# Special Effects

ScicomP 11

Charles Grassl
IBM
May, 2005

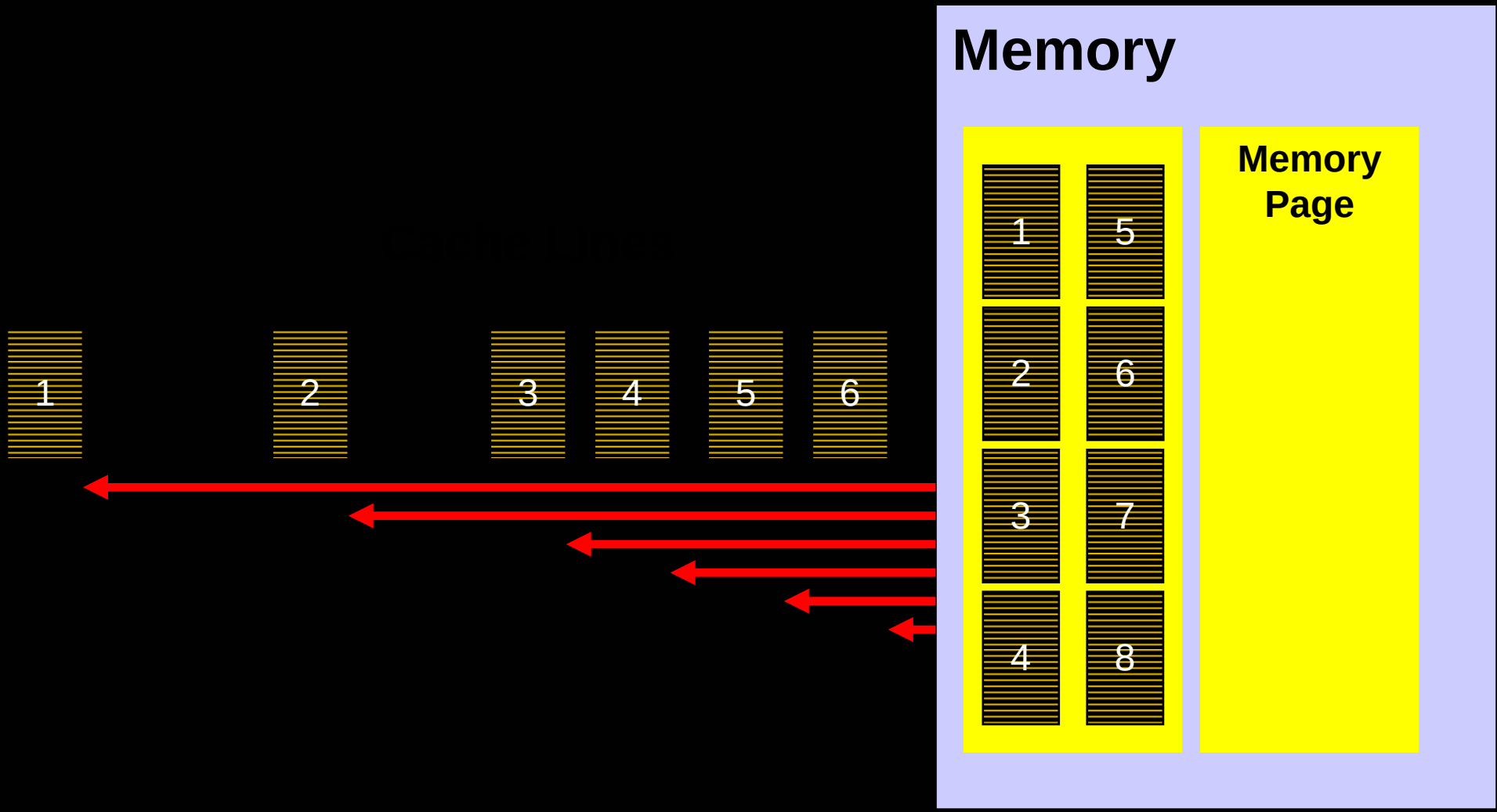
# Agenda

- Large Pages
- Memory Affinity
- Processor binding
- Simultaneous Multi Threading

# Large Pages

- Enhance memory bandwidth
  - Prefetch performance is limited by page size
- Enhance Translation Lookaside Buffer (TLB) coverage
  - POWER3: 128 TLB entries
    - Small page coverage: 512 kbyte
  - POWER4, POWER5: 1024 TLB entries
    - Small page coverage: 4 Mbyte

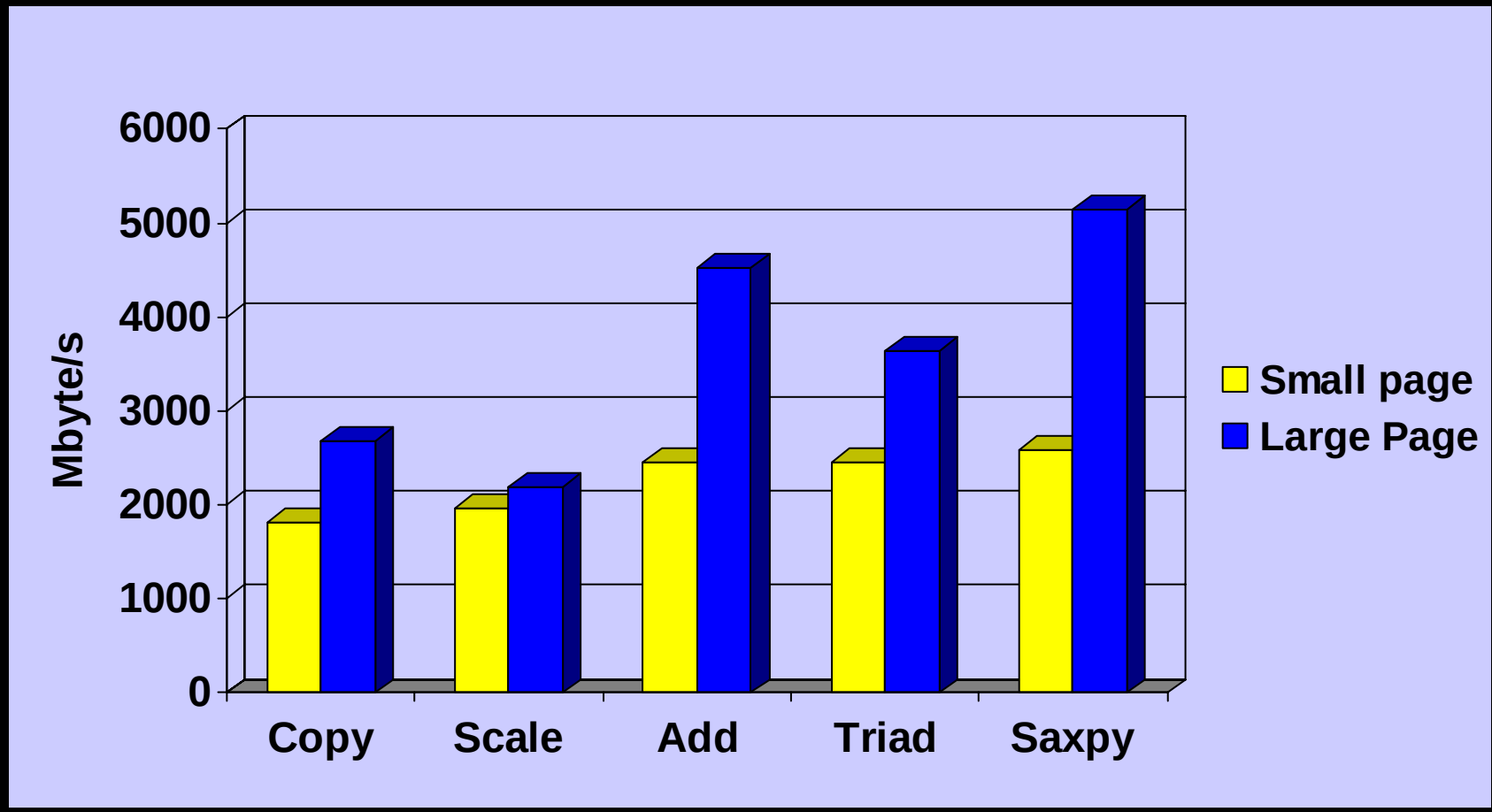
# Prefetching



# Prefetch

- Prefetch ends at page boundary
  - Location of next cache line not known by hardware
- Small pages (4096 bytes):
  - 32 cache lines
    - Frequent startup
- Large pages (16 Mbyte):
  - 131072 cache lines

