



Special Effects

IBM
July, 2006

Agenda

- **Large and Medium Pages**
- **Memory Affinity**
- **Processor binding**
- **Simultaneous Multi Threading**

VMM – Virtual Memory Manager

- **VMM separates physical memory from application address space**
 - **Advantages:**
 - Can use addresses greater than physical RAM
 - Can block out sections of memory for special purposes
 - **Disadvantages:**
 - If addresses greater than physical RAM swap device used
 - Memory bandwidth \gg 1 Gbyte/s; swap device bw \sim 20 Mbyte/s
- **Unit of memory = page**
 - Various page sizes
 - Default page size = 4096 bytes
 - Only page size for POWER through POWER3.

Large and Medium Pages Benefits

- **Enhance memory bandwidth**
 - Prefetch performance is limited by page size
- **Enhance Translation Lookaside Buffer (TLB) coverage**
 - If memory page information is in TLB: zero cost translations
 - If memory page information is not in TLB: wait for info
- **TLB sizes**
 - **POWER3: 128 or 256 TLB entries**
 - Small page coverage: 512 or 1024 Kbyte
 - No large pages
 - **POWER4, POWER5: 1024 TLB entries**
 - Small page coverage: 4 Mbyte
 - 16 Mbyte large pages
 - **POWER5+: 2048 TLB entries**
 - Introduction of 64 Kbyte medium pages

