Code Optimization and Parallelization on the Origins - Looking from Users' Perspective

Sherry Chang

schang@nas.nasa.gov
Scientific Consultant
NASA Advanced Supercomputing Division
NASA Ames Research Center
Moffett Field, CA

An “Old” Topic that is Hard to Master

Why ?

Cache !!

The 10 Steps and the Bottlenecks

- **Single-Processor Optimization and Tuning**

  1. get the right answers : porting issues, debugging etc. 🟢
  2. use existing tuned code : -lm, -lfastm, -lscs 🟢
  3. find out where to tune : time, perfex, SpeedShop 🟡
  4. let the compiler do the work : -LNO, -SWP 🟡
  5. tune cache performance : ex: cache thrashing 🔴

Cache is the key factor !

🔴 **Bottleneck**

Understanding **Cache Structure** of a Single CPU

The 10 Steps and the Bottlenecks (continued)

- **Multi-Processor Optimization and Tuning**

Cache is the key factor !

6. tune for 1-cpu first ☹
7. parallelize code : APO, OpenMP, MPI, MLP ☹
8. identify and solve bottlenecks (time, perfex, ssrun) ☹
9. fix false sharing ☹
10. tune for data placement ☹

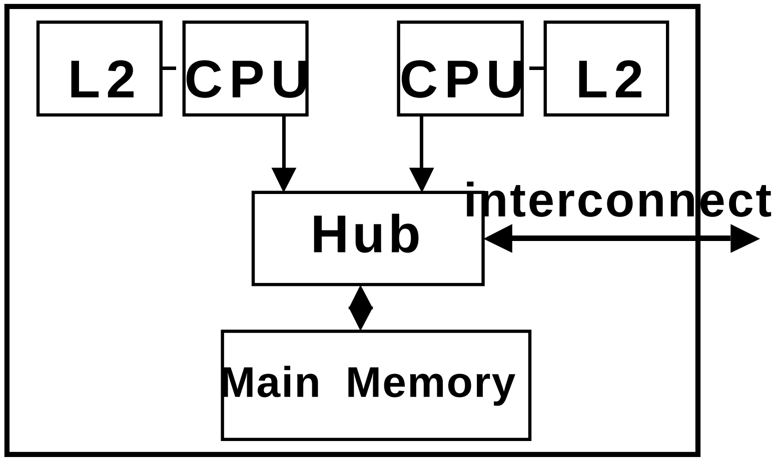
☹ **Bottleneck**

- Cache for each CPU (steps 6,7,8)
- Cache Coherency and Cache Contention (step 9)
- Non-Uniform Memory Access (cc-NUMA) (step 10)
- Page and Data Placement (step 10)

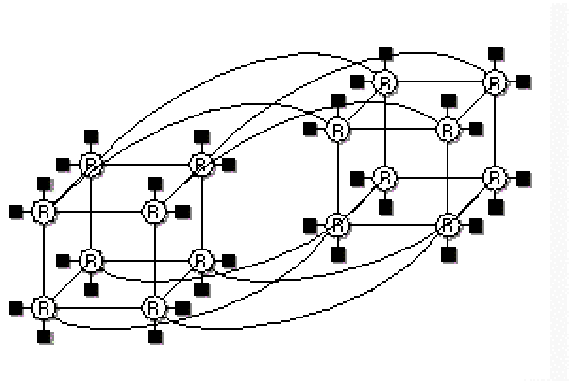
Learn these key concepts WELL !

The Memory Hierarchy on the Origins

Memory level	Total Capacity	Memory latency ! (CPU cycles)
registers	Bytes	0
L1 cache	KB	2 - 3
L2 cache	~ MB	8 - 10
Local memory	~ GB	75 - 200
Remote memory	< TB	> 200*
disk	> TB	Long long time



! actual values vary depending on CPU and memory speeds
* Memory access time increases with the number of router hops to access requested data.



Reduce Accesses to Main Memory (especially remote memory)

Be Cache Friendly is the Key to Good Performance !

Cache Structure

- Cache Size and Cache Line
- Two-Way Set Associative ☹
- Least Recently Used Policy

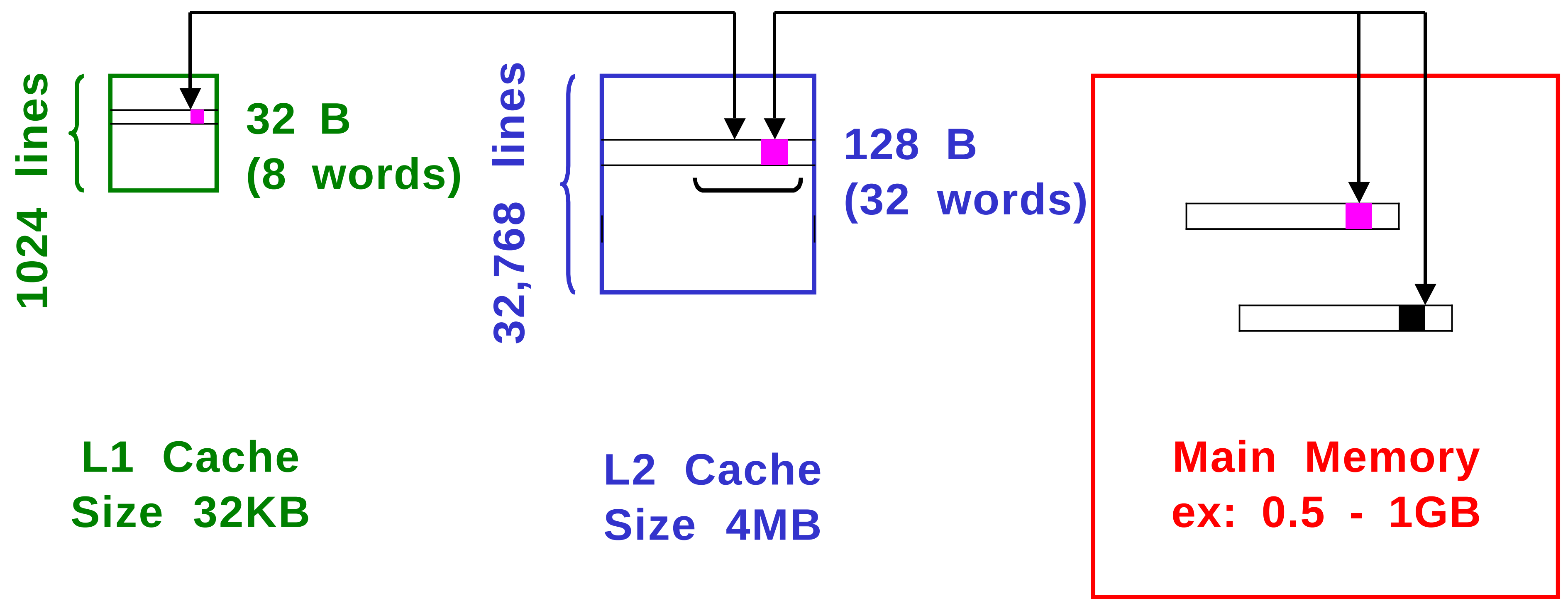
Cache Size and Cache Line

Cache Size : total capacity of cache << size of memory

Consequence : many memory locations map to same cache location(s)