# Large Pages: Bandwidth Enhancement



# Large Pages: TLB Coverage

| CPU | Page Size | TLB Entries | Memory Coverage |
|---|---|---|---|
| POWER3 | 4096 byte | 128 | 500 kbyte |
| POWER4, POWER5, Small Pages | 4096 byte | 1024 | 4 Mbyte |
| POWER4, POWER5, Large Pages | 16 Mbyte | 1024 | 16 Gbyte |

# Large Pages

- POWER4 supports two sizes of pages:
  - 4096 bytes  (Default)
  - $2^{14}$ (~16 million) bytes
    - Also, $2^{18}$ (~256 million) bytes (legacy support)
    - Actually treated as 16 Mbyte pages
- Expect single CPU bandwidth to improve
  - Prefetch runs out of room with 4096 byte pages
  - Cannot prefetch across pages
  - Bandwidth to increase 5-20%
- Large pages detrimental to forks

# Large Pages: Configuration

- Large pages are allocated statically at boot-time
  - $ vmo … -
- AIX “limits” size of large page pool:
  - 85% of memory maximum

# Large Page Verification

- $ vmstat –l # Small “L”
- …..

$ vmstat -l 1

System configuration: lcpu=64 mem=128000MB

```
kthr    memory              page              faults       cpu    large-page
----- ----------- ------------------------ ------------ ----------- -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa   alp   flp
 1  0 18411462 14344396   0   0   0   0    0   0   8  242 296  2  0 98  0    12  4084
 0  0 18411463 14344395   0   0   0   0    0   0  11   22 299  2  0 98  0    12  4084
```

# 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}

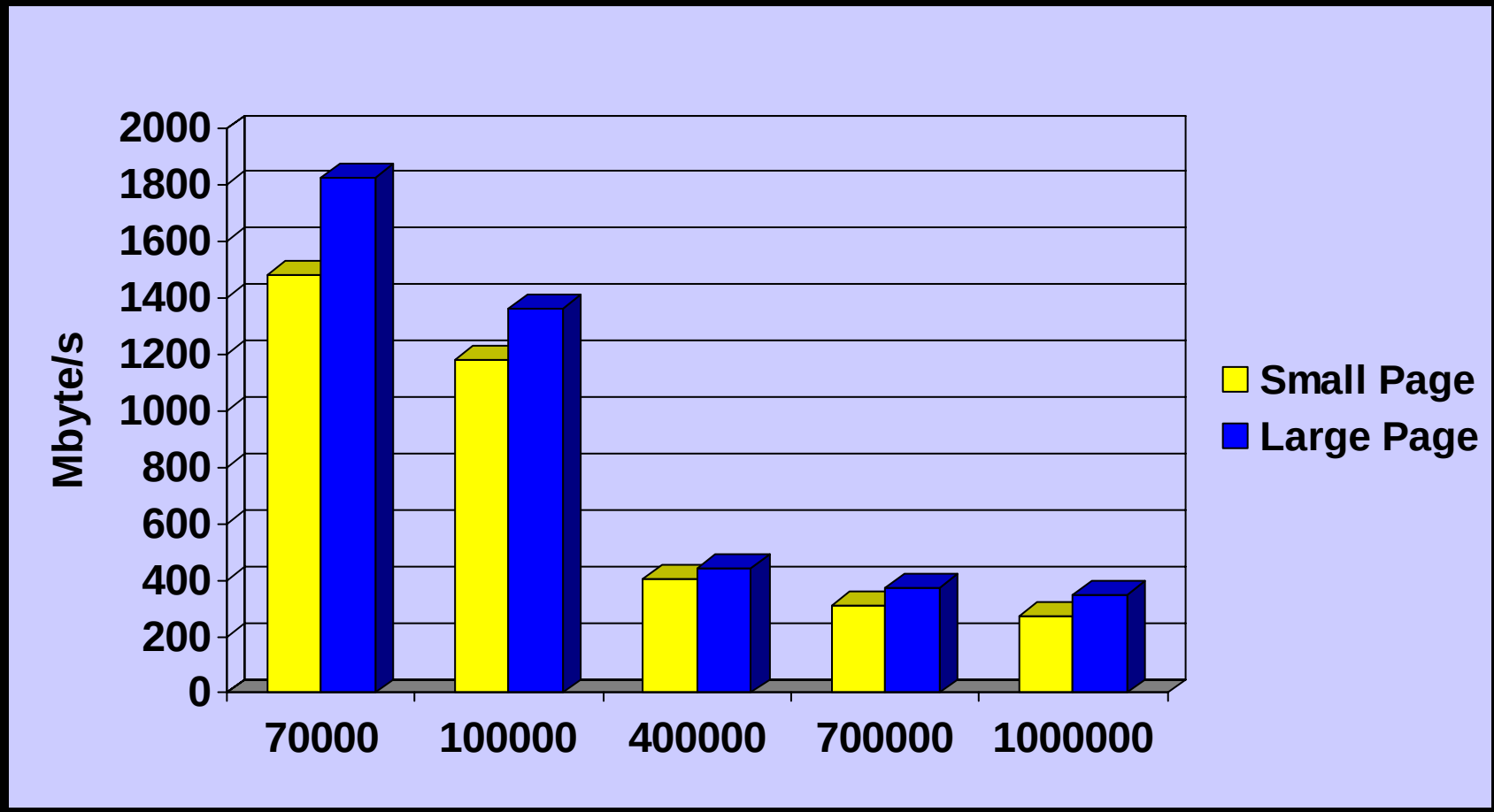
# Large Page Usage

- Loader and ldedit:
  - $ xlf .... -blpdata  -o a.out
  - $ /usr/ccs/bin/ldedit -blpdata a.out
- Environment variable:
  - $  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

# Large Page Usage: TLB Pressure

- Large memory applications
  - Greater than 4096 byte/pate * 1024 TLB entries
  - Gather - Scatter algorithms
  - Sparse matrix
- Large pages extend TLB coverage to 16 Gbyte

