

# Lecture 9

Shared memory:  
Architecture and programming

# Announcements

# Lab reports

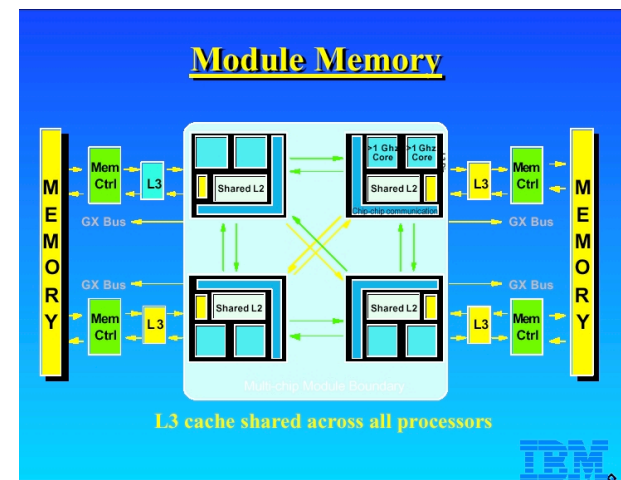
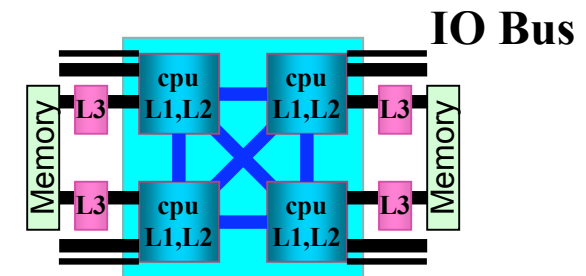
- Develop the discussions: goals, methods, commentary, speculation, reflection
- How will your plots look in Black and White?
- Electronic vs. hard copy
  - Code fragments in hard copy, full listings in electronic version
  - Tabular data: small tables in the report, large tables (multiple pages, raw data) in the electronic turnin
- See this document  
<http://www.cse.ucsd.edu/classes/wi08/cse260/HW/Writeups>

## Core Dump for Lab 2

- Peak bandwidth on-node:  $\sim 3\text{GB/sec}$ , 1.5GB off node
- Copying cost:
- Interaction with message size sampling interval, and starting message size
- Power4 memory hierarchy

# Power4 memory hierarchy

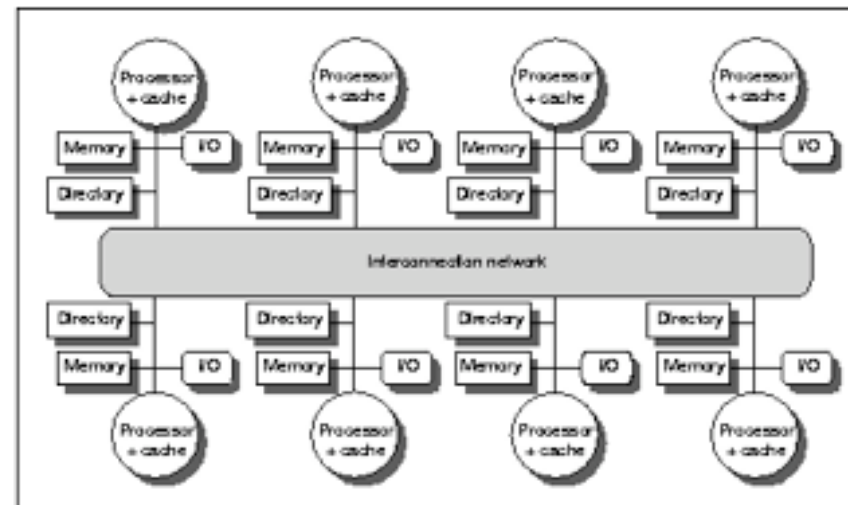
- 4-way SMP Multi-Chip Module
  - >41.6 GB/sec chip-to-chip interconnect & MCM-MCM
  - Logically shared L2 and L3 caches
- Caches
  - 128B cache lines
  - L1: 32KB/data 2-way assoc. (write through, 64KB/instruction direct mapped)
  - L2: 1.44MB (unified) 8-way assoc. (write-in), 4x128 byte lines  
Cache lines hashed across three L2 cache controllers
  - L3: 32MB 8-way assoc.
- 1024 entry TLB (4K or 16M page sizes)



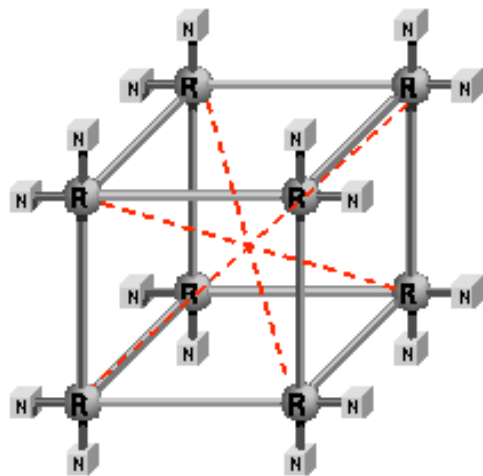
<http://www.sdsc.edu/us/training/workshops/institute2002/PPT/power4-overview.ppt>

