubles every 1.5 years

Gordon Moore (Intel co-founder) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

Microprocessors keep getting smaller, denser, and more powerful.

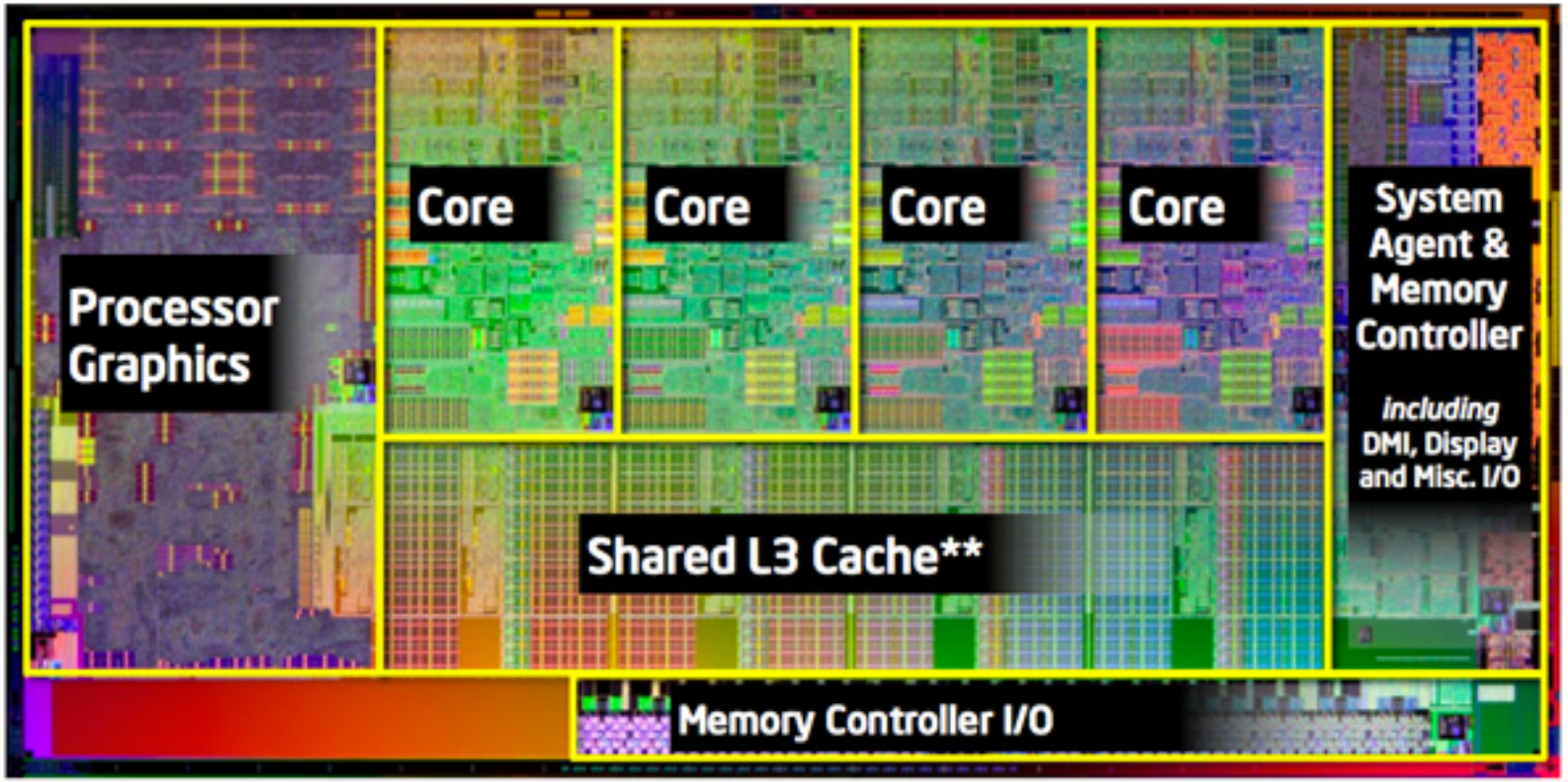
Trends in processor clock speed



Triton's clockspeed is still only 2600 Mhz in 2015!