# Large Page: Gather



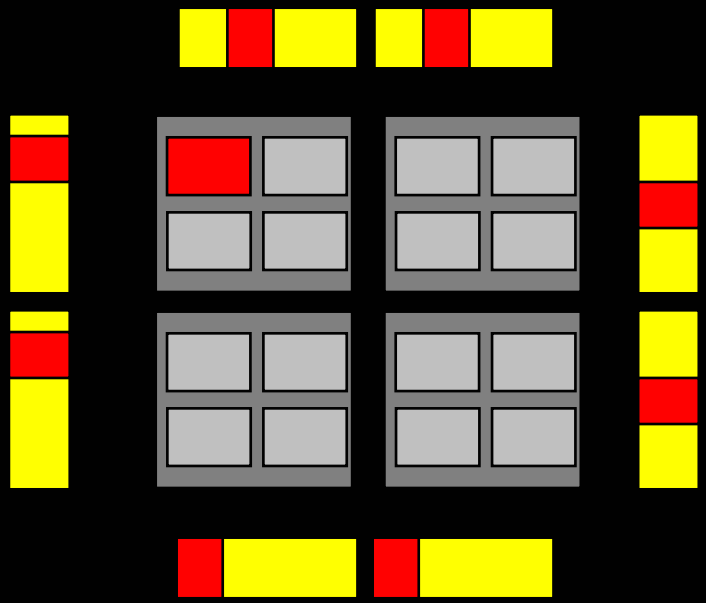

S = s + A(index(i))

# Memory Affinity

- Page Distribution
- Access to local boards is robust
- Little contention (except for CPU pairs on chip)
- Lower latency
- Access to remote modules uses buses
- Contention
- Slightly higher latency (~10%)

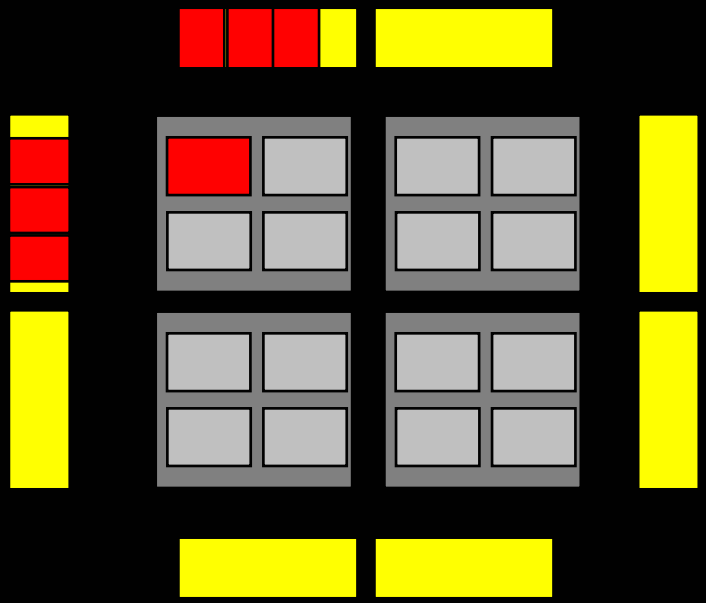
# Memory Allocation

- Pages are allocated by module
- Approximately uniform distribution
- Approximately round robin



# Memory Allocation

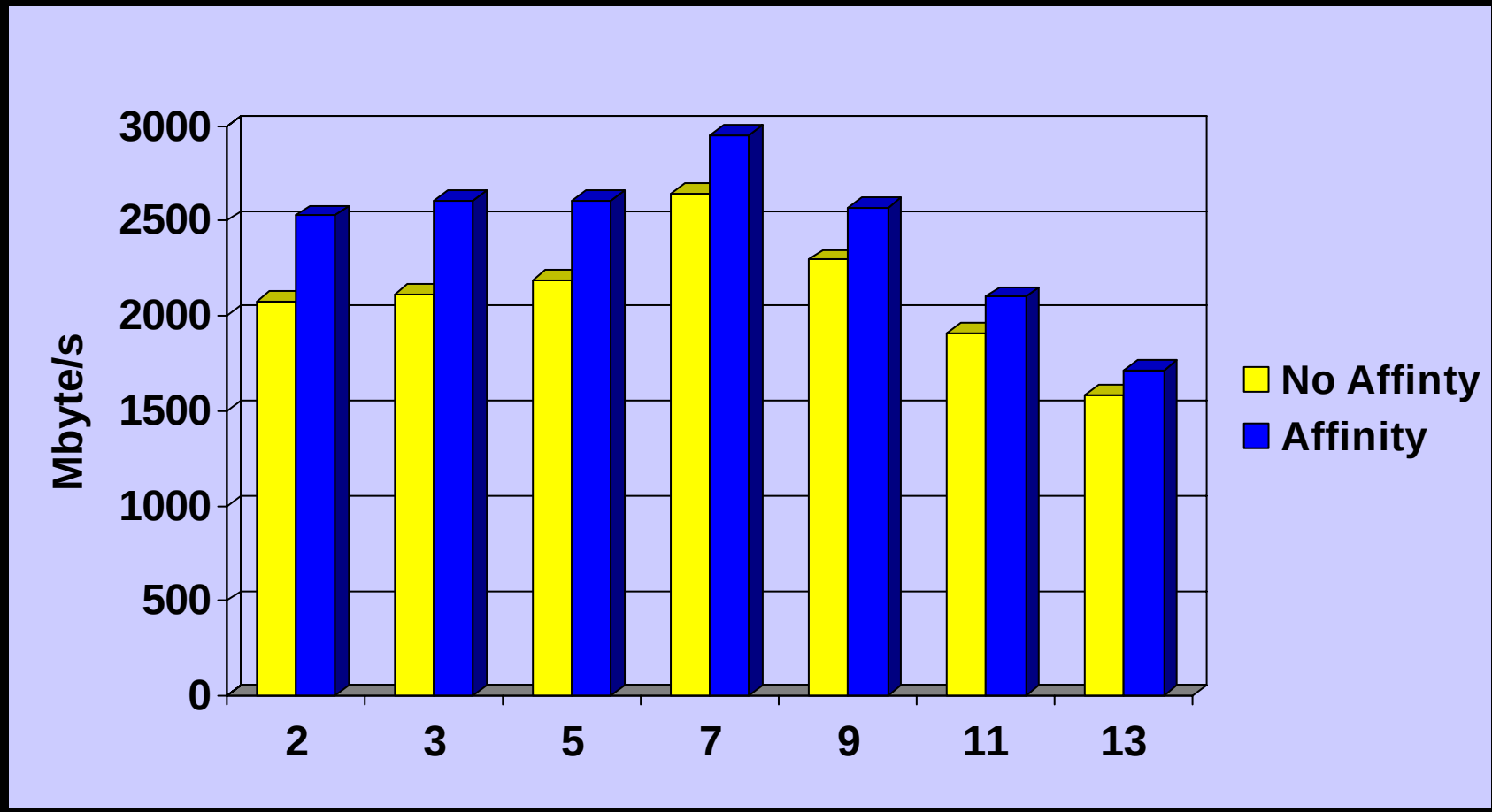
- Memory Affinity
  - Allocate pages on memory local to module
- $ MEMORY_AFFINITY=MCM