# NUMA Architectures

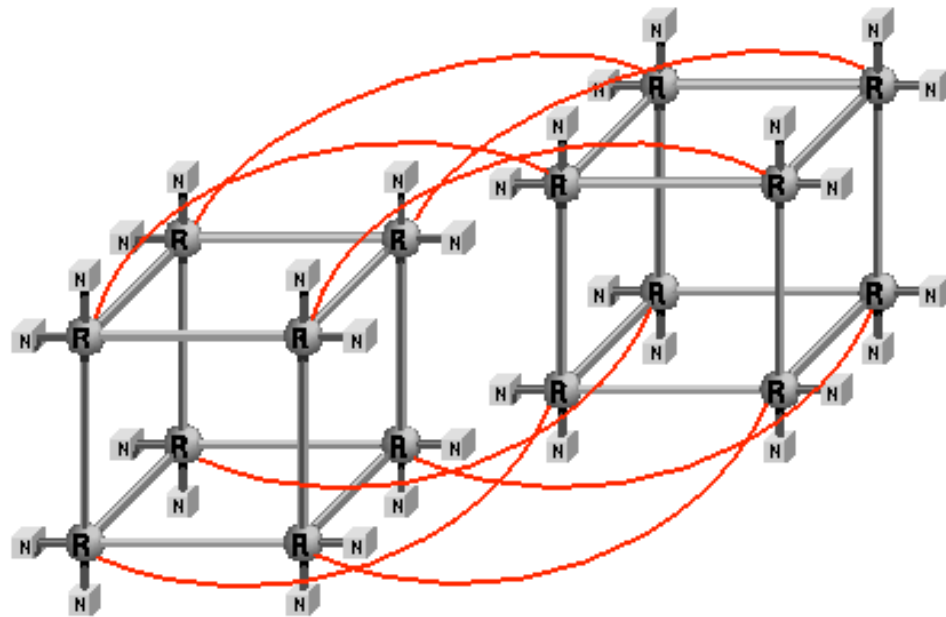
- The address space is global to all processors
- Point-to-point messages manage coherence
- A **directory** keeps track of sharers, one for each block of memory
- Stanford Dash; SGI Origin 2000