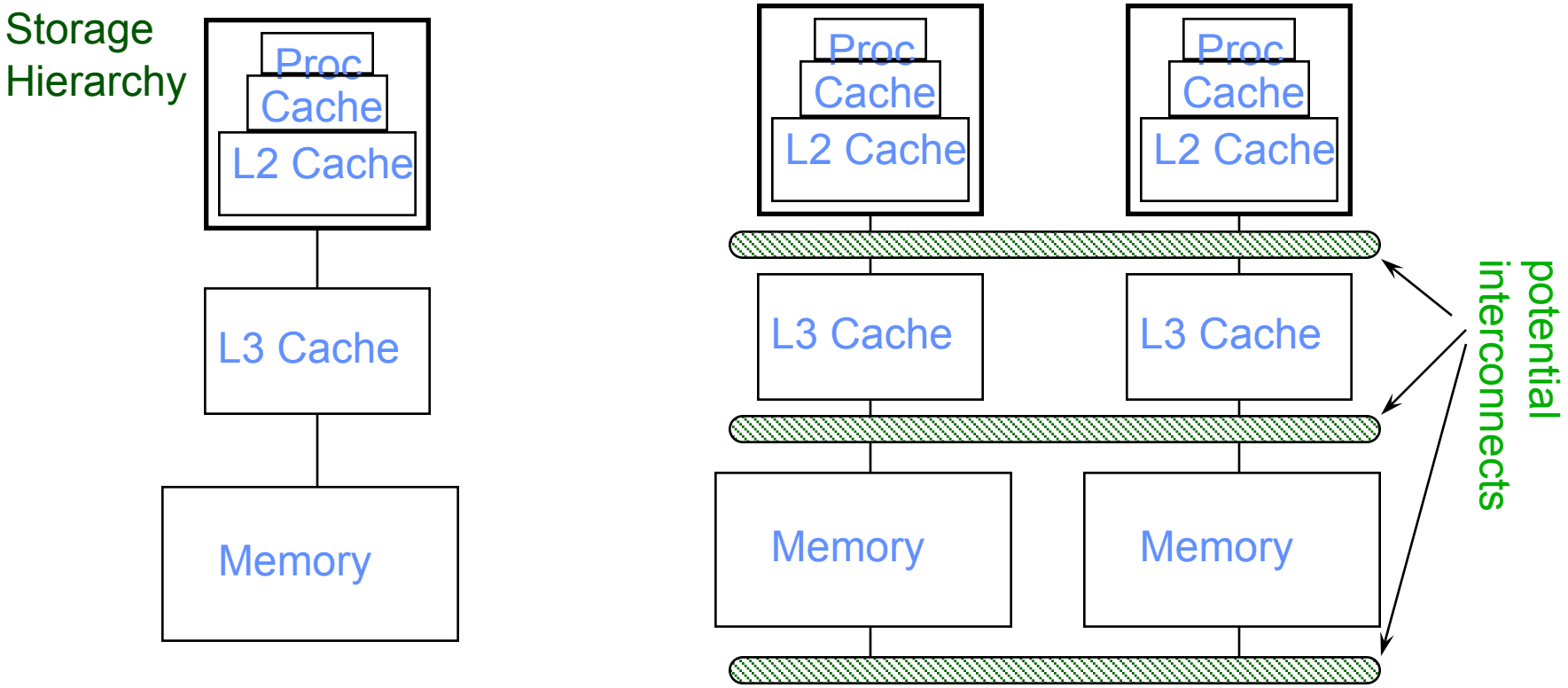
4-core Intel Sandy Bridge

(Triton uses an 8-core version)