# Memory Affinity

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

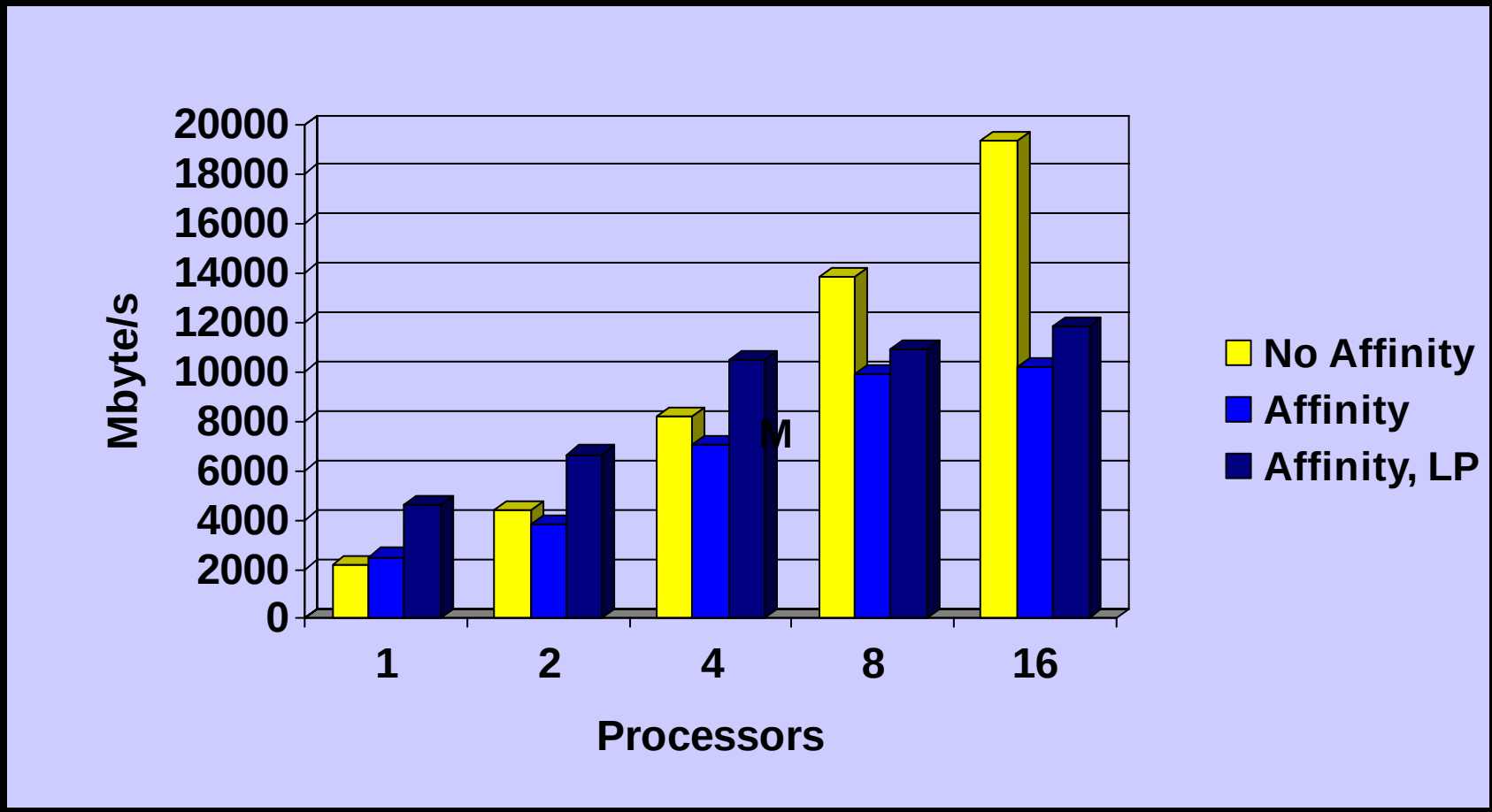
# Memory Affinity: Bandwidth



# Memory Affinity and SMP

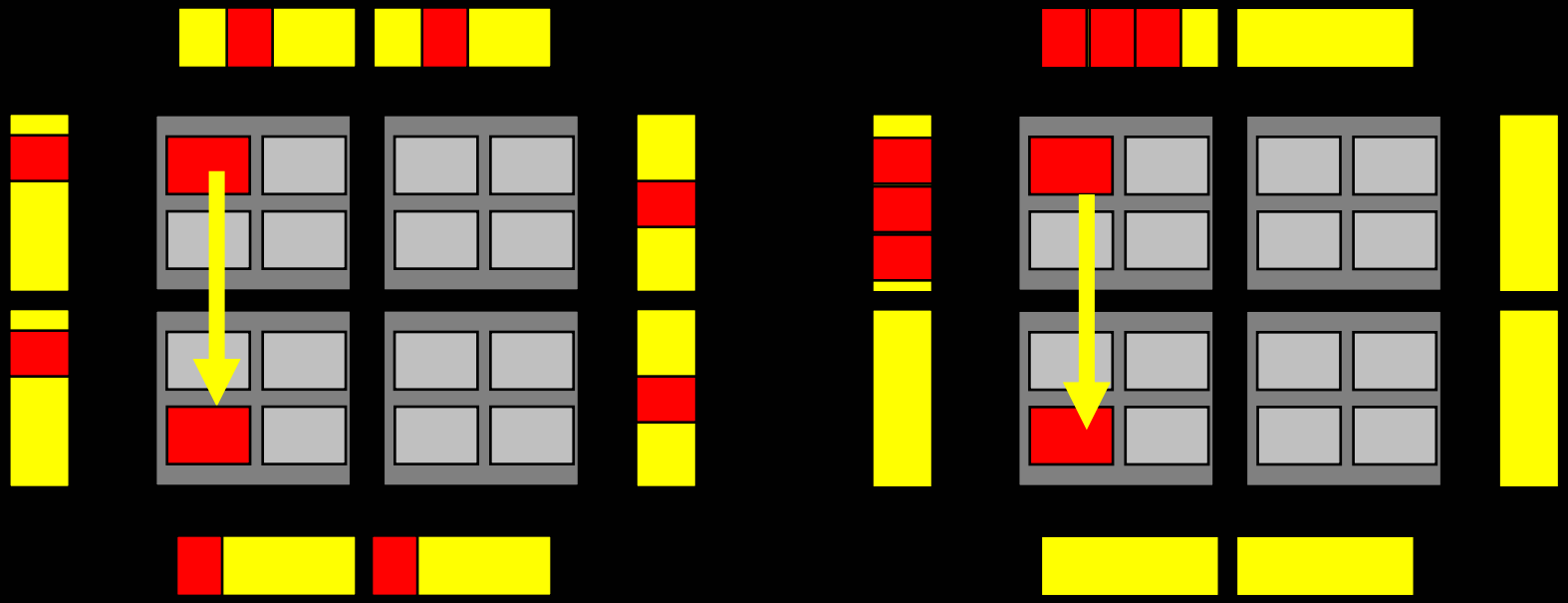
- Desire memory pages on same module as process
- Memory allocated ~ first touch
- Spawned process often access shared memory
- Shared memory allocated by earlier process
- No locality
- Try "first touch" strategy

# Memory Affinity and SMP



# Process Binding

- Processes migrate
  - AIX has concept of “affinity”
    - Intended to keep caches coherent
  - Trouble with memory affinity
    - 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}
  - …
  - binprocessor  {pid[i]} {processor}

# Process Binding

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

# Process Binding

- Library
  - C
    - void bindprocessor(…)
  - Binding libraries (Not supported)
    - libbindUtils.a