# Origin 2000 Interconnect

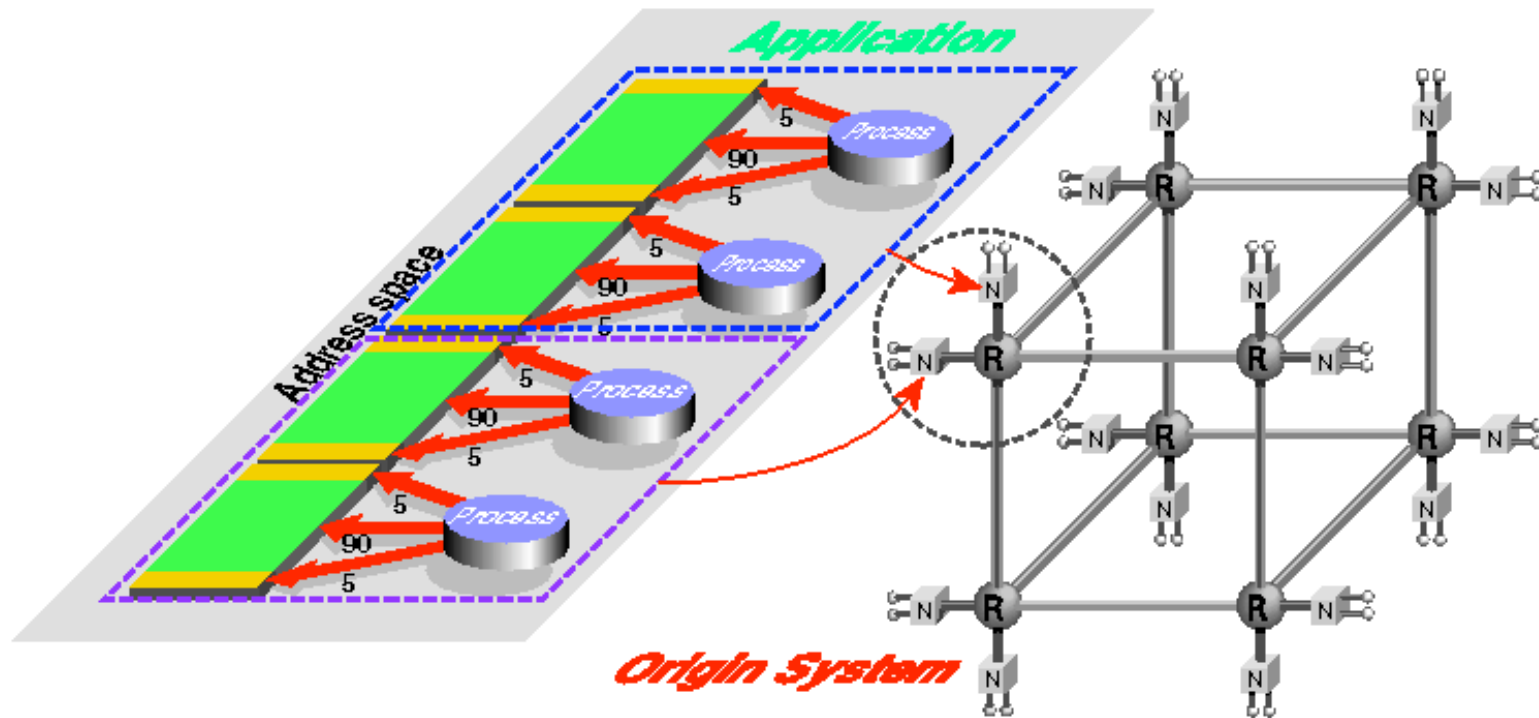


32 processor system

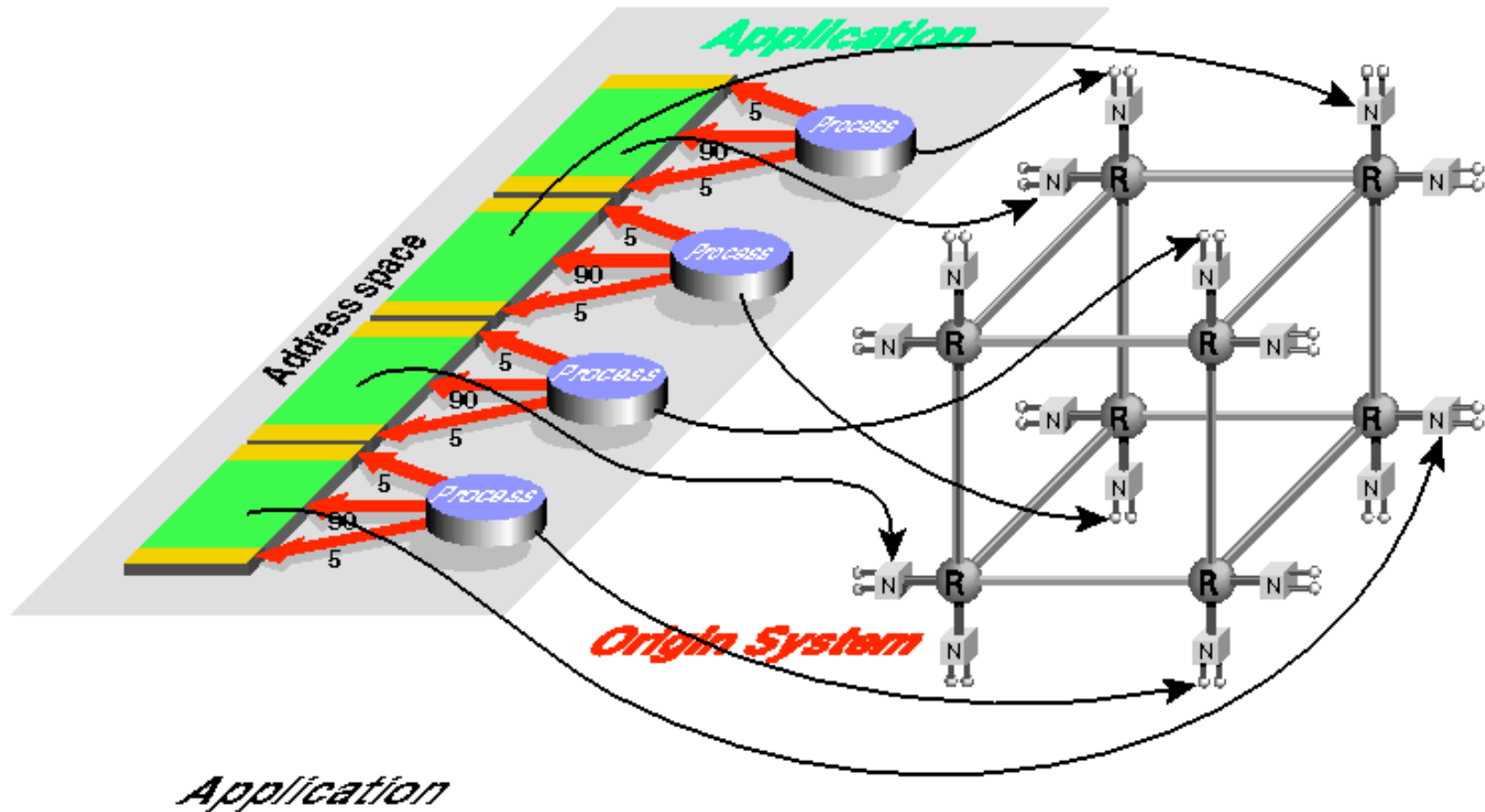


64 processor system

# Locality

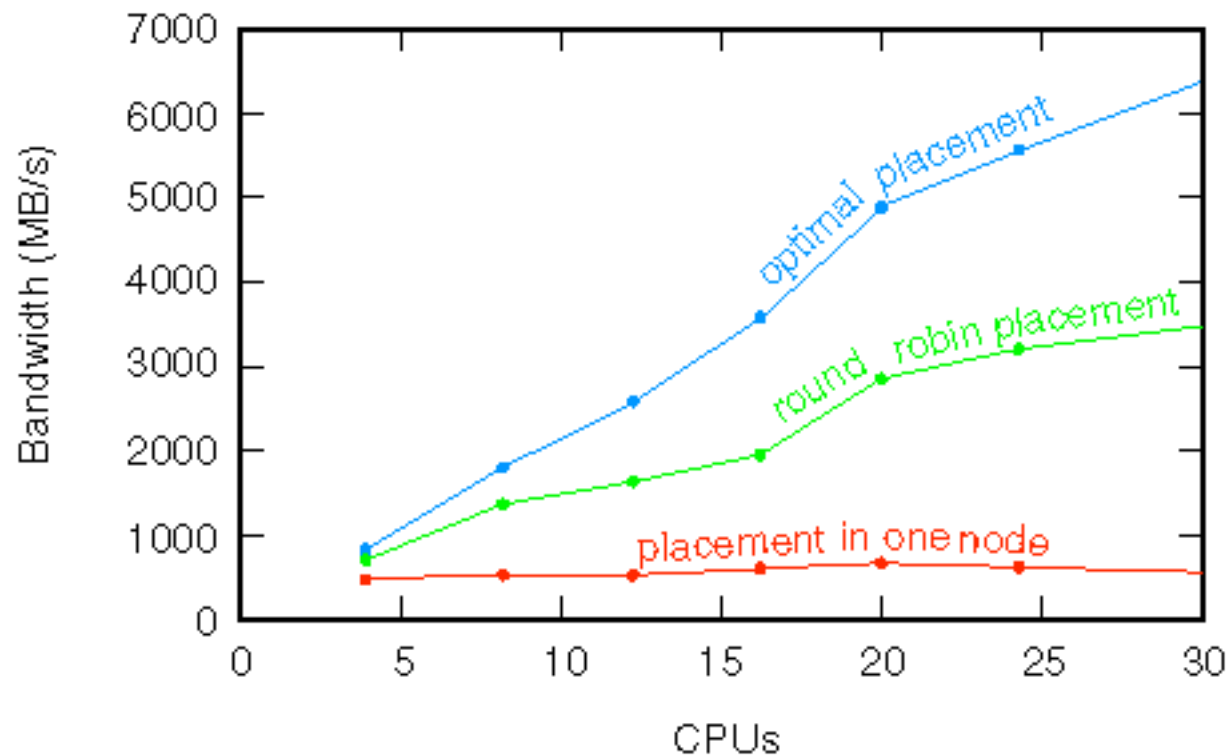


# Poor Locality



# Bisection Bandwidth and Latency

$$a(i) = b(i) + q * c(i)$$

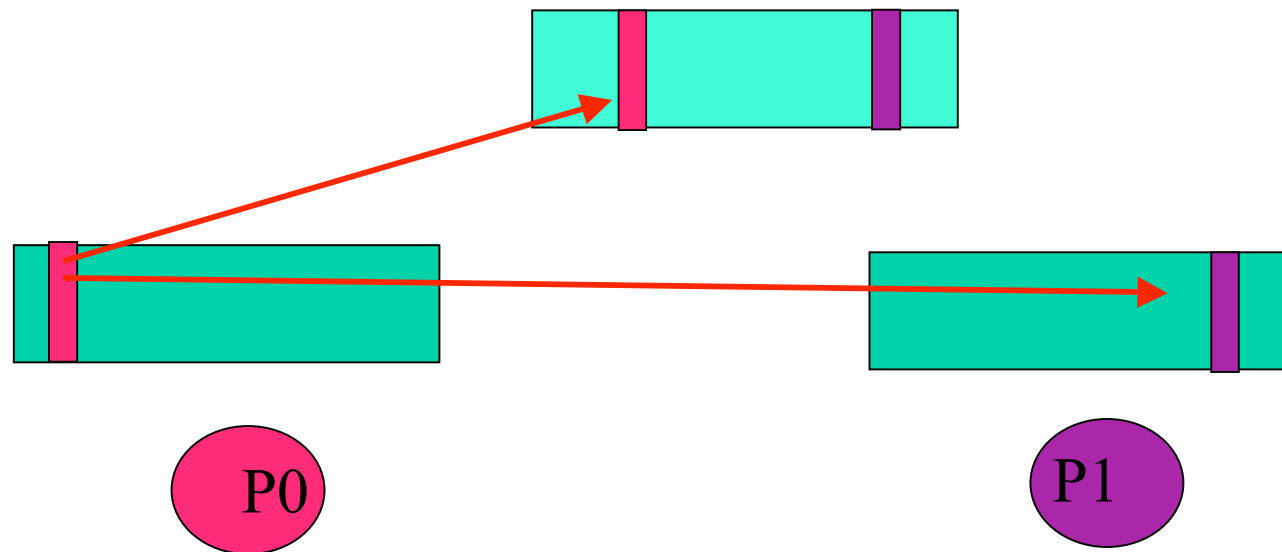


# Parallelization via the compiler

```
integer n, lda, ldr, lds, i, j, k