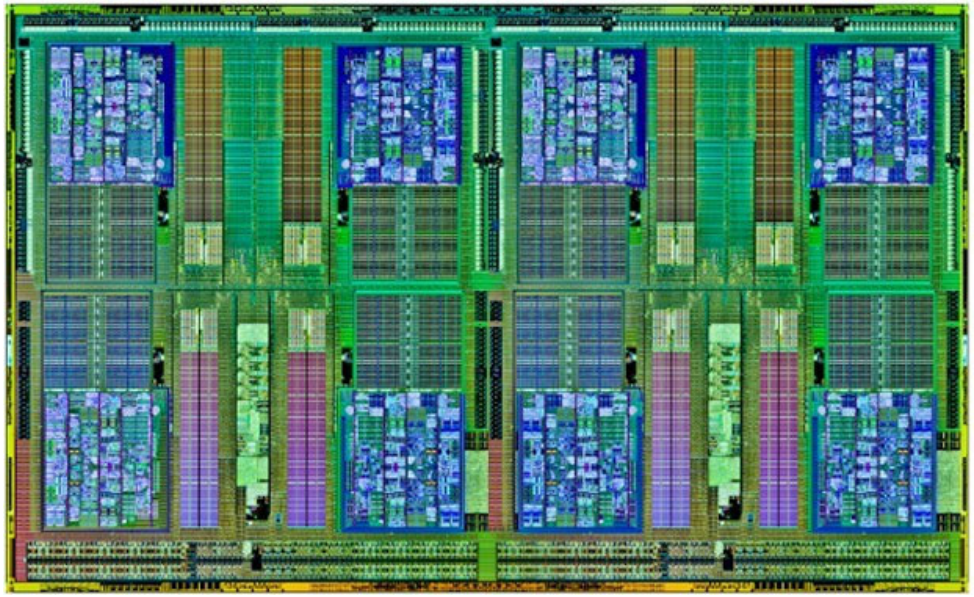
2600 Mhz clock speed

Generic Parallel Machine Architecture

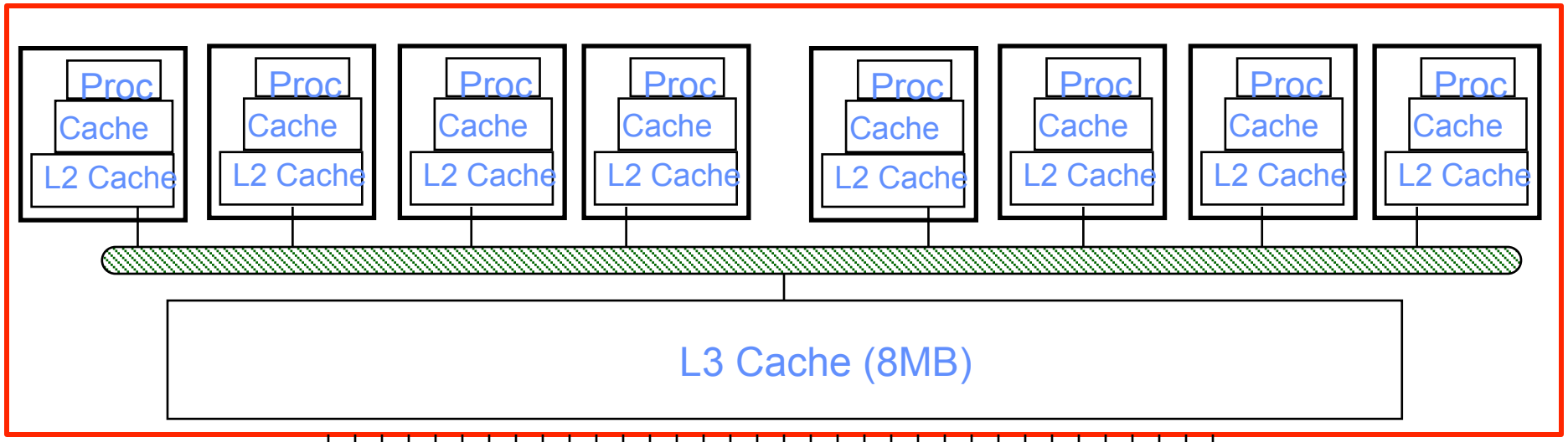


- Key architecture question: Where and how fast are the interconnects?
- Key algorithm question: Where is the data?