      $(CC) … -l {bindOMP, bindMPI, bindMPIOMP}

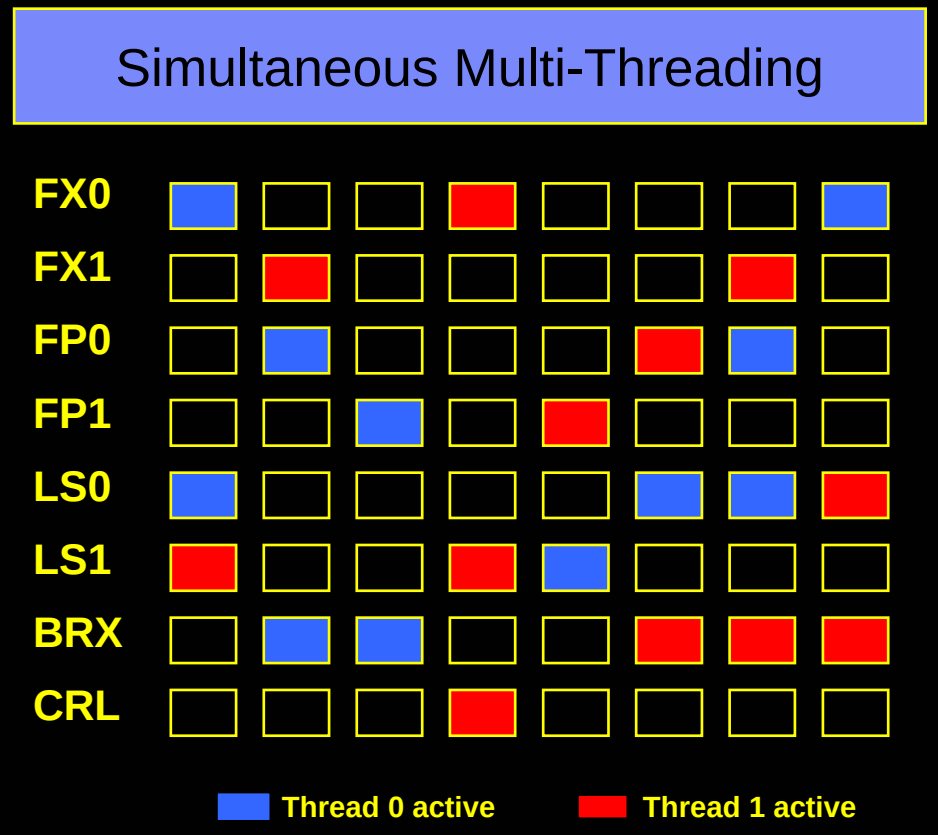
      export CPU0={}  Delta={}
    - libbindUtils2.a

      $(CC) … -l {bindOMP, bindMPI, bindMPIOMP}

      Export TARGET_ALL="0 2 4 6"