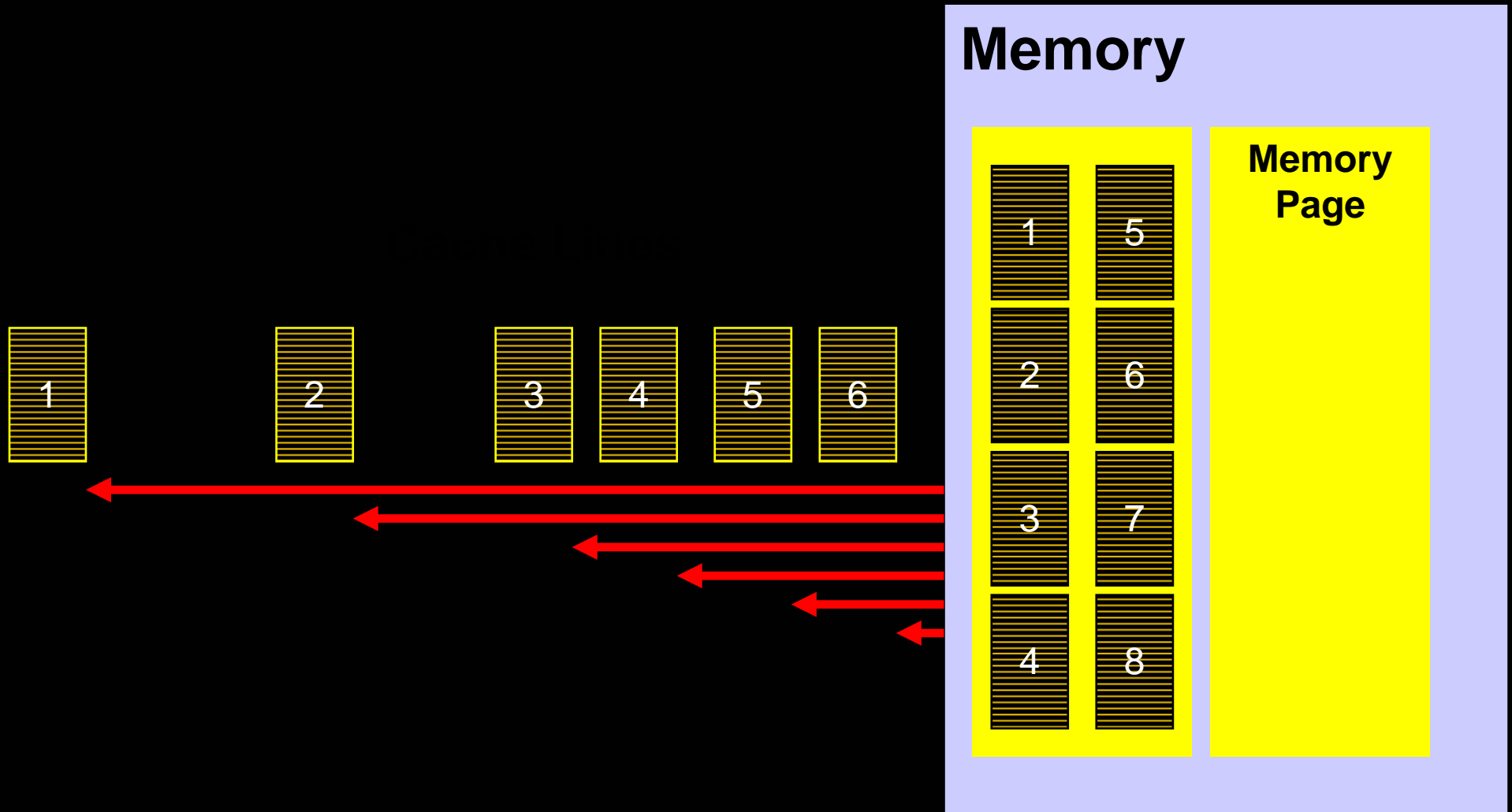
Large and Medium Pages Benefits

Stride 1

- Enhance memory bandwidth
 - Prefetch performance is limited by page size
- Enhance Translation Lookaside Buffer (TLB) coverage
 - If memory page information is in TLB: zero cost translations
 - If memory page information is not in TLB: wait for info
- TLB sizes
 - POWER3: 128 or 256 TLB entries
 - Small page coverage: 512 or 1024 Kbyte
 - No large pages
 - POWER4, POWER5: 1024 TLB entries
 - Small page coverage: 4 Mbyte
 - 16 Mbyte large pages
 - POWER5+: 2048 TLB entries
 - Introduction of 64 Kbyte medium pages