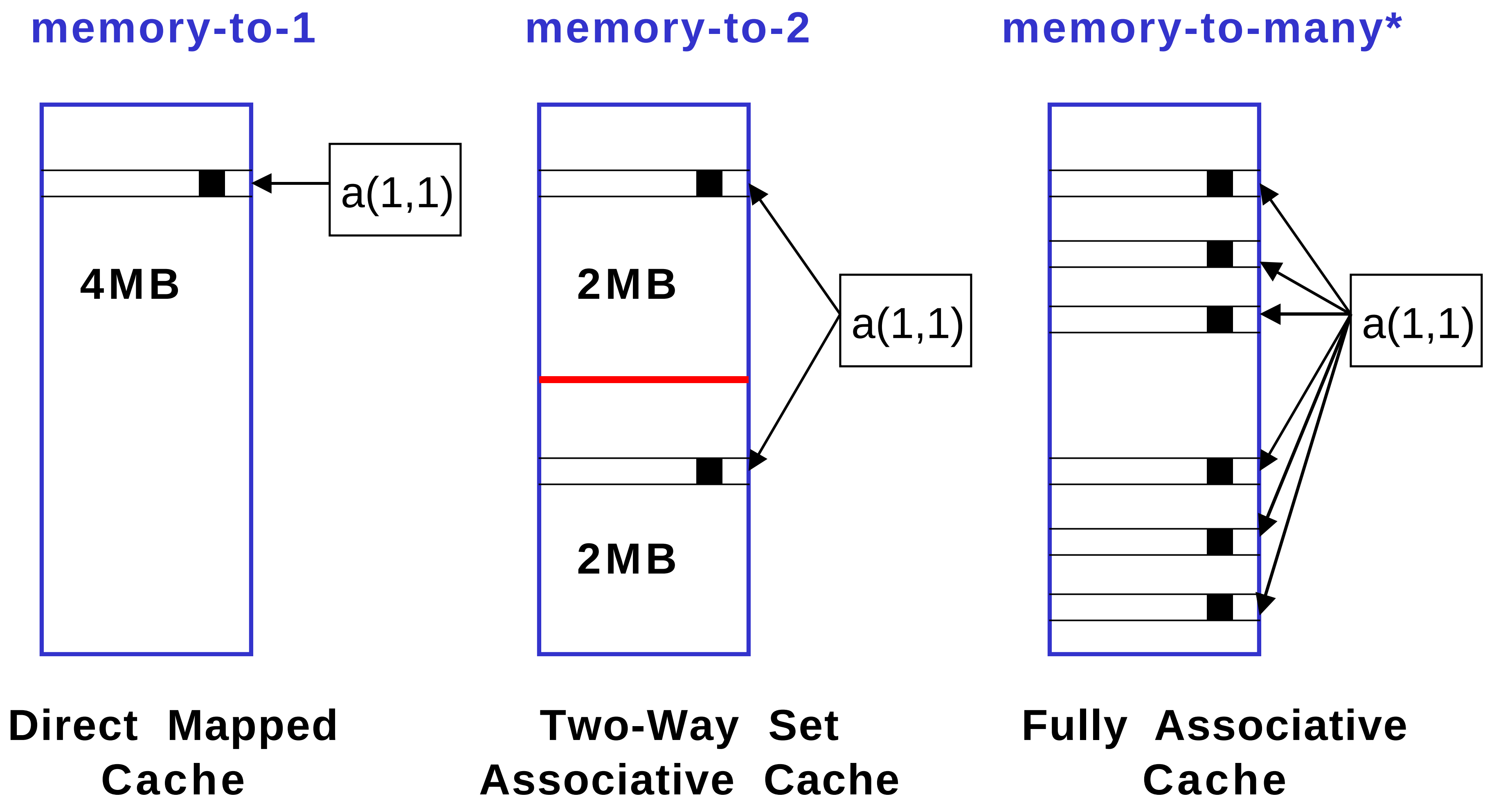
Cache Line: unit of transfer between main memory and L2 cache;
between L2 and L1 cache (typically, a few words)

Consequence : when a data is accessed, nearby data is also accessed



Memory-Cache Mapping

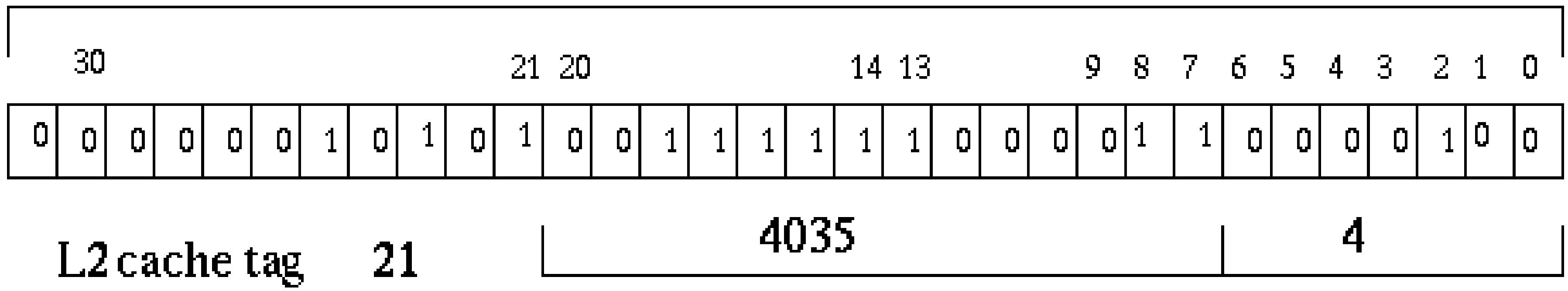
Can a memory location map to more than 1 cache location ?