real*8 a(lda,n), r(ldr,n), s(lds,n)
!$OMP PARALLEL DO private(j,k,i), shared(n,a,r,s),
  schedule(static)
  do j = 1, n
    do k = 1, n
      do i = 1, n
        a(i,j) = a(i,j) + r(i,k)*s(k,j)
      enddo
    enddo
  enddo
```

# False sharing

Successive writes by P0 and P1 cause the processors to uselessly invalidate one another's cache

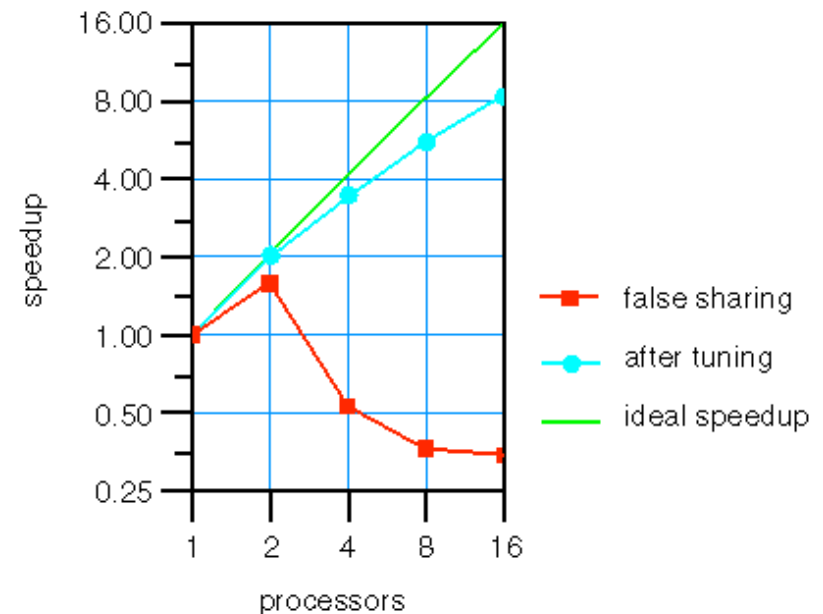


## An example of false sharing

```
integer m, n, i, j
real  a(m,n), s(m)
! At each stage all CPUs update s(i)
!$omp parallel do private(i,j), shared(s,a)
  do i = 1, m
    s(i) = 0.0
    do j = 1, n
      s(i) = s(i) + a(i,j)
    enddo
  enddo
```

# Removing false sharing

```
integer m, n, i, j
real  a(m,n), s(32,m)
!$omp parallel do private(i,j), shared(s,a)
do i = 1, m
  s(1,i) = 0.0
  do j = 1, n
    s(1,i) = s(1,i) + a(i,j)
  enddo
enddo
```