Indirect
addressing
or large
strides

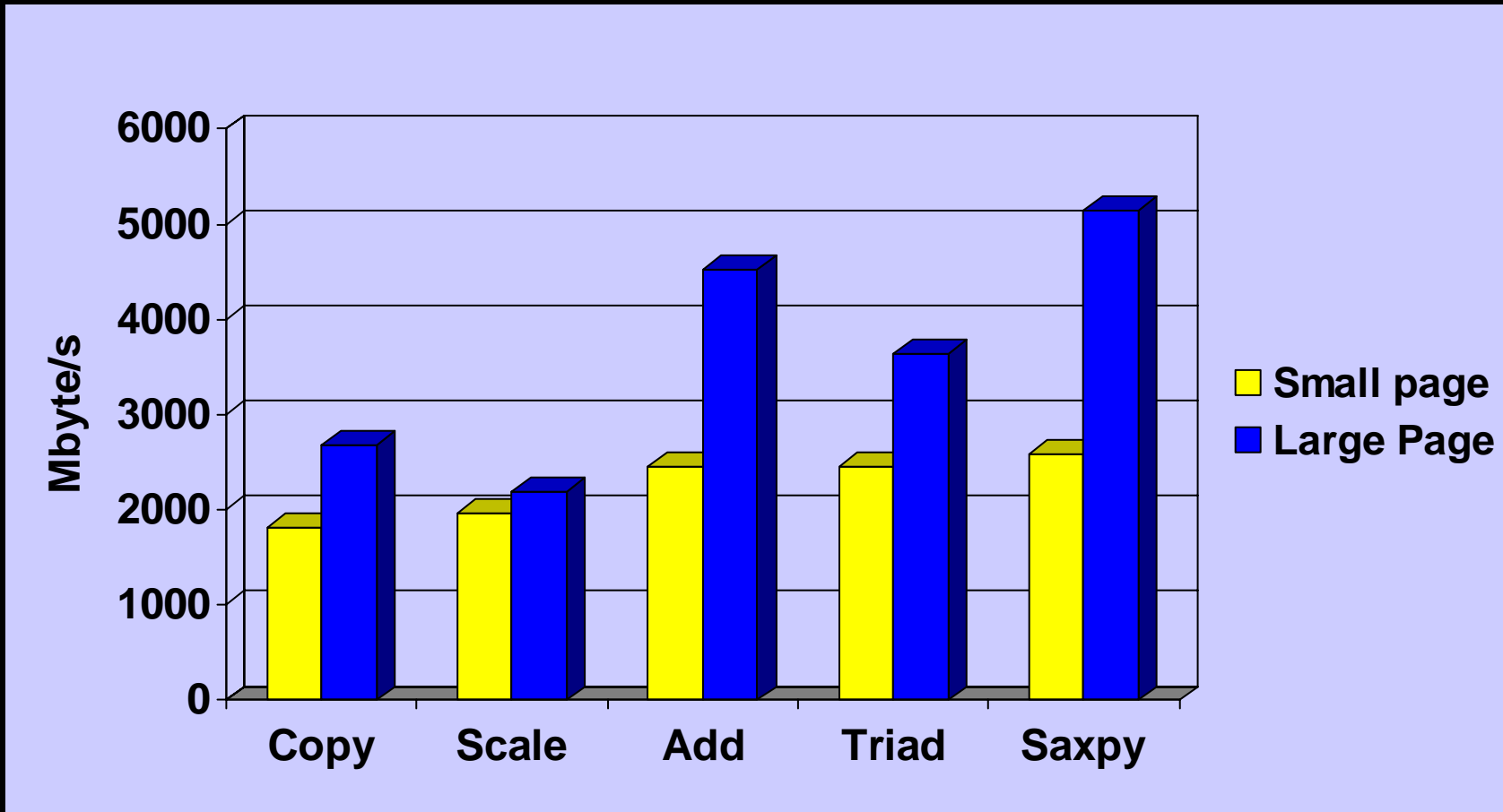
Prefetching



Prefetch

- **Prefetch ends at page boundary**
 - Location of next cache line not known by hardware
- **Small pages (4096 bytes):**
 - 32 cache lines
 - Frequent startup; cannot achieve asymptotic speeds for stride 1
- **Large pages (16 Mbyte):**
 - 131072 cache lines
- **Medium pages (64 Kbyte)**
 - POWER5+ only
 - 512 cache lines
 - Large enough to reach asymptotic performance

Large Pages: Bandwidth Enhancement



POWER5 1.45 GHz

POWER5+: TLB Coverage

	Page Size	TLB Entries	Memory Coverage
Small Pages	4096 byte	2048	8 Mbyte
Medium Pages	64 Kbyte	2048	128 Mbyte
Large Pages	16 Mbyte	2048	32 Gbyte

- If memory page information is in TLB: zero cost translations
- If memory page information is not in TLB: wait for info

Large Pages: Configuration

- **Large pages are allocated statically at boot-time**
 - **\$ vmo ... -**
- **Or dynamically with AIX 5.3**
- **Need small pages for AIX and other jobs**
 - **Recommendation:**
 - **No more than 85% of memory in Large Pages**

Large Page Verification

```
$ vmstat -l # Small "L"
```

kthr		memory ...		cpu				large-page	
r	b	avm	fre	us	sy	id	wa	alp	flp
1	0	18411462	14344396	296	2	0	98	12	4084
0	0	18411463	14344395	2	0	98	0	12	4084

alp: Allocated Large Pages

flp: Free Large pages

Large Pages: Security (access)

- **Administrator (security group) needs to validate user for Large Page usage**
 - **Require**
 - **CAP_BYPASS_RAC_VMM**
 - **CAP_BYPASS_PROPOGATE**
 - **\$ chuser capabilities=**
 - **CAP_BYPASS_RAC_VMM,CAP_PROPOGATE**
 - **Use lsuser to see if large pages permitted**
 - **\$ /usr/sbin/lsuser {user_id}**
- **Or make it default within /etc/security/user file**

Large Page Usage

- **Loader and Idedit:**
 - `$ xlf -blpdata -o a.out`
 - `$ /usr/bin/ldedit -blpdata a.out`
- **Environment variable (NOT RECOMMENDED!!):**
 - `$ LDR_CNTRL=LARGE_PAGE_DATA={Y,N,M}`
 - **Y: Yes ("Advisory") mode**
 - Use large pages if available
 - This mode used by loader and ldedit
 - **N: No large pages**
 - **M: Mandatory**
 - Do not run if large pages not available
 - **“env LDR_CNTRL=LARGE_PAGE_DATA=M date” to see if large pages are working for you**

Large Page System Resource Usage

- **Large pages are accessed in increments of a memory Segment**
 - Segment = 256 Mbyte = 16 large pages.
- **MPI example, 8 tasks, each task allocates 300 Mbytes**
 - Each task will get 2 segments if using large pages
 - If using small pages, usage = 2.4 Gbytes
 - If using large pages, usage = 4.0 Gbytes
- **If using whole node, not as big of an issue**

Large Page Summary

- **Set up memory pools**
 - Decide amount of memory in Small and in Large pages
 - Administrator:
 - vmo
 - chuser capabilities= ...
- **Tag binary:**
 - {xlf,xlc} ... -blpdata -o a.out
 - ledit ... -blpdata a.out
 - \$ksh LDR_CNTRL=LARGE_PAGE_DATA=M a.out

Medium Page Summary

- Performance is same as large pages for most applications
- Allocated dynamically by system as needed
- Different Idedit options:
 - Idedit -bdatapsize=64K a.out
 - Idedit -bstackpsize=64K a.out
 - Idedit -btextpsize=4K a.out
- None of the administrative headaches that large pages can cause