Triton memory hierarchy: I (Chip level)

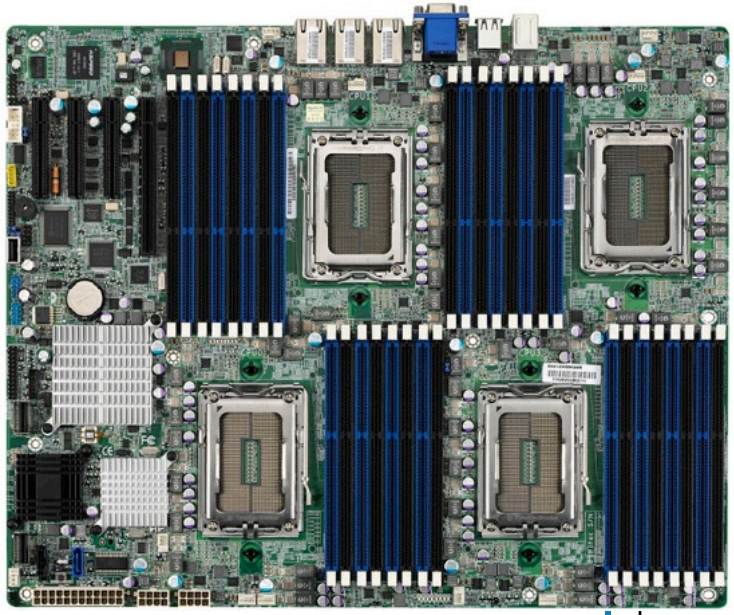


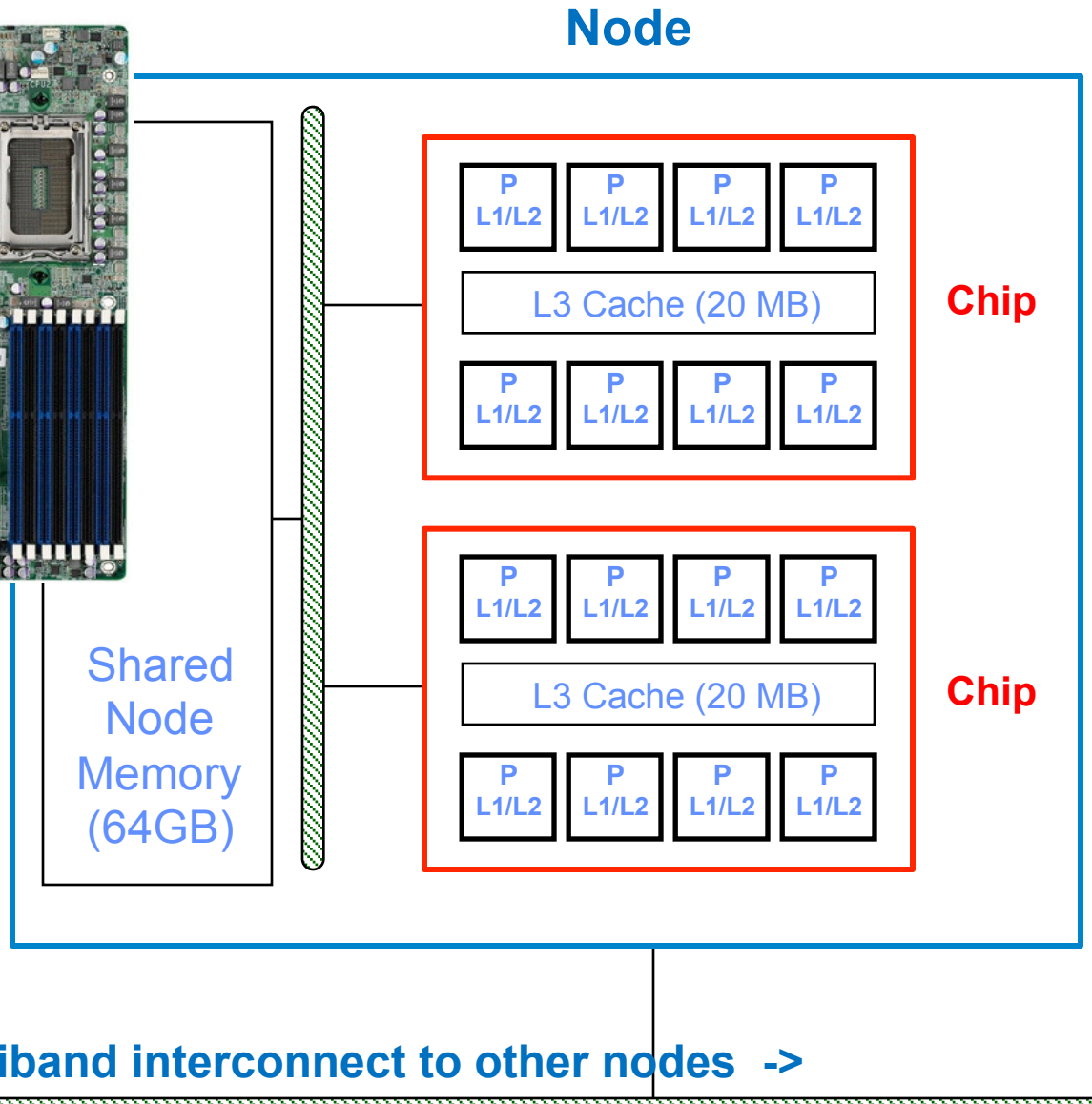
(AMD Opteron 8-core Magny-Cours, similar to Triton's Intel Sandy Bridge)