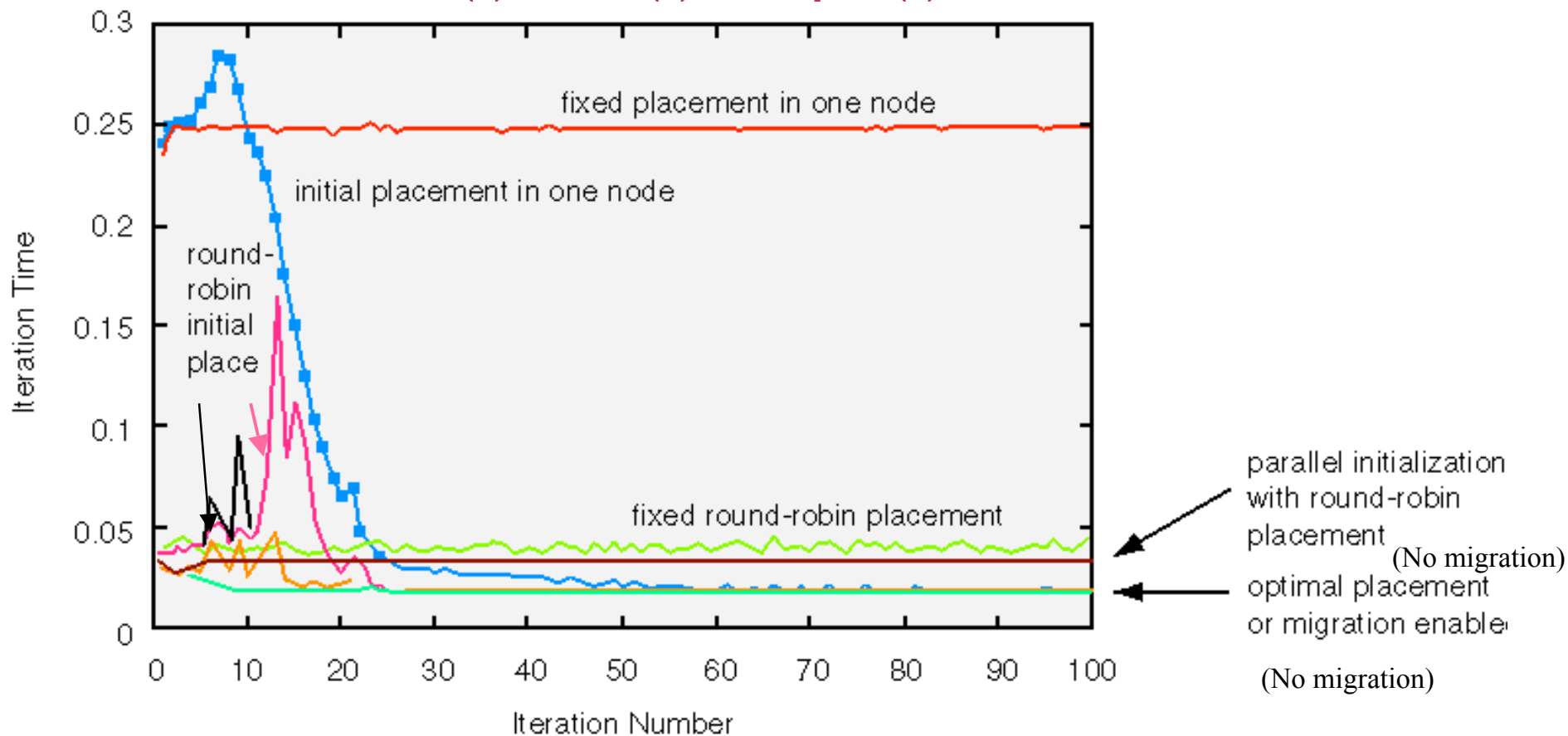


# Remote access latency

- When we allocate a block of memory, which processor(s) is (are) the owner(s)?
- Page allocation policies
  - First touch
  - Round robin
- We can control memory locality with the same kind of data layouts that we use with message passing
- Page migrations