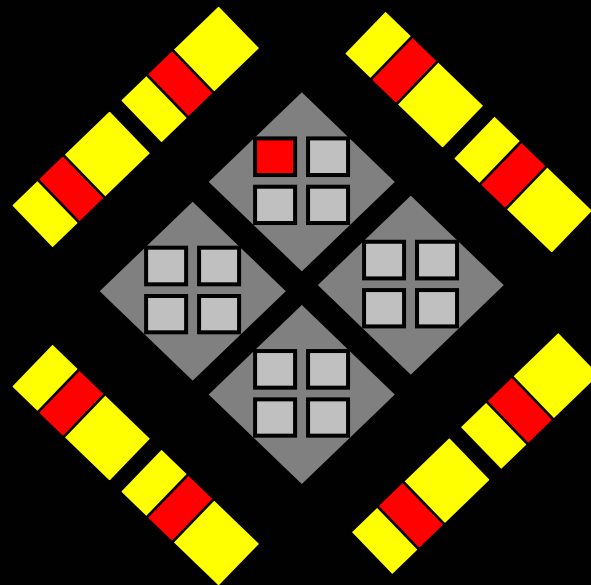
Memory Affinity

Memory Affinity

- **Page Distribution**
 - **Access to local boards is robust**
 - **Little contention (except for processor pairs on chip)**
 - **Lower latency (slightly)**
- **Access to remote modules uses busses**
 - **Contention**
 - **Higher latency (depends on node, ~10%)**

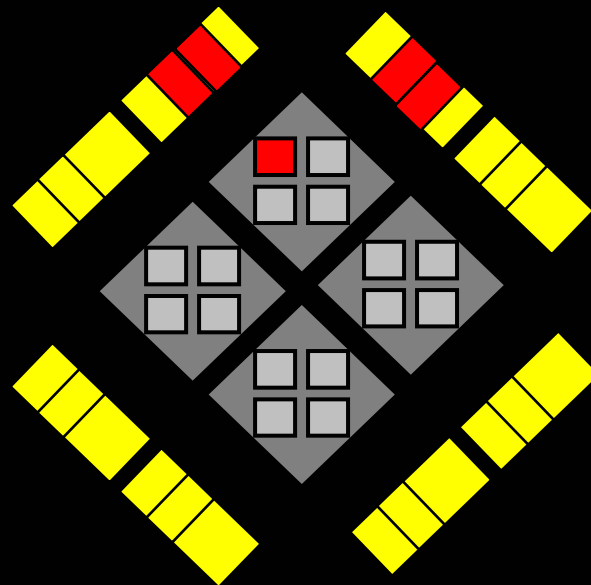
Memory Allocation

- Pages are allocated by module (p590, p595)
- Pages are allocated by SMI (p575)
- Approximately uniform distribution
- Approximately round robin