Two-Way Set Associative - which set ?

memory address

address in decimal 44556676



L2 cache tag 21

Set number

Example:

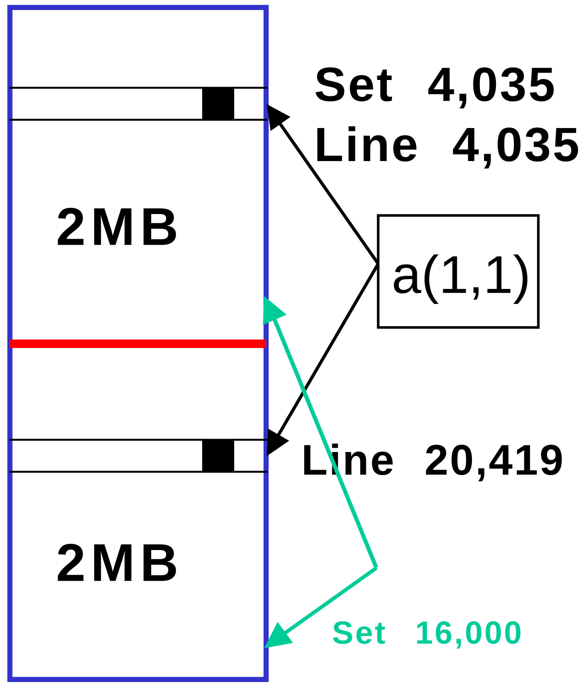
L2 cache size : 4MB = 222**
size in each way : 2MB = 221**

Two-way set : use lower 21 bits to determine cache location (bits 0-20)

Cache line : 128B = 27**

Bits 0-6 : which word in a cache line
Bits 7-20 : which set (two cache lines) to use

* assume L2 cache size : 4MB, cache line = 128B and 32 bits addressing

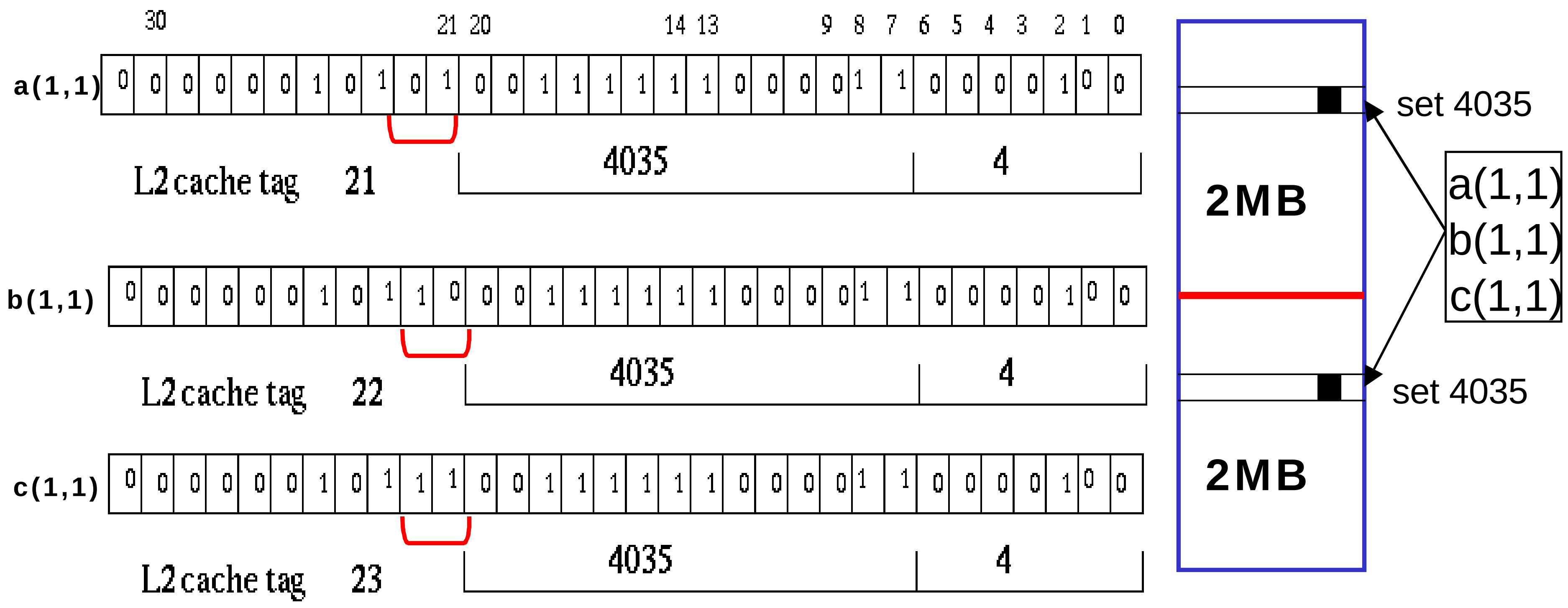


Cache Tag and Least-Recently-Used Policy

Assumption: L2 cache size : 4MB, 2-way Set

If two variables are 2MB (2**21) apart, they will have the same lower 21 bits (0-20). Thus they will map to the same (two) cache lines.

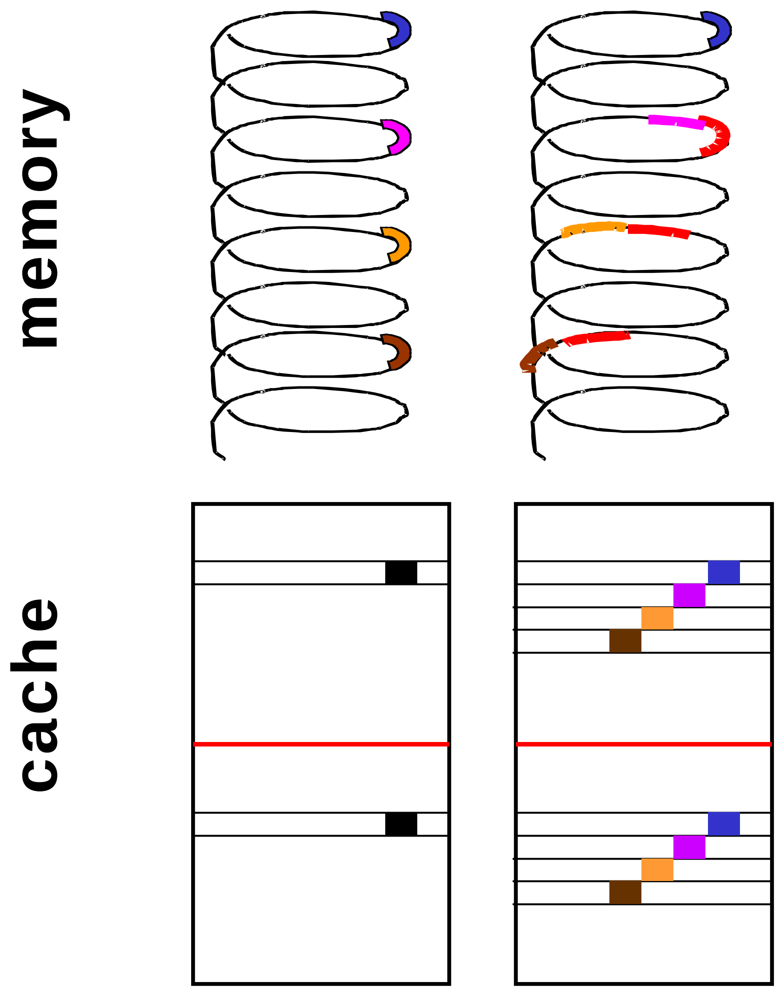
Ex: real*4 a(1024,512), b(1024,512), c(1024,512)