# Page Migration

$$a(i) = b(i) + q * c(i)$$



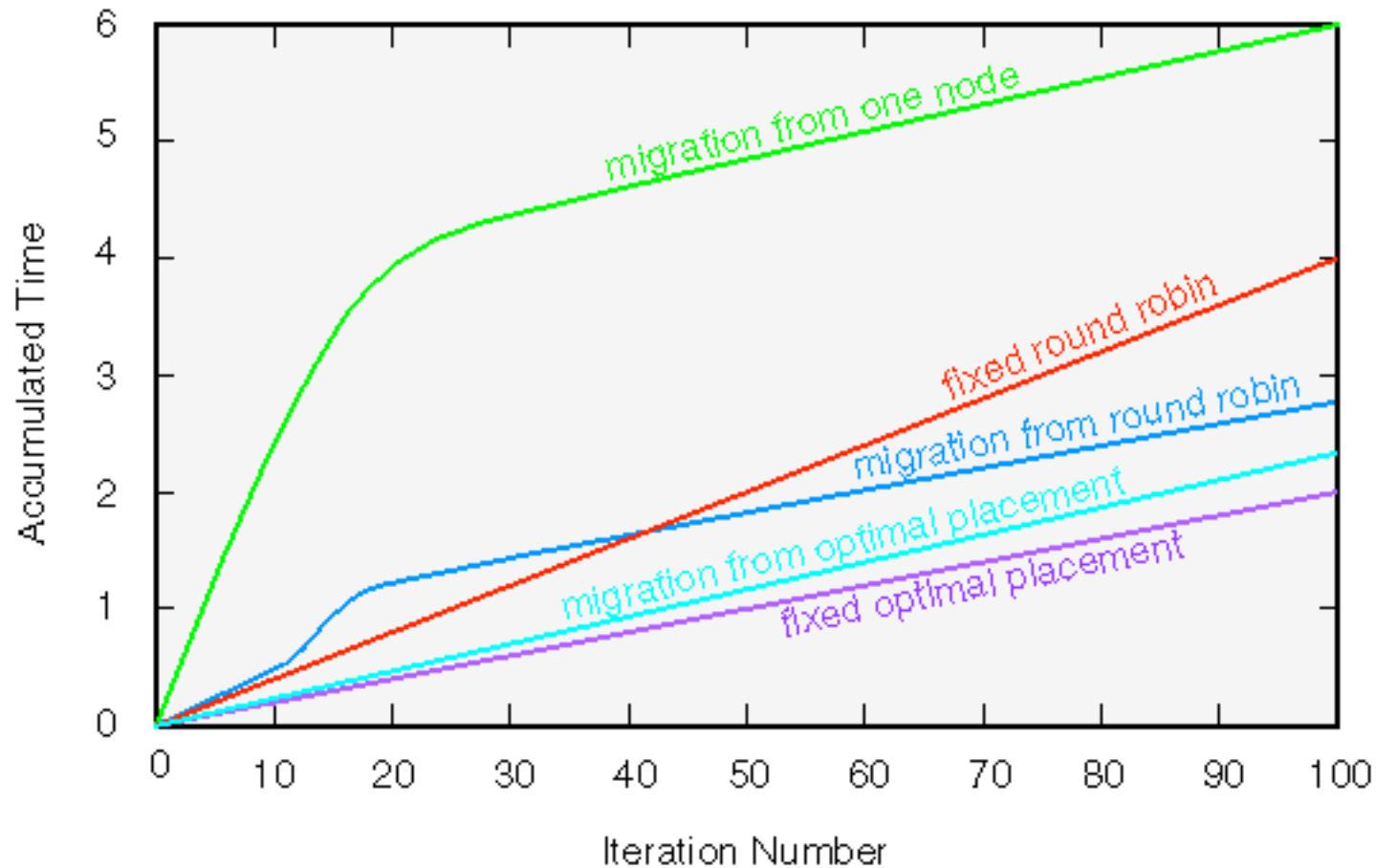
[http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi/0650/bks/SGI\\_Developer/books/OrOn2\\_PfTune/sgi\\_html/ch08.html#id5224855](http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi/0650/bks/SGI_Developer/books/OrOn2_PfTune/sgi_html/ch08.html#id5224855)