Memory Allocation

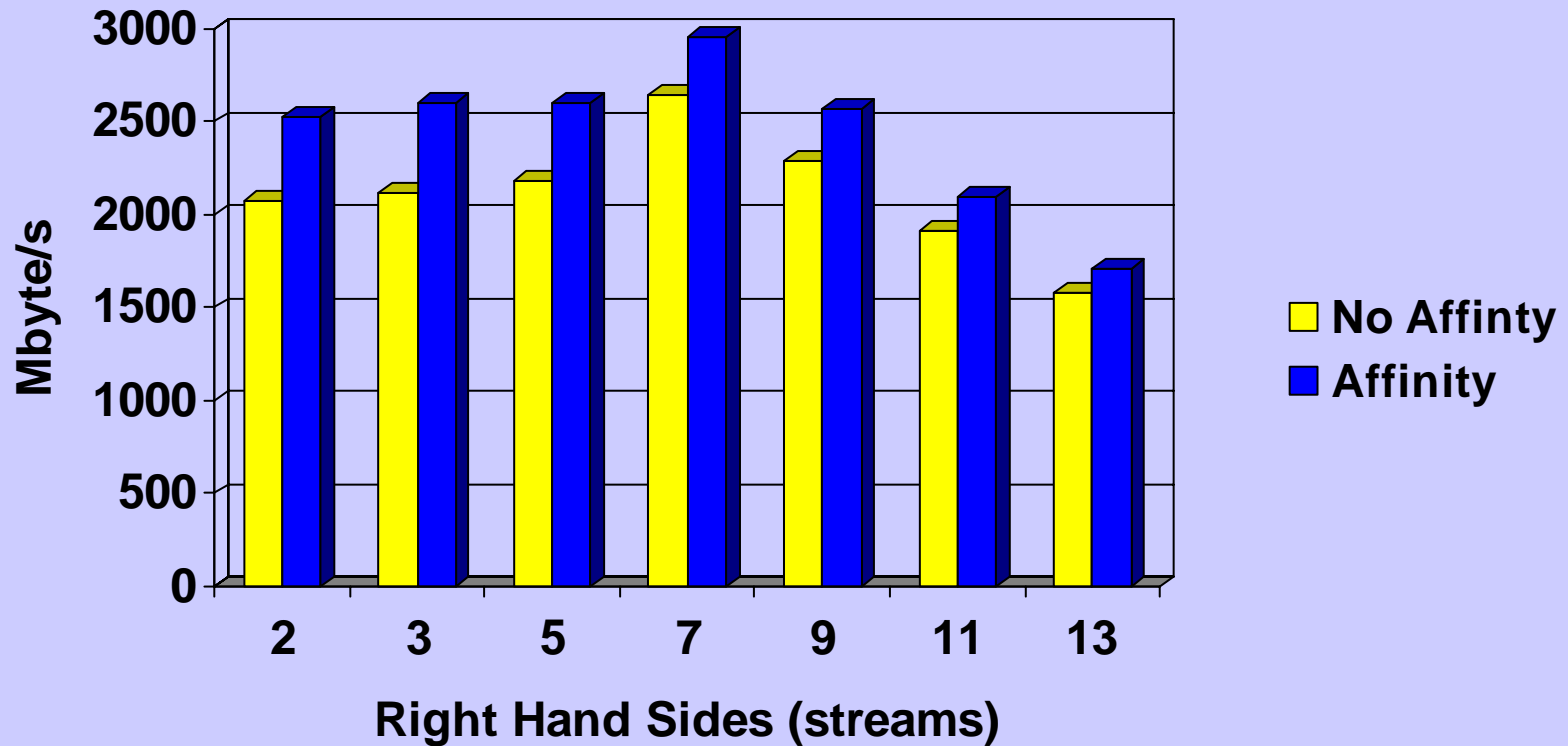
- **Memory Affinity**
 - Allocate pages on memory local to module/SMI
 - Environment variable:
 - `export MEMORY_AFFINITY=MCM`



Memory Affinity

- **Less system-wide contention**
 - Works well for MPI
 - Memory localization
- **Difficulty with threads**
 - Threads use shared memory
 - New threads may require references to remote memory
 - "First touch" strategies