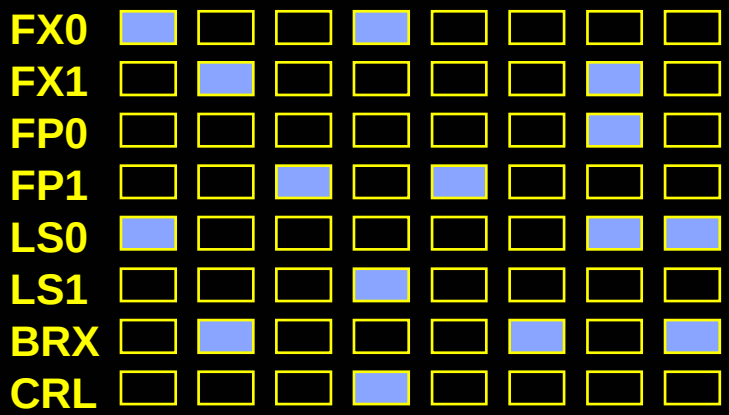
# Simultaneous Multi-Threading

- 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

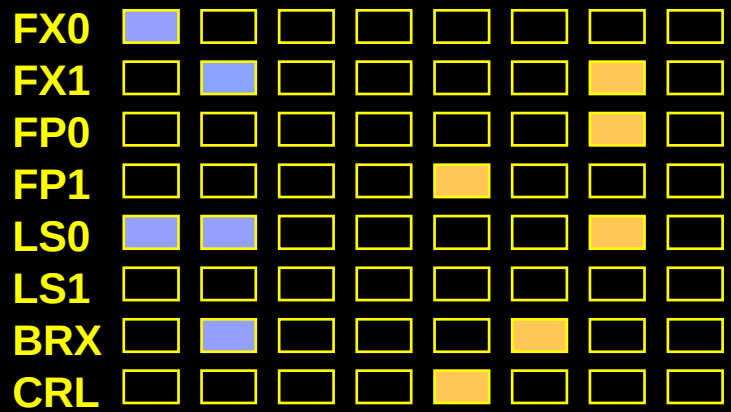


# Multi-threading Evolution

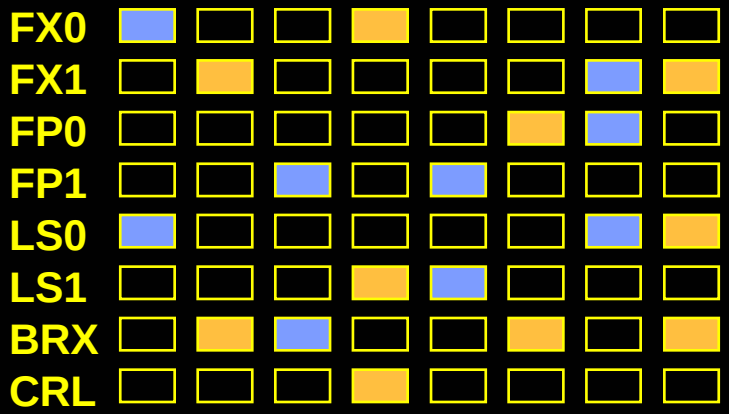
## Single Thread



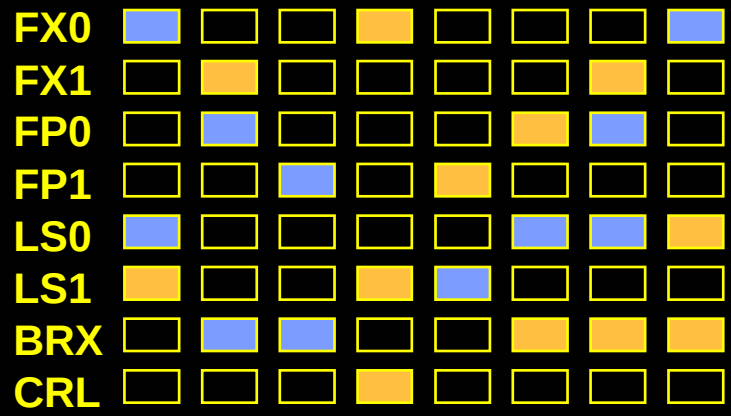

## Coarse Grain Threading




## Fine Grain Threading




## Simultaneous Multi-Threading



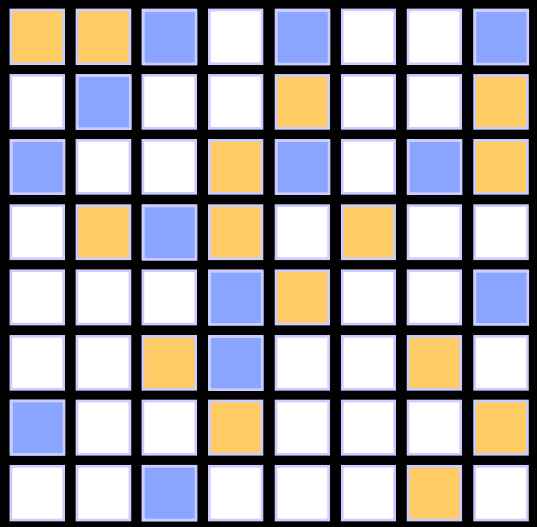

Thread 0 Executing

Thread 1 Executing

No Thread Executing

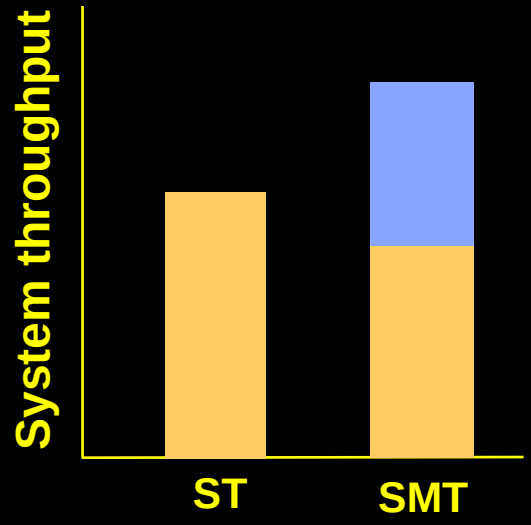
# Simultaneous multi-threading

**POWER5 Simultaneous Multi Threading**



Thread0 active
No thread active
Thread1 active

Appears as 4 CPUs per chip to the operating system (AIX 5L V5.3 and Linux)




- Utilizes unused execution unit cycles
- Symmetric multiprocessing (SMP) programming model
- Natural fit with superscalar out-of-order execution core
- Dispatch two threads per processor. Net result:
  - Better processor utilization

# SMT Experiment

- Build 32 MPI task executable
- Test runs (64 processor p5-595):
  - A: Two tasks on two threads of same processor
    - Use 16 processors with 32 threads
      - 8 Chips
  - B: Two tasks on two processors on same chip
    - Use 32 processors
      - 16 chips
  - C: One task per chip
    - 32 processors
      - 32 chips
  - D: Default operating system scheduling

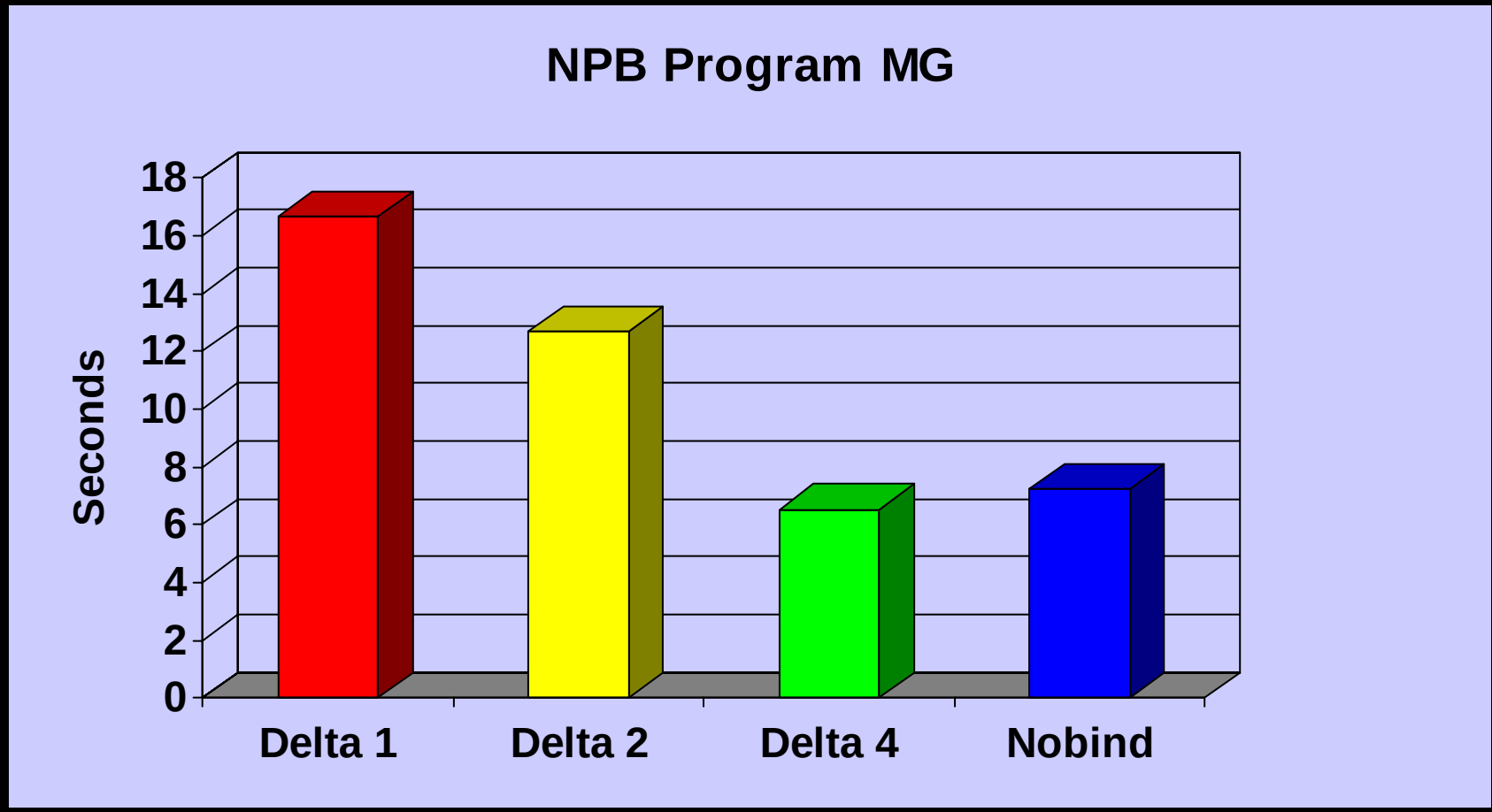
# SMT Experiment

- mpxlf –o a.out –l MPIbind
- export CPU0=0 Delta={1,2,4}
  - Binding library libMPIbind.a
    - Binds task (MPI) and or threads (OpenMP) the processors
      - CPU): Define starting processor number
      - Delta: Processor number increment
- poe a.out –procs 32
  - A: Delta=1: Two tasks on same processor
  - B: Delta=2: Two tasks on same chip
  - C: Delta=4: Separate chip for each processor
  - D: Nobinding: Default; Operating System control