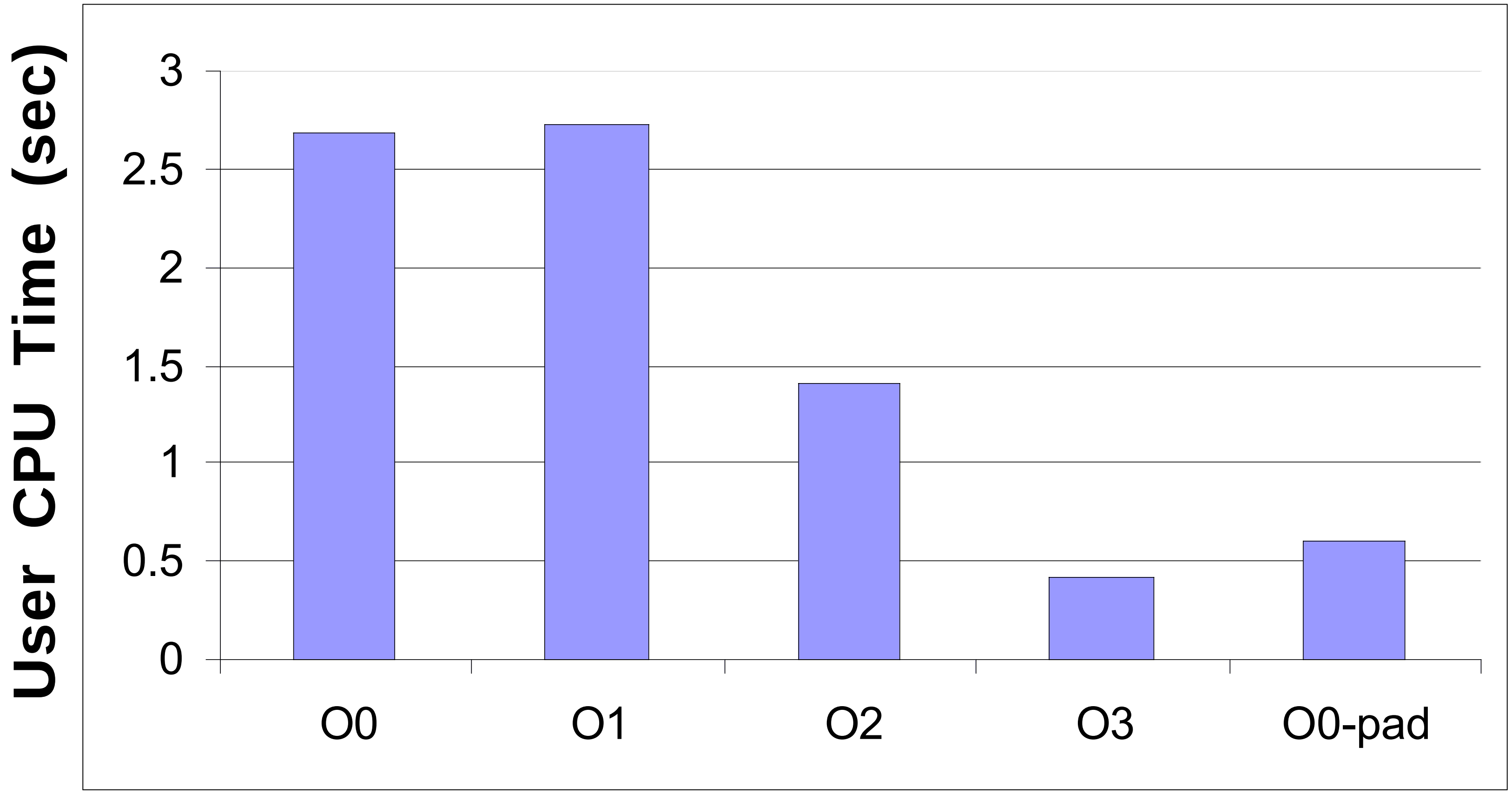
Cache Thrashing (Example 6-5)

```
  Real*4 a(1024,1024),
&          b(1024,1024),
&          c(1024,1024),
&          d(1024,1024)
                   .....
        do j=1,1024
        do i=1,1024
          a(i,j) =
      b(i,j)+c(i,j)*d(i,j)
          end do
          end do
```

Apply padding so that a(i,j), b(i,j), c(i,j), d(i,j) will not map to the same cache lines, thus preventing cache thrashing



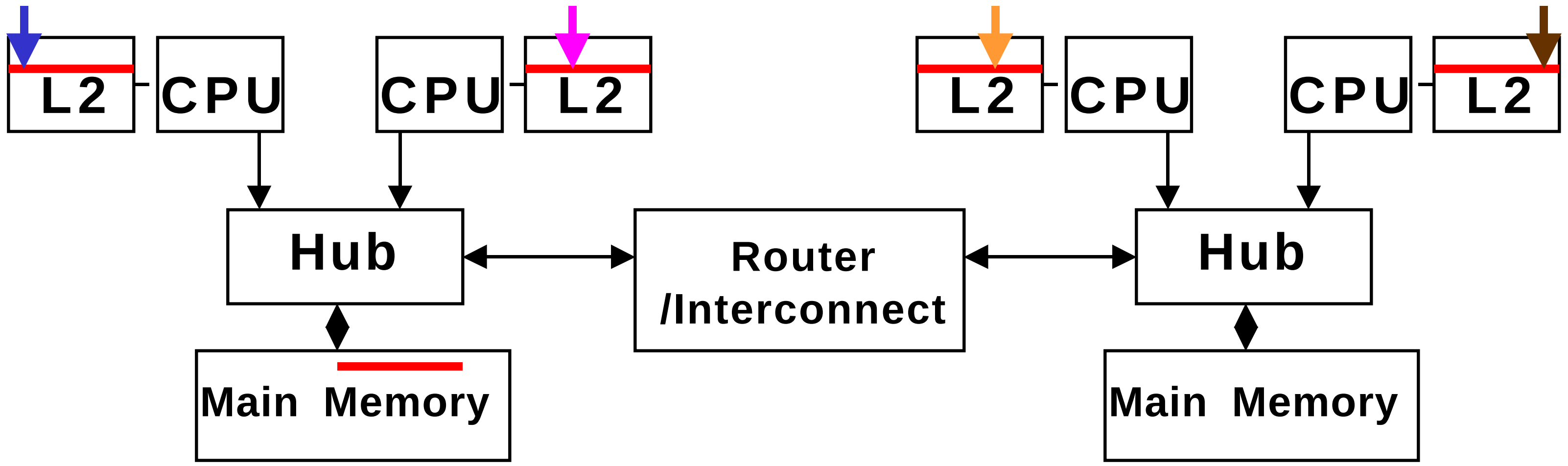
Performance Comparison



perfex -a -x -y (-O0 is used)

	event	Counter value	Typical time
0	Cycles	653654512	2.614618
16	Cycles	653654512	2.614618
26	**Secondary data cache misses**	**4178336**	**1.261857**
14	ALU/FPU progress cycles	82554064	0.330216
7	Quadwords written back from scache	8635504	0.221069
25	Primary data cache misses	4348624	0.156724
2	Issued loads	29675728	0.118703
18	Graduated loads	29524944	0.118100
3	Issued stores	29524944	0.118100
19	Graduated stores	11248768	0.044995
22	Quadwords written back from primary data cache	2475040	0.038116
21	Graduated floating point instructions	7787744	0.031151