Memory Affinity: Bandwidth



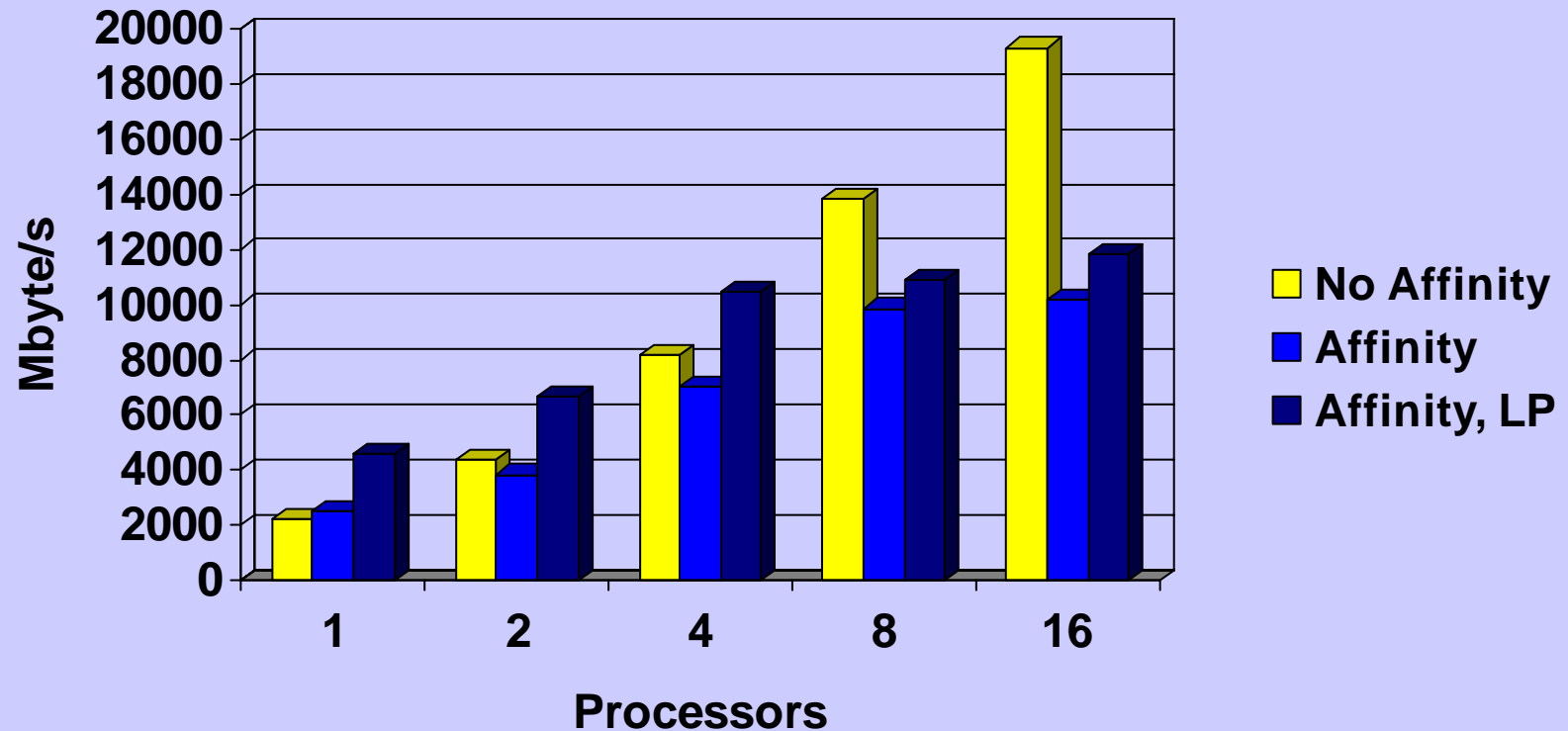
Memory Affinity and SMP

- **Desire memory pages on same module as process**
- **Memory allocated ~ first touch**
- **Spawned process often access shared memory**
- **Shared memory allocated by earlier process**
- **No locality**
- **Try "first touch" strategy**

Process binding

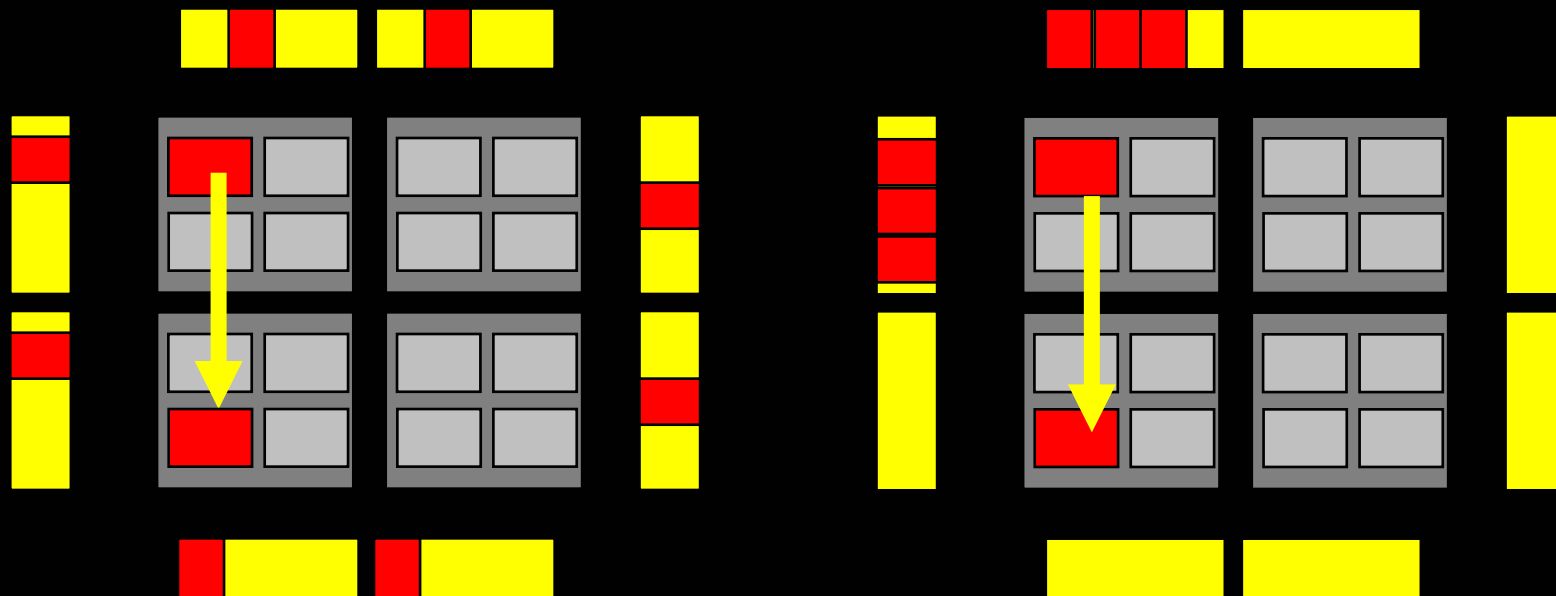
- **Can force an AIX process to run exclusively on logical processor**
 - Benefits cache reuse and memory affinity
- **Similar to LINUX 2.6 “taskset” utility**
- **CAUTION: If more than one process binds to same logical processor performance will be > 2X worse!**
- **Useful if application has exclusive access to node**
 - **#@ node_usage = not_shared**