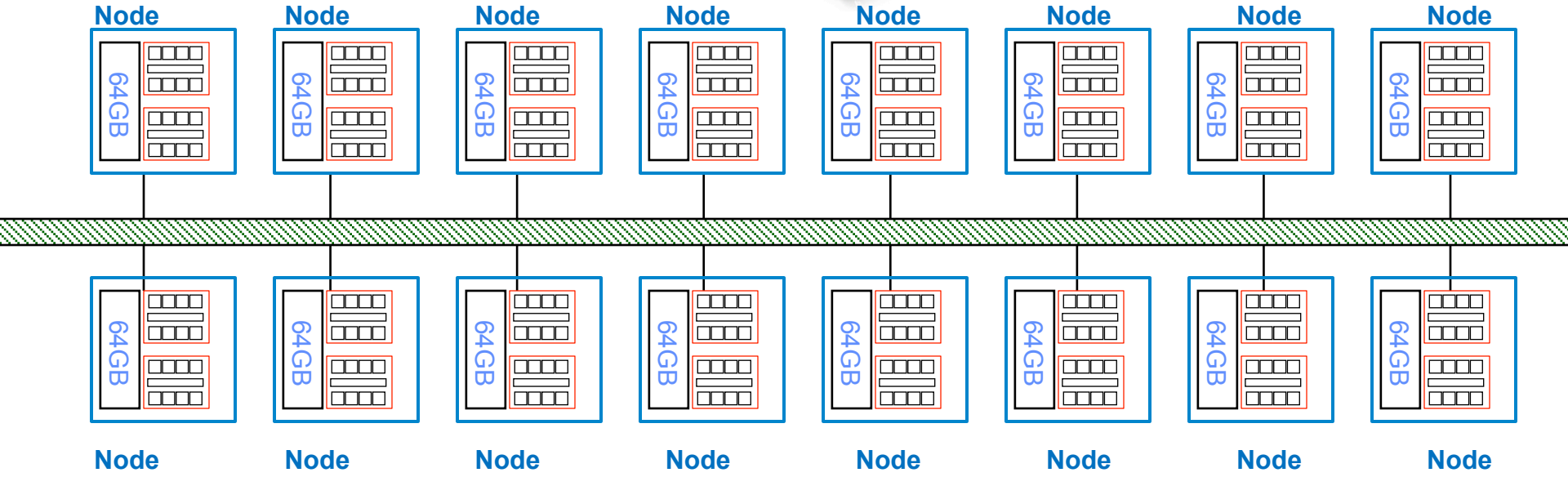
Chip sits in socket, connected to the rest of the node . . .