Cache Coherency and Cache Contention



Memory contention: two or more CPUs try to update the same variables on the same cache line (may need algorithm change)

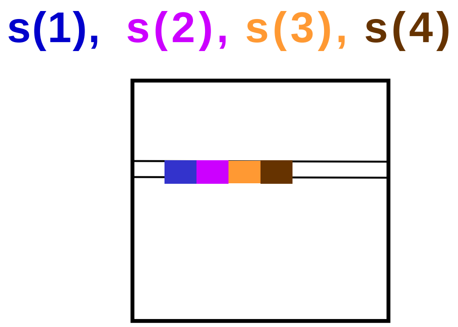
False-sharing : two or more CPUs repeatedly update independent data elements on the same cache line (fix can be simple)

When a CPU updates a cache line, the same copies in the other CPUs are invalidated. A new copy is then obtained from the CPU that has the fresh copy.

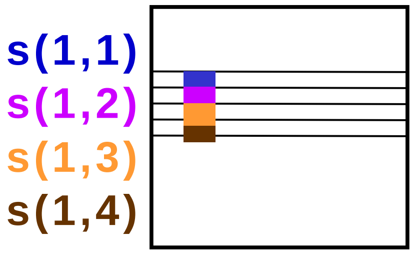
False Sharing (Examples 8-5; 8-6 modified)

False Sharing

```
parameter (m=4, n=100000)
real*4 a(n,m), s(m)
do k=1,100
call sum85(a,s,m,n)
end do

subroutine sum85 (a,s,m,n)
!$omp parallel do private(i,j),
!$omp&   shared(s,a)
do i = 1, m
s(i) = 0.0
do j = 1, n
s(i) = s(i) + a(j,i)
enddo
```



Fix

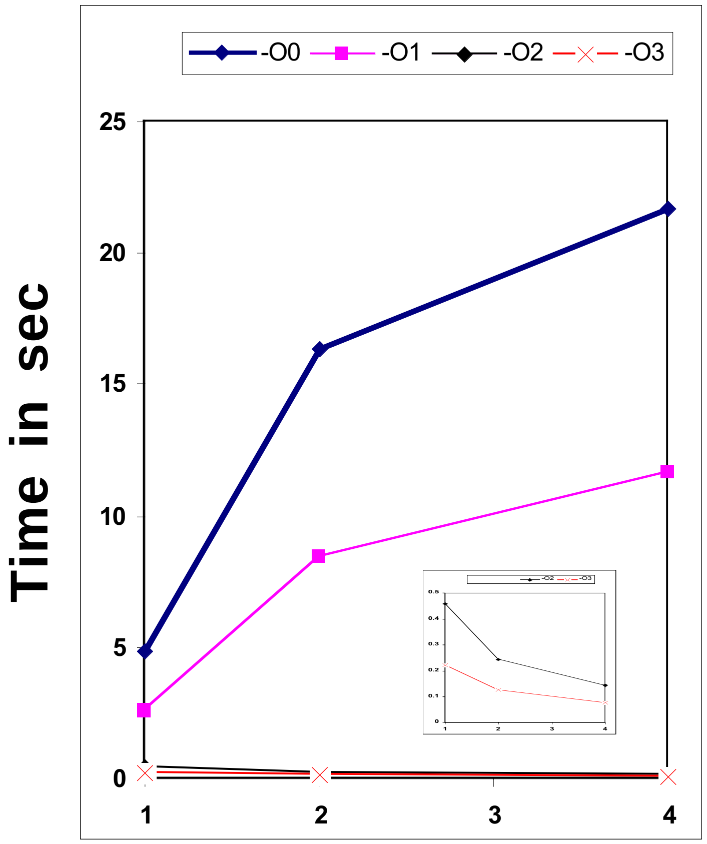


No False Sharing

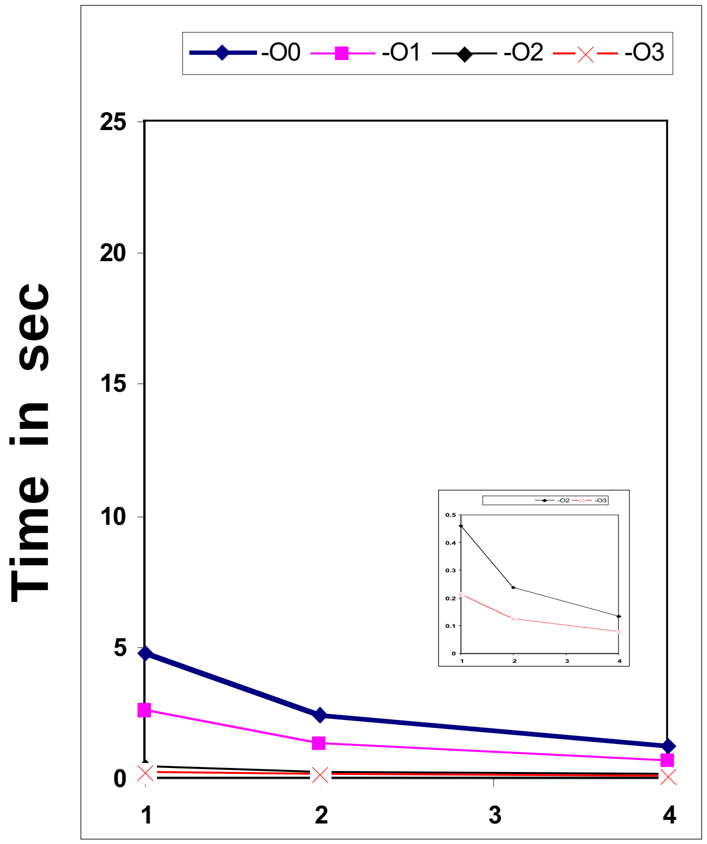
```
parameter (m=4, n=100000)
real*4 a(n,m), s(32,m)
do k=1,100
call sum85(a,s,m,n)
end do

subroutine sum85 (a,s,m,n)
!$omp parallel do private(i,j),
!$omp&   shared(s,a)
do i = 1, m
s(1,i) = 0.0
do j = 1, n
s(1,i) = s(1,i) + a(j,i)
enddo
```

Performance Comparison (Walltime of real work)

False Sharing



No False Sharing



* ran with 250MHz R10000 O2K, 4MB L2 cache, 490MB memory per node

Perfex Output - Counters 29 and 31 (-O0)

False Sharing

	1 cpu	4 cpus
Counter 29	3,766	482,491
		487,530
		539,327
		516,084
Counter 31	0	991,489
		1,020,035
		479,600
		535,563

No False Sharing

	1 cpu	4 cpus
Counter 29	4,643	1,638
		1,796
		2,060
		2,338
Counter 31	0	112
		110
		106
		721