Memory Affinity and SMP



Why Process Binding?

- **Processes migrate**
 - AIX has concept of “affinity”, but things happen...
 - Intended to keep caches coherent
 - **Conflicts with MEMORY_AFFINITY=MCM**
 - Desire to keep process and memory together



Process Binding

- **Bindprocessor utility**
 - **bindprocessor -q**
 - 0 1 2 3 ... 127
 - **bindprocessor {PID} {processor number}**
- **Scripting:**
 - **a.out &**
 - **ps | grep {pid}**
 - ...
- **bindprocessor {pid[i]} {processor}**

Process Binding

- **SMP Run Time Environment (RTE)**
 - Threads run time library
 - Export `XLSMPOPTS=startproc={}:stride={}`

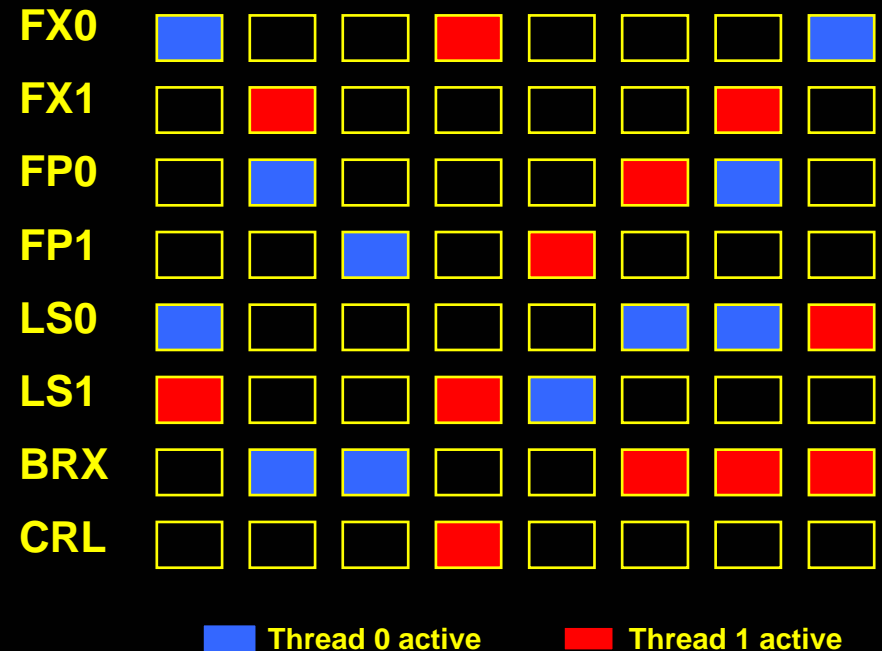
Prebinding

- **“launch”**
 - setenv ...launch environment for stride/pattern...
 - poe launch a.out
 - launch binds to processor and fork/execl a.out
- **Other similar tools becoming available**

Simultaneous Multi-Threading

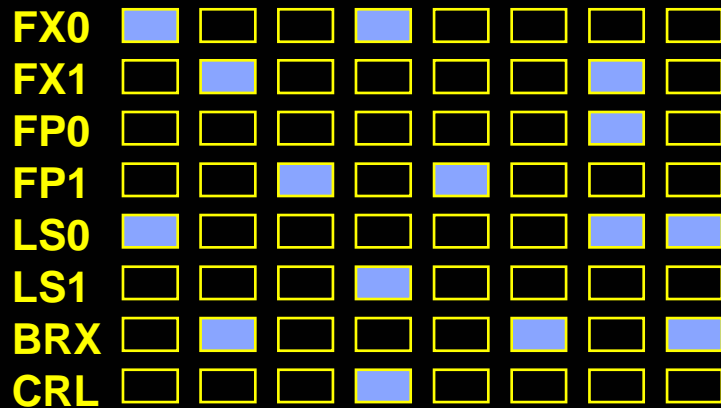
- Each chip appears as a 4-way SMP to software
 - 2 processors
 - 2 threads per processor
- Processor resources optimized for enhanced SMT performance
- Software controlled thread priority
 - Dynamic feedback of runtime behavior to adjust priority
- Dynamic switching between single and multithreaded mode