# SMT Experiment

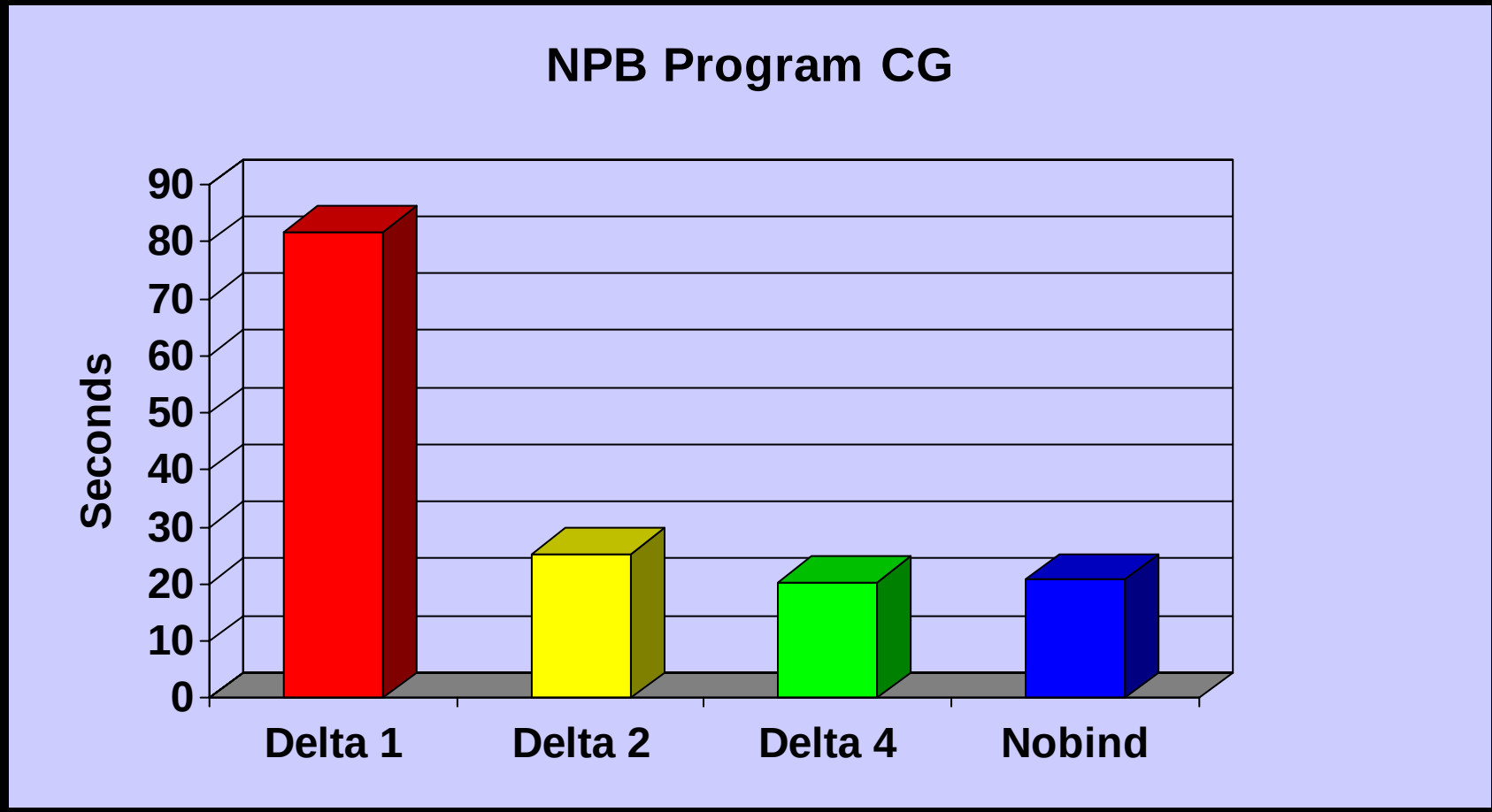
| Test | Threads per processor | Processors | Processor Chips |
|---|---|---|---|
| A | 2 | 16 | 8 |
| B | 1 | 32 | 16 |
| C | 1 | 32 | 32 |
| D | 1 | 32 | 32 |