2/5/08

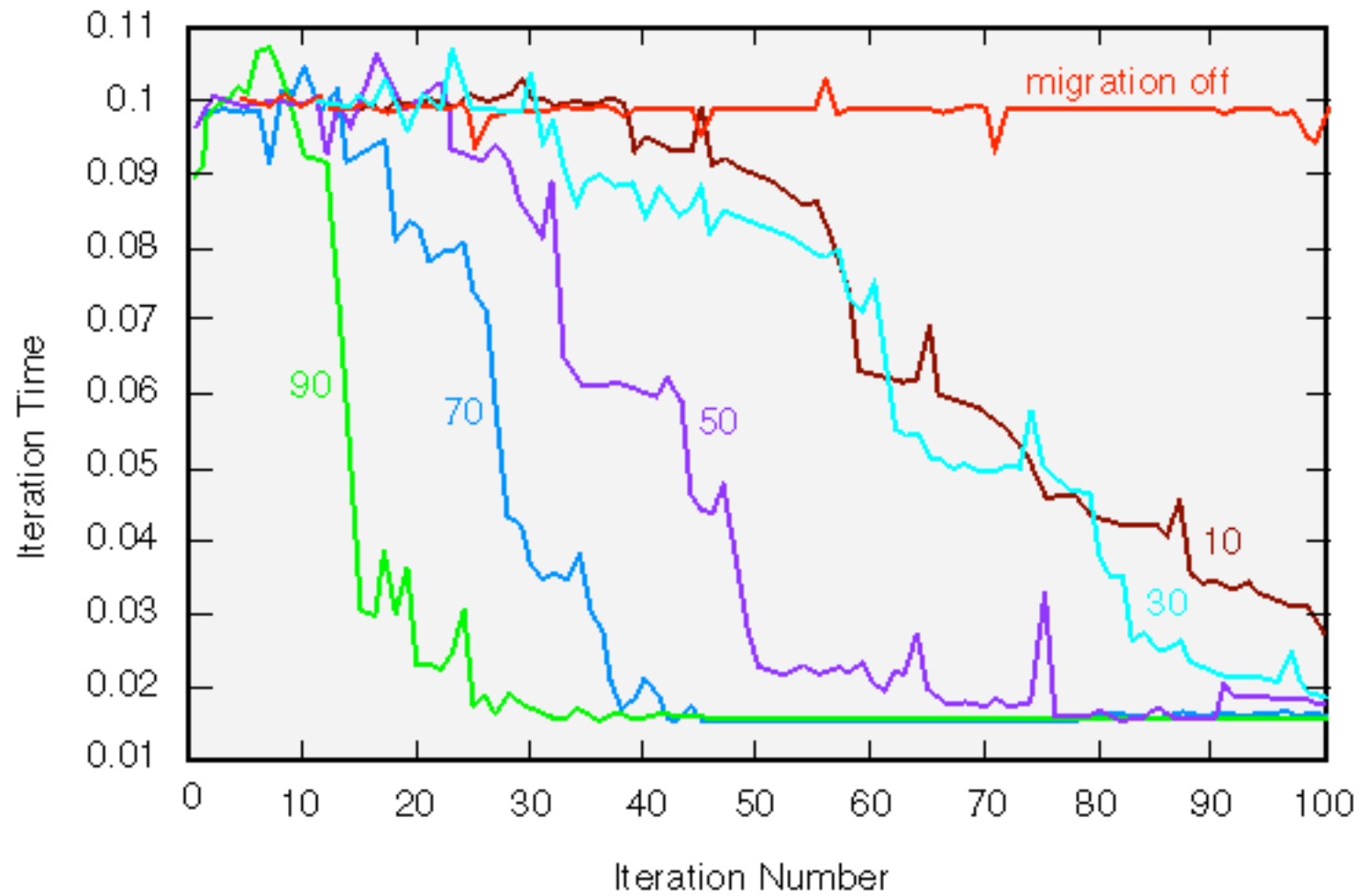
Scott B. Baden/CSE 260/Winter 2008

16

# Cumulative effect of Page Migration



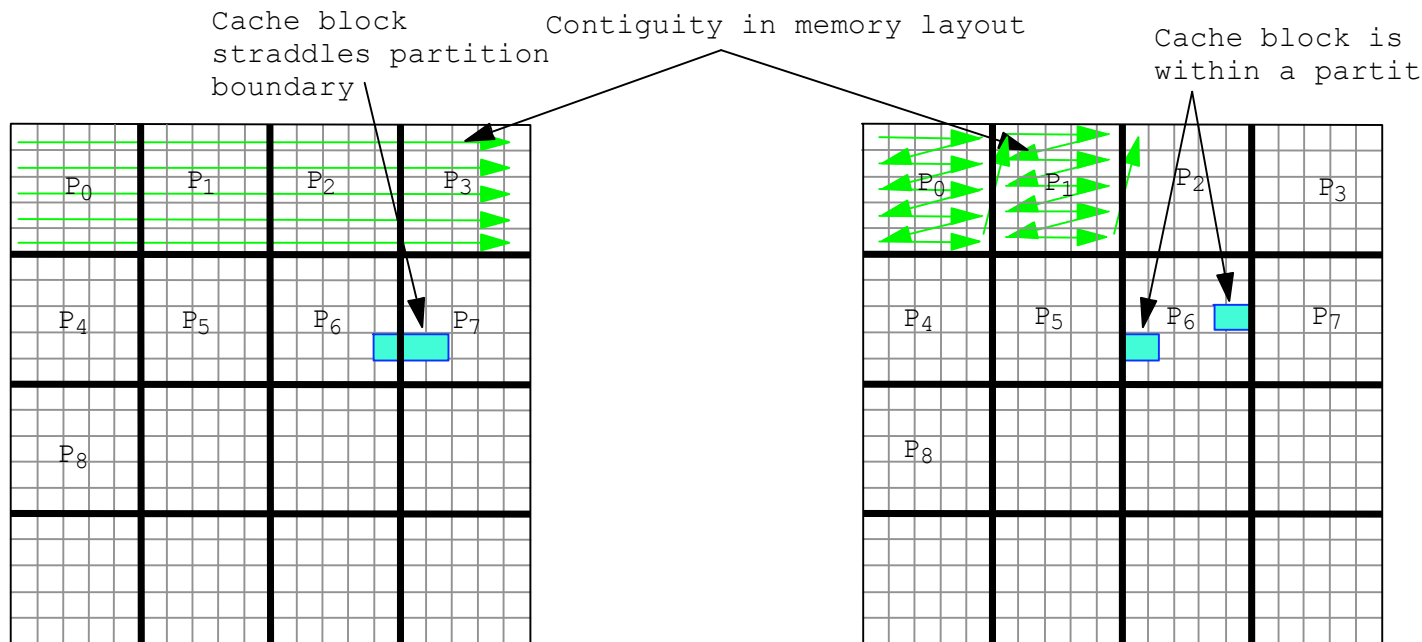
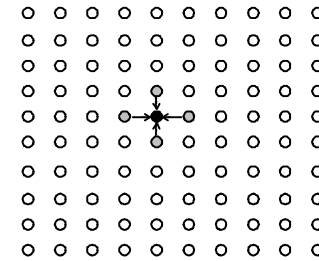
# Migration Level



# False sharing in higher dimension arrays

Jacobi's method

$$u_{new}[i,j] = (u[i-1,j] + u[i+1,j] + u[i,j-1] + u[i,j+1])/4;$$



**In global memory**

**Distributed memory**

*Parallel  
Computer  
Architecture,  
Culler,  
Singh, &  
Gupta*