Triton memory hierarchy II (Node level)





Triton memory hierarchy III (System level)



324 nodes, message-passing communication, no shared memory

Some models of parallel computation

Computational model

- Shared memory
- SPMD / Message passing
- SIMD / Data parallel
- PGAS / Partitioned global
- Loosely coupled
- Hybrids …

Languages

- Cilk, OpenMP, Pthreads …
- MPI
- Cuda, Matlab, OpenCL, …
- UPC, CAF, Titanium
- Map/Reduce, Hadoop, …
- ???

Parallel programming languages

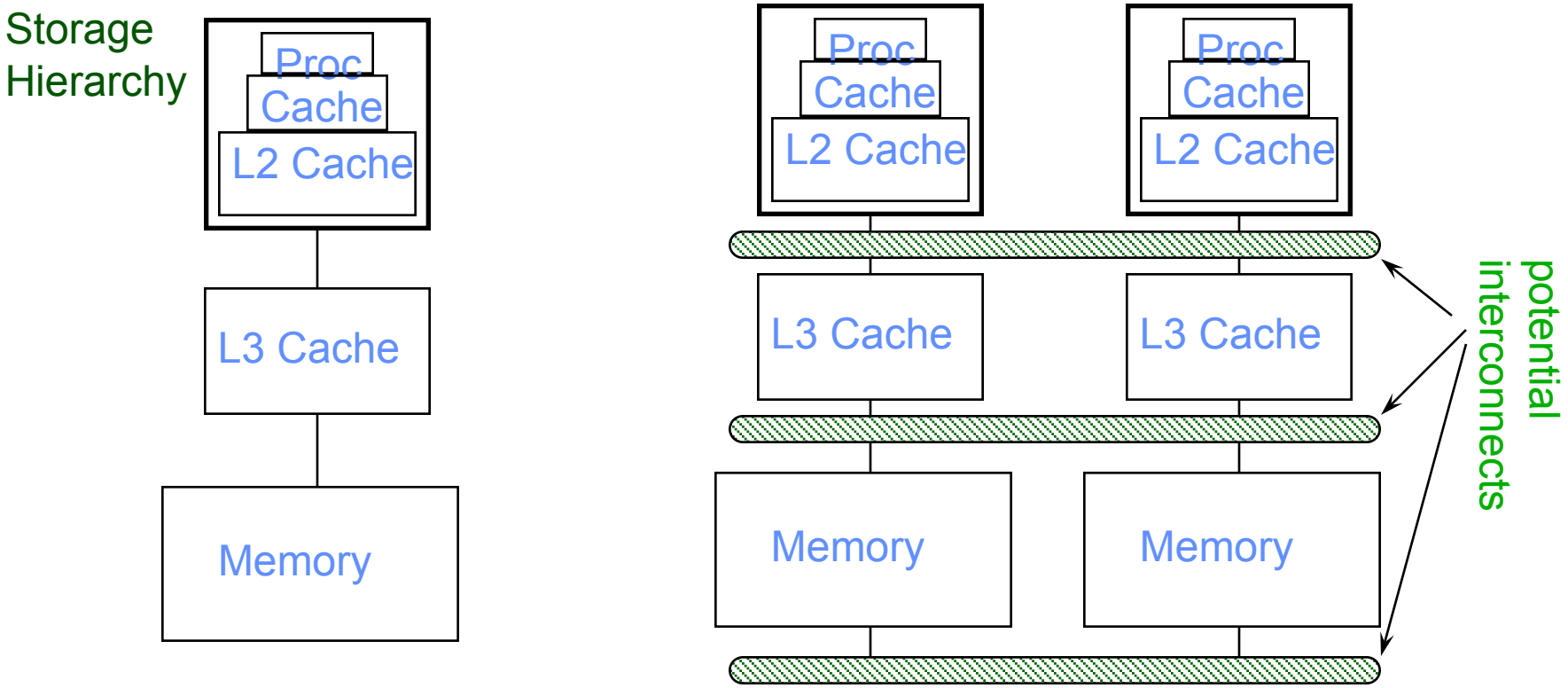
- Many have been invented – *much* less consensus on what are the best languages than in the sequential world.
- Could have a whole course on them; we'll look just a few.