# SMT Experiment



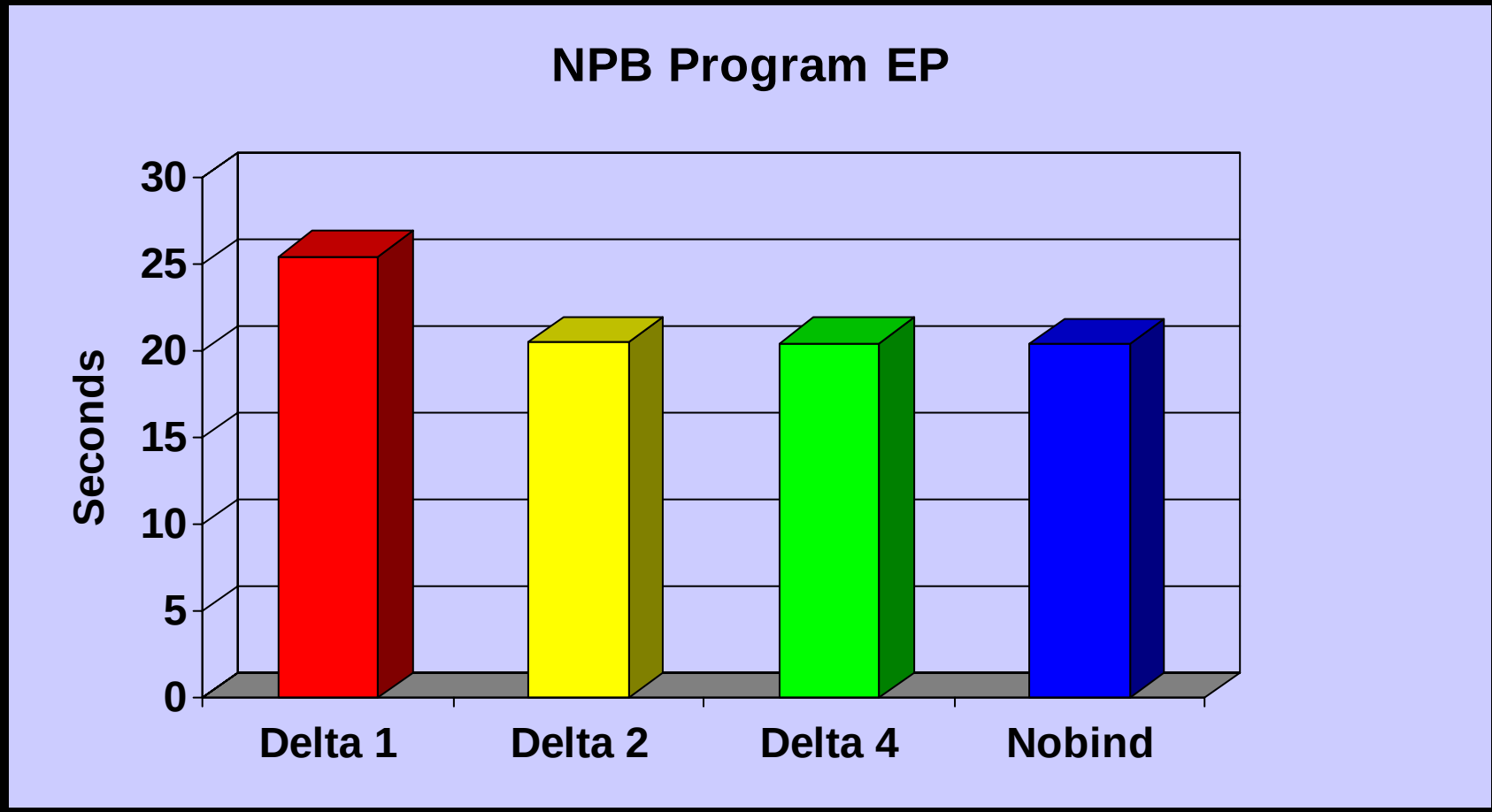

p5-595 1.9 GHz

# SMT Experiment



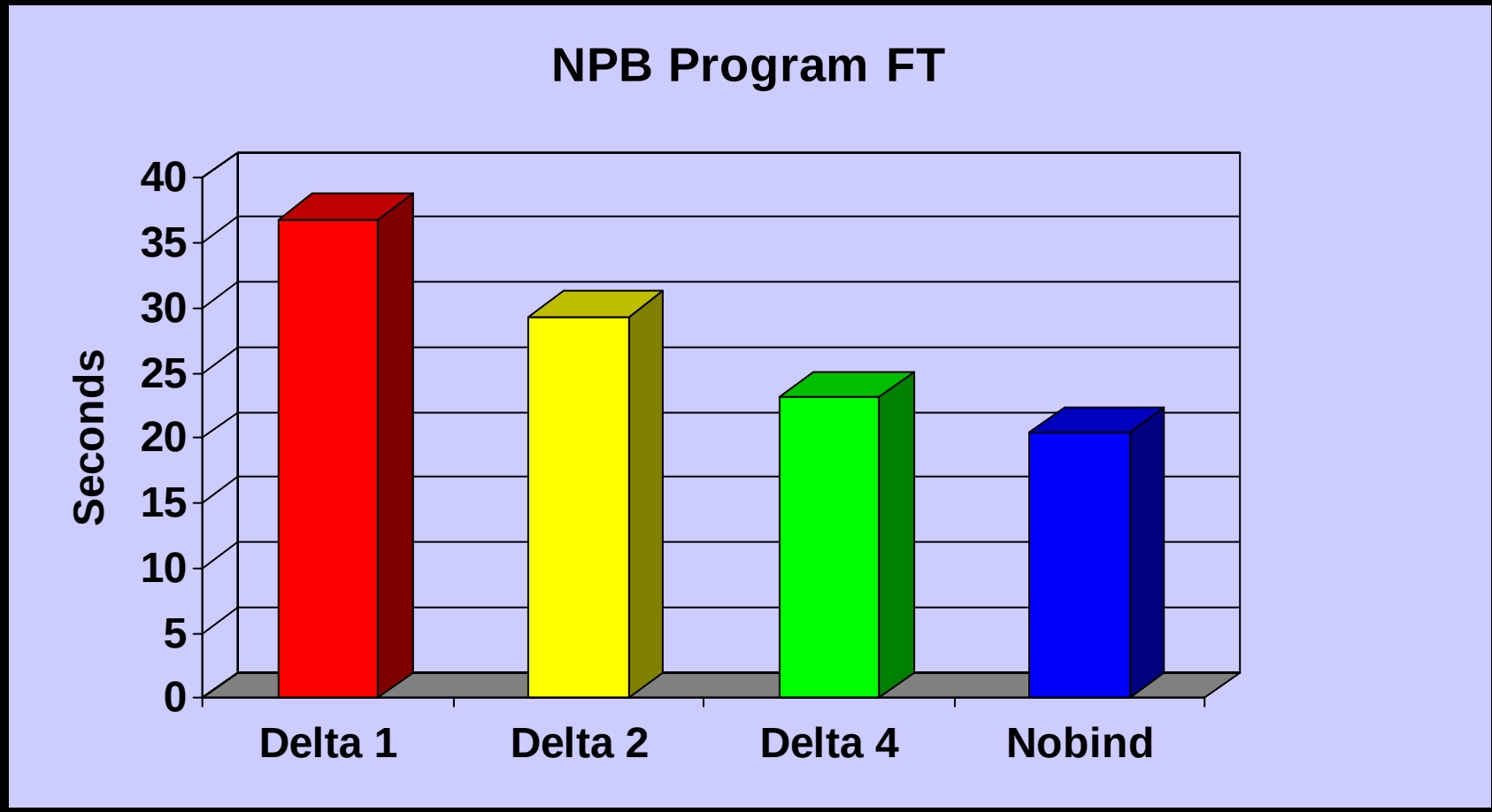

p5-595 1.9 GHz

# SMT Experiment



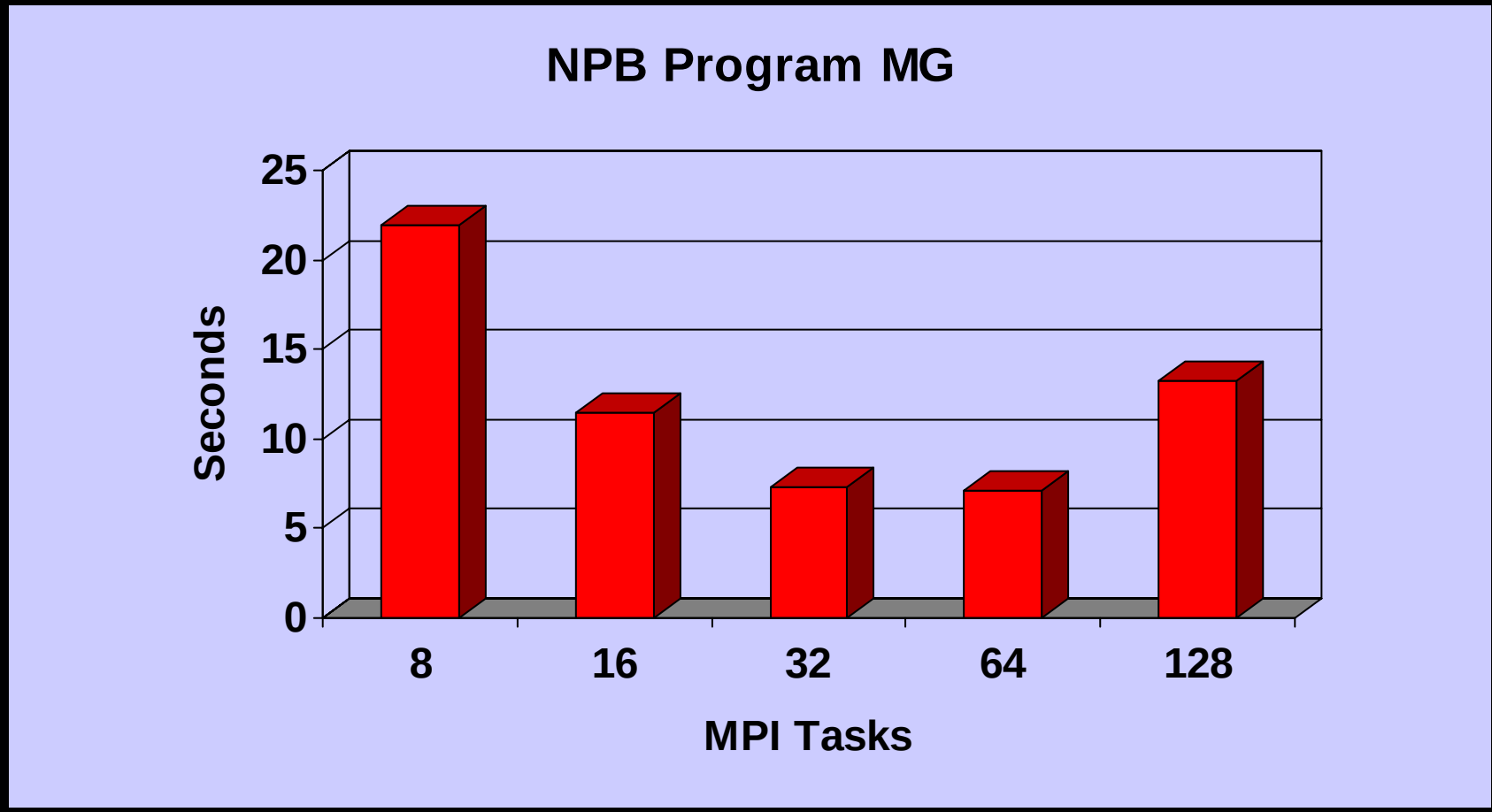

p5-595 1.9 GHz

# SMT Experiment