Simultaneous Multi-Threading

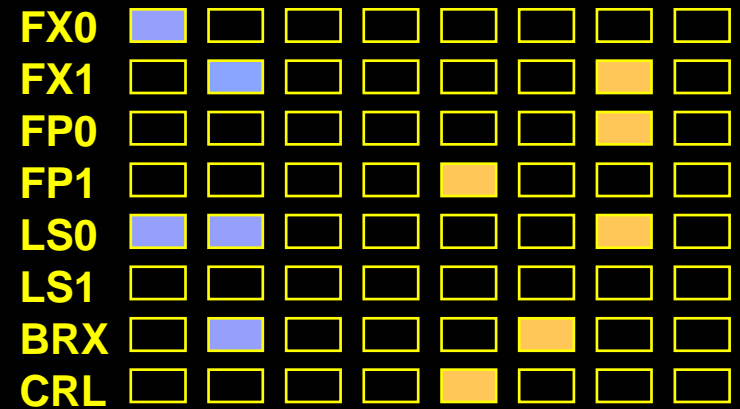


Multi-threading Evolution

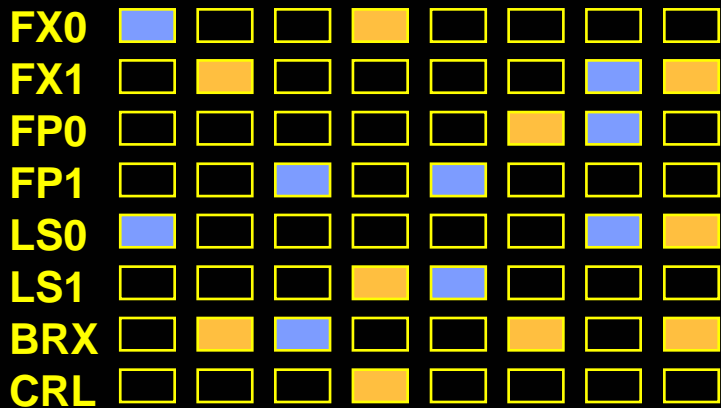
Single Thread



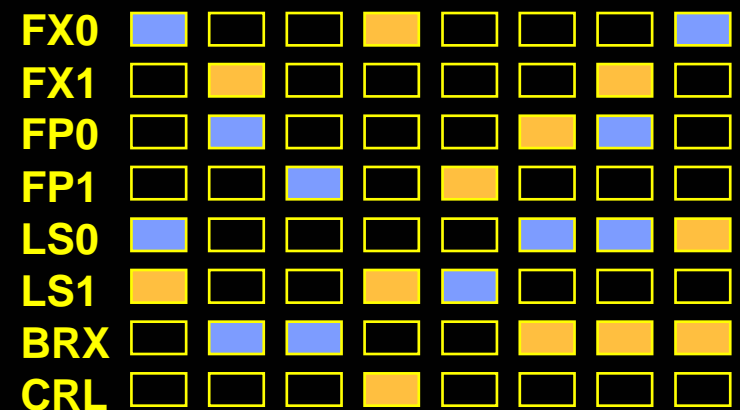
Coarse Grain Threading



Fine Grain Threading



Simultaneous Multi-Threading



■ Thread 0 Executing

■ Thread 1 Executing

□ No Thread Executing

SMT experience

- **SMT gives -2% to +30% performance boost for typical SMT MPI codes in benchmarking environment**
 - **Run 8-way MPI job without SMT**
 - **Run 16-way MPI job with SMT**
 - **16-way job with SMT is typically 10-30% faster**
- **If reaching scaling limits, then SMT will not help performance**
- **Licensing costs may be issue for 3rd party software**

Summary

- **Large and Medium Pages**
 - Useful for bandwidth enhancement and “gather/scatter”
- **Memory affinity:**
 - Useful for MPI
 - Not as useful for OpenMP
- **Process binding**
 - Useful – if you know what you are doing
 - Need exclusive access to node
- **SMT**
 - May give 20% or more for typical MPI code
 - May be very useful in diverse environment

Auxiliary