Languages you'll use in homework:

- C with MPI (very widely used, very old-fashioned)
- Cilk Plus (a newer upstart)

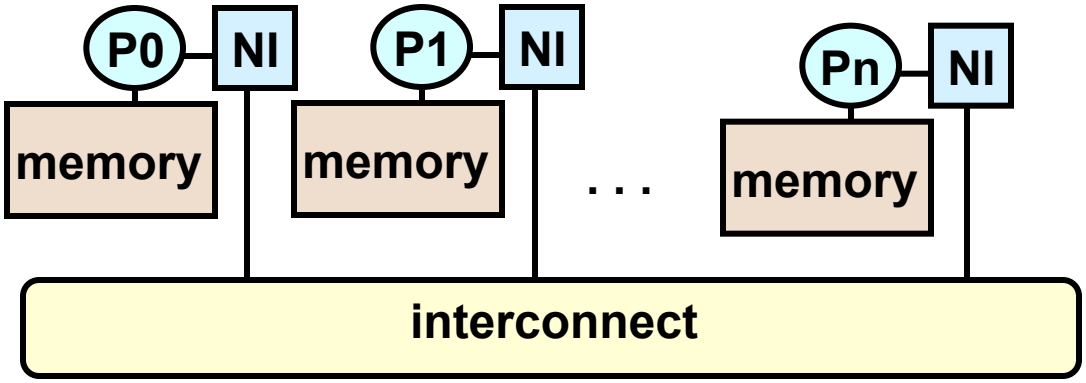
- You will choose a language for the final project

Generic Parallel Machine Architecture



- Key architecture question: Where and how fast are the interconnects?
- Key algorithm question: Where is the data?

Message-passing programming model



- Architecture: Each processor has its own memory and cache but cannot directly access another processor's memory.
- Language: MPI ("Message-Passing Interface")
  - A least common denominator based on 1980s technology
  - Links to documentation on course home page
  - SPMD = "Single Program, Multiple Data"

Hello, world in MPI

```
#include <stdio.h>
#include "mpi.h"

int main( int argc, char *argv[])
{
    int rank, size;
    MPI_Init( &argc, &argv );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    printf( "Hello world from process %d of %d\n",
            rank, size );
    MPI_Finalize();
    return 0;
}
```

MPI in nine routines (all you really need)

`MPI_Init`	Initialize
`MPI_Finalize`	Finalize
`MPI_Comm_size`	How many processes?
`MPI_Comm_rank`	Which process am I?
`MPI_Wtime`	Timer
`MPI_Send`	Send data to one proc
`MPI_Recv`	Receive data from one proc
`MPI_Bcast`	Broadcast data to all procs
`MPI_Reduce`	Combine data from all procs

Ten more MPI routines (sometimes useful)

More collective ops (like Bcast and Reduce):

```
MPI_Alltoall, MPI_Alltoallv
MPI_Scatter, MPI_Gather
```

Non-blocking send and receive:

```
MPI_Isend, MPI_Irecv
MPI_Wait, MPI_Test, MPI_Probe, MPI_Iprobe
```

Synchronization:

```
MPI_Barrier
```

Example: Send an integer x from proc 0 to proc 1

```
MPI_Comm_rank(MPI_COMM_WORLD,&myrank); /* get rank */

int msgtag = 1;
if (myrank == 0) {
   int x = 17;
   MPI_Send(&x, 1, MPI_INT, 1, msgtag, MPI_COMM_WORLD);
} else if (myrank == 1) {
   int x;
   MPI_Recv(&x, 1, MPI_INT,0,msgtag,MPI_COMM_WORLD,&status);
}
```

Some MPI Concepts

- **Communicator**
  - A set of processes that are allowed to communicate among themselves.
  - Kind of like a "radio channel".
  - Default communicator: MPI_COMM_WORLD
  - A library can use its own communicator, separated from that of a user program.