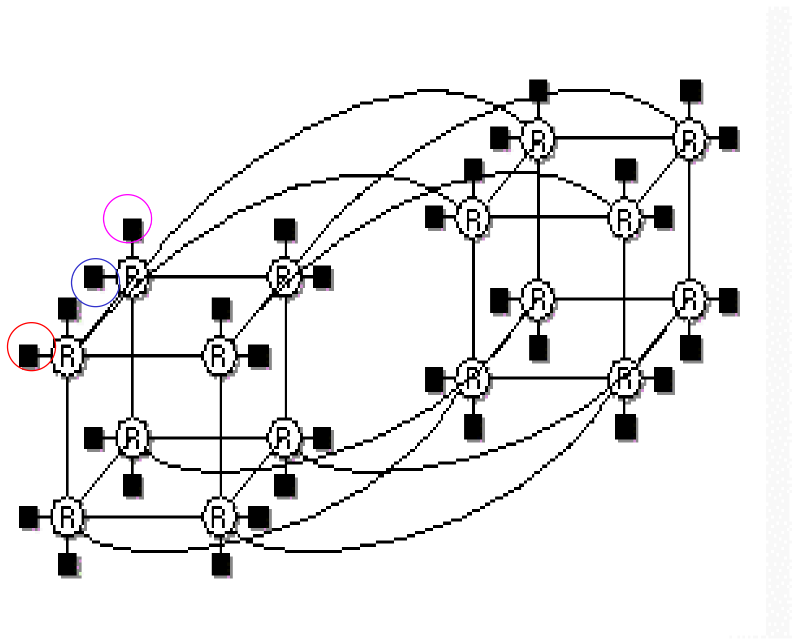
Large counts indicate false sharing is occurring

* for 4 cpus, value for each thread is reported
* ran with 250MHz R10000 O2K, 4MB L2 cache, 490MB memory per node
* results vary from run to run; but the orders of magnitude should be the same

Non-Uniform Memory Access

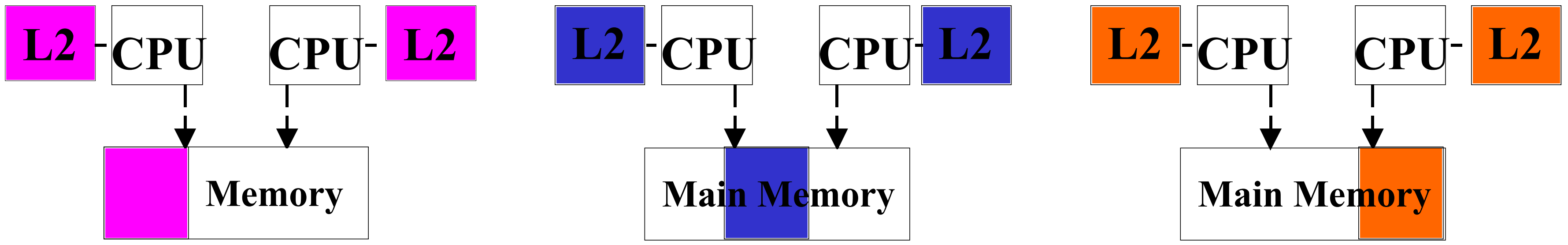
- O2K, memory access latency: ~ 485ns + 100ns/hop*

NCPUS	Maximum Router Hops	Latency
1 or 2	0	485ns
3 or 4	1	585ns
5-8	2	685ns
9-16	3	785ns
17-32	4	885ns
33-64	5	985ns



* values vary depending on system topology and hardware

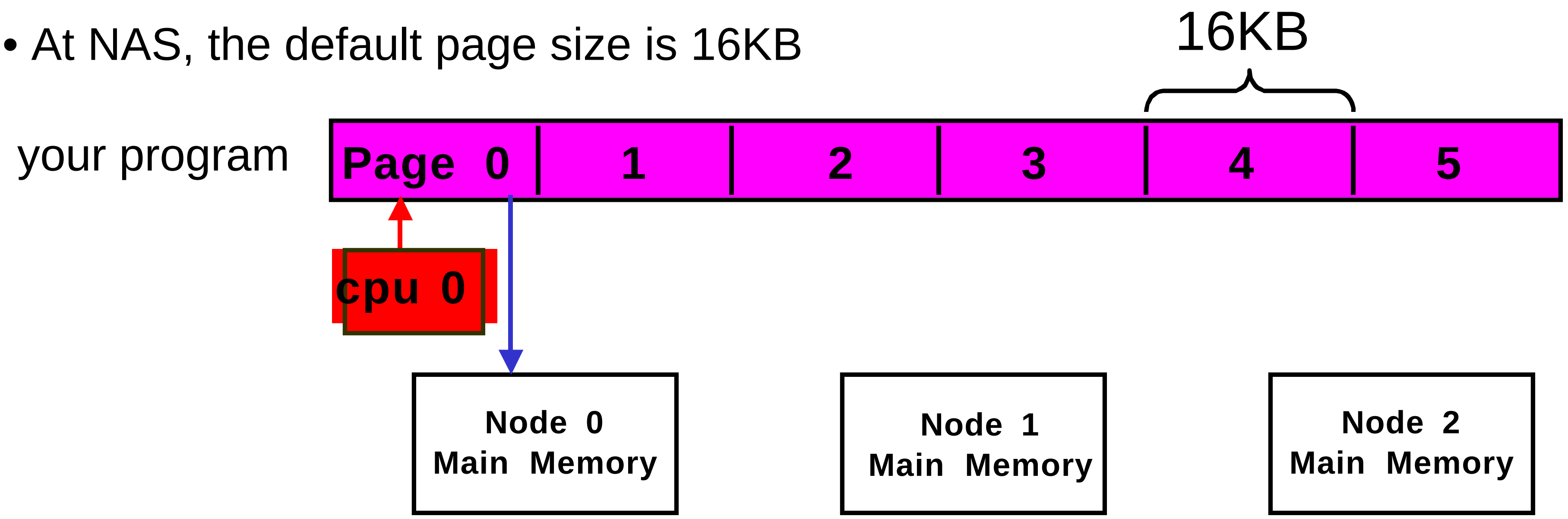
- Codes perform well when data are local to the CPU that need them



Page and Data Placement

• a page is the smallest continuous memory that
the operating system can allocate to your program

• At NAS, the default page size is 16KB



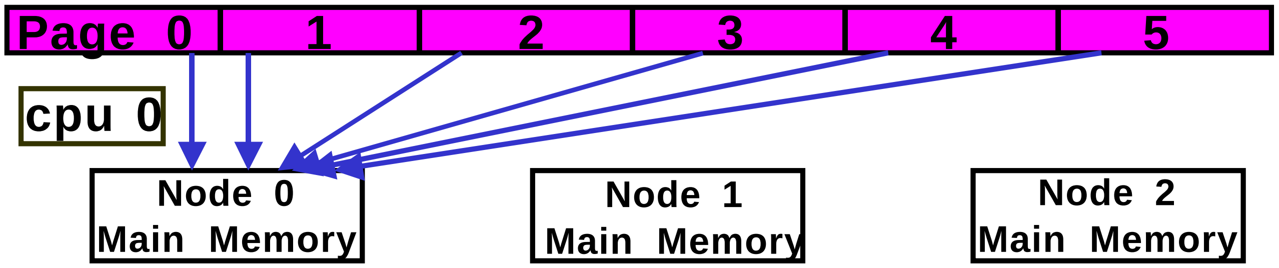
• Default Data Placement Policy - First Touch

Memory for a page is allocated in the node whose CPU(s) first touches this page; if this is not possible, it is allocated in a node as close as possible to the CPU that first touches this page

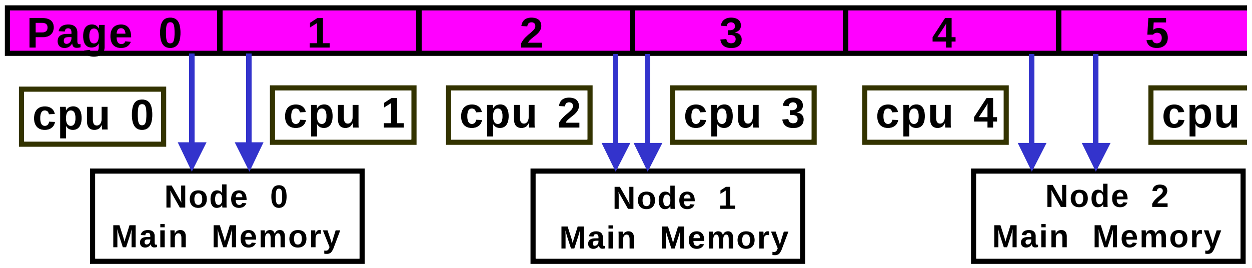
Data Initialization

First Touch:
One CPU first touches all pages

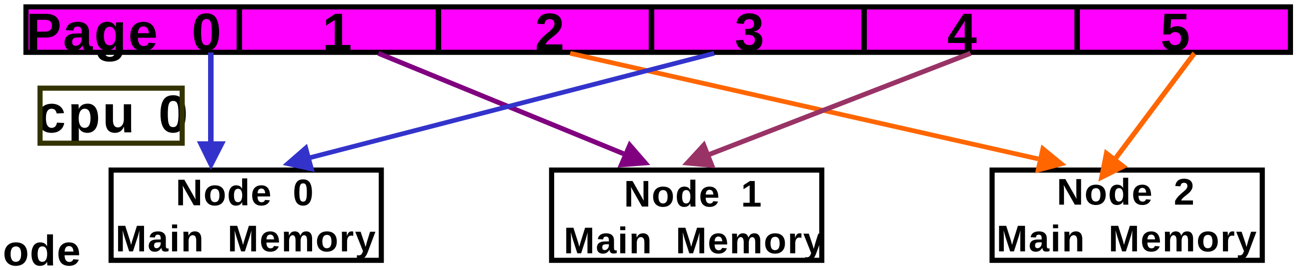
Drawbacks:
Non-local memory access bottleneck at a single node



First Touch:
Many CPUs first touch different pages and place in their local memory



Round Robin:
One CPU distributes pages to memory in many nodes in a round robin fashion; No bottleneck at a single node



Data Placement (Example 8-7, modified)

```
n= 8*1024*1024
 real*4 a(n), b(n), c(n), d(n)
←
do i = 1, n
initialize a, b, c, d
enddo
←
do i = 1, n
real work with a, b, c, d
enddo
```

Compiled with
f90 -O3 -mp

```
Nothing
!$omp parallel do shared(a,b,c,d)
```

First Touch
serial initialization

```
!$omp parallel do shared(a,b,c,d)
!$omp parallel do shared(a,b,c,d)
```

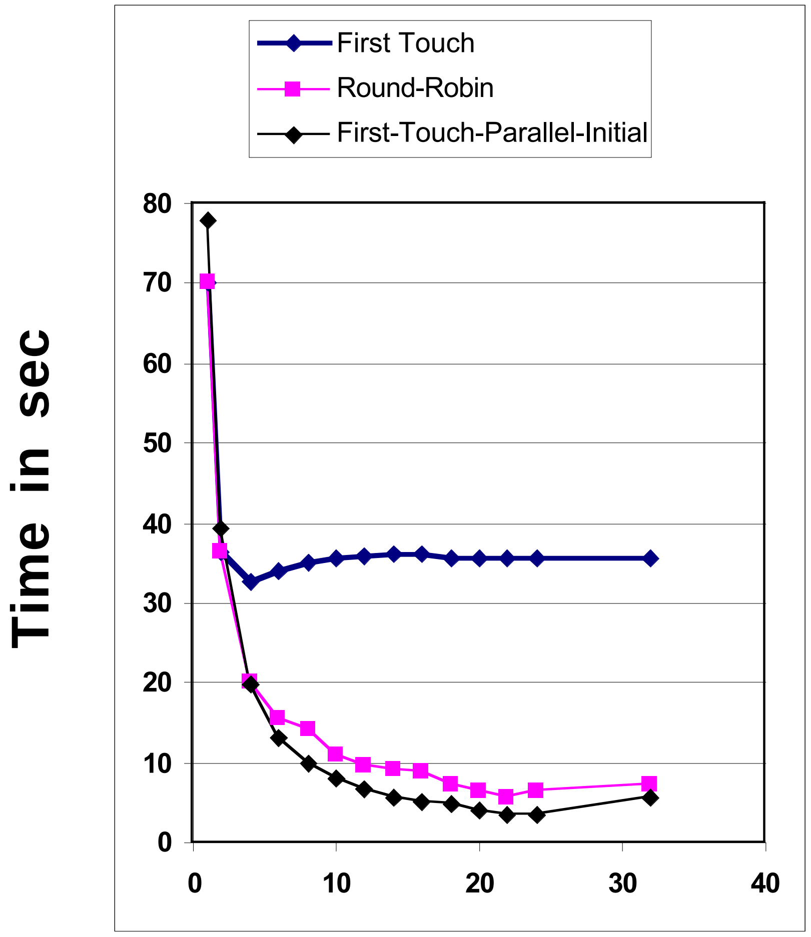
First Touch
parallel initialization

```
setenv _DSM_ROUND_ROBIN
Nothing
!$omp parallel do shared(a,b,c,d)
```

Round Robin

Performance (Walltime) Comparison

Real Work



Number of CPUs used

* ran with 250MHz R10000 O2K, 4MB L2 cache, 490MB memory per node

ssrun -numa

Output from one of the processes

```
------------------------------------------------------------------
    ...... PID 160571
------------------------------------------------------------------
Summary of NUMA memory profiling data (numa)--
 Secondary cache D misses (26): Counter Name (Number)
                          100: Counter Average Overflow
                           82: Sampled Memory Accesses
                           71: Remote Memory Accesses
                       86.585: Percent Remote Memory Accesses
                        0.866: Average ccNUMA Routing Distance
------------------------------------------------------------------
Function list, in descending order by percent remote memory accesses
------------------------------------------------------------------
Sampled     Remote    Pct Rmt    Avg Dist  Function (dso: file, line)
      1          1    100.000       1.000  libss_caliper_point

     81         70     86.420       0.864  __mpdo_MAIN__1 (*.f, 24)
```

* the data placement example with no parallelization on the initialization loop and no round robin

* ran with 6 CPUs on 300 MHz R12000 O2K, 8MB L2 cache

Suggestions for General Users

- Attending training sessions is not enough !
- Understand these topics well :
  - Origin Architecture
  - Cache (cache line, cache size, n-way set associative
    least recently use policy, cache coherence)
  - Page (page size, virtual and physical pages, TLB)
  - Data Placement
- Do hands-on experiments
- Find out why the examples do not work as expected

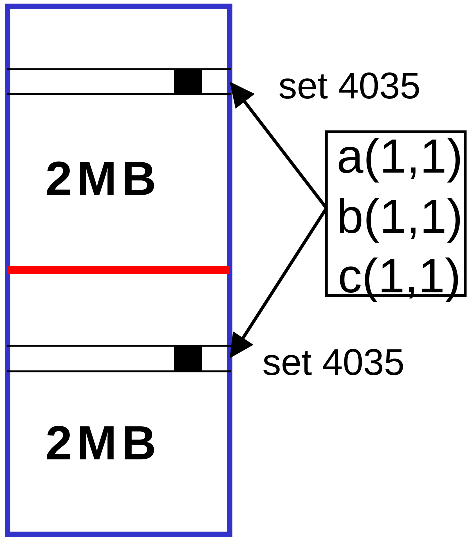
Optimization level ; cache size;etc

Quiz

L2 Cache - Quiz 1

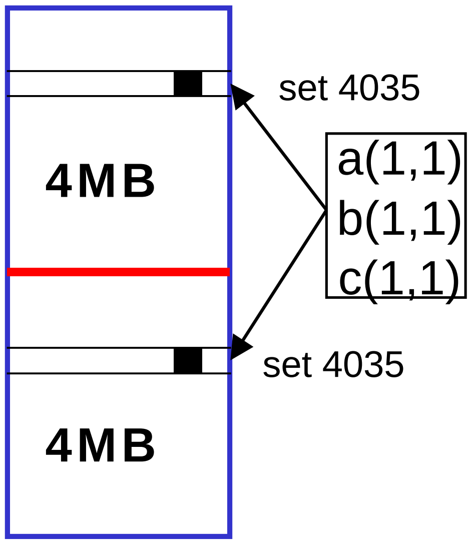
real*4 a(1024,512), b(1024,512), c(1024,512)

If L2 cache size is 4MB and two-way set associative; L2 cache line is 128B; a(i,j), b(i,j), c(i,j) will map to the same cache set

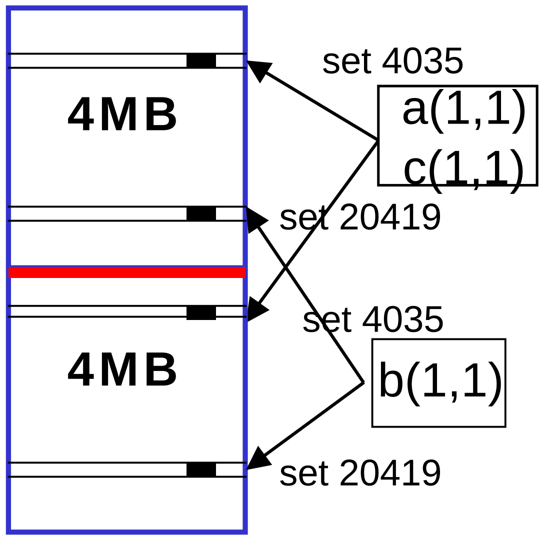


If L2 cache size is changed to 8MB will a(i,j), b(i,j), c(i,j) still map to the same cache set? A or B ?

(A)



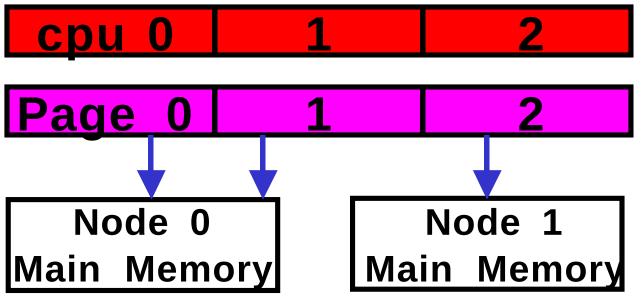
(B)



Page and Data Placement - Quiz 2

OMP_NUM_THREADS 3

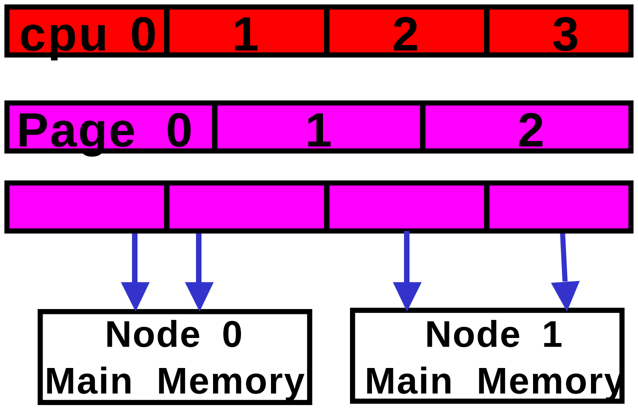
```
real a( 3 pages )
!$OMP PARALLEL DO
do i = 1, n
initialize a(i)
enddo
```



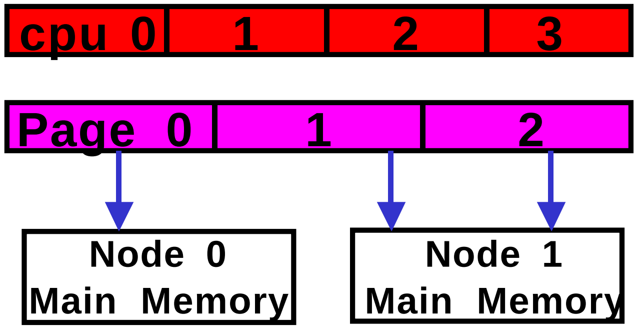
*** simple schedule type is used**

If OMP_NUM_THREADS 4
Which one (A or B) is correct ?

(A)



(B)



Acknowledgement

- Bron Nelson - SGI on-site analyst
- NASA Advanced Supercomputing Division
  - Managers : Alan Powers, Chuck Niggley
  - HEC Staff : HSP, Control Room, SciCon
- CUG