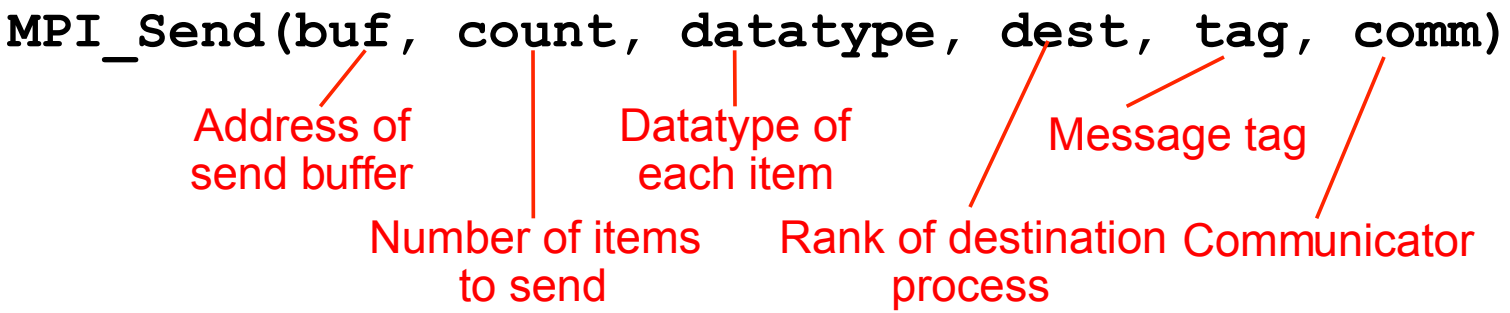
Some MPI Concepts

- **Data Type**
  - **What kind of data is being sent/recvd?**
  - **Mostly just names for C data types**
  - **MPI_INT, MPI_CHAR, MPI_DOUBLE, etc.**

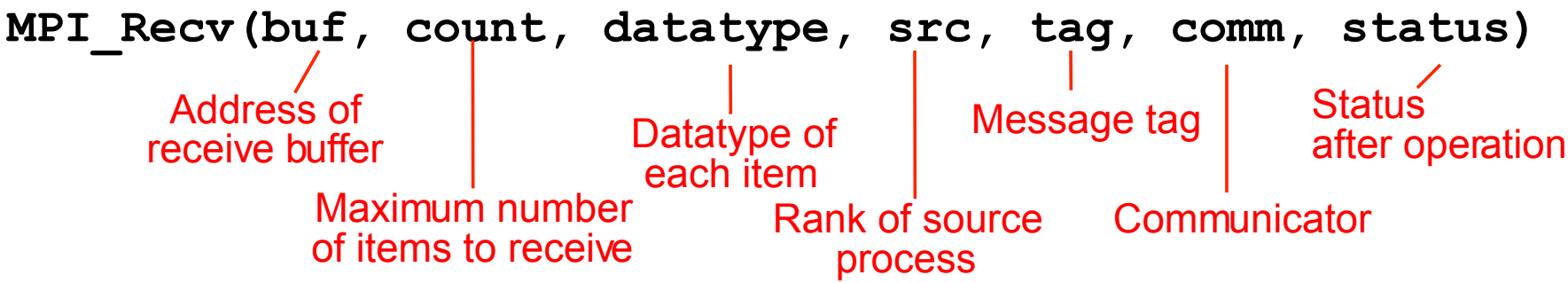
Some MPI Concepts

- **Message Tag**
  - **Arbitrary (integer) label for a message**
  - **Tag of Send must match tag of Recv**
  - **Useful for error checking & debugging**

Parameters of blocking send



Parameters of blocking receive



Example: Send an integer x from proc 0 to proc 1

```
MPI_Comm_rank(MPI_COMM_WORLD,&myrank); /* get rank */

int msgtag = 1;
if (myrank == 0) {
   int x = 17;
   MPI_Send(&x, 1, MPI_INT, 1, msgtag, MPI_COMM_WORLD);
} else if (myrank == 1) {
   int x;
   MPI_Recv(&x, 1, MPI_INT,0,msgtag,MPI_COMM_WORLD,&status);
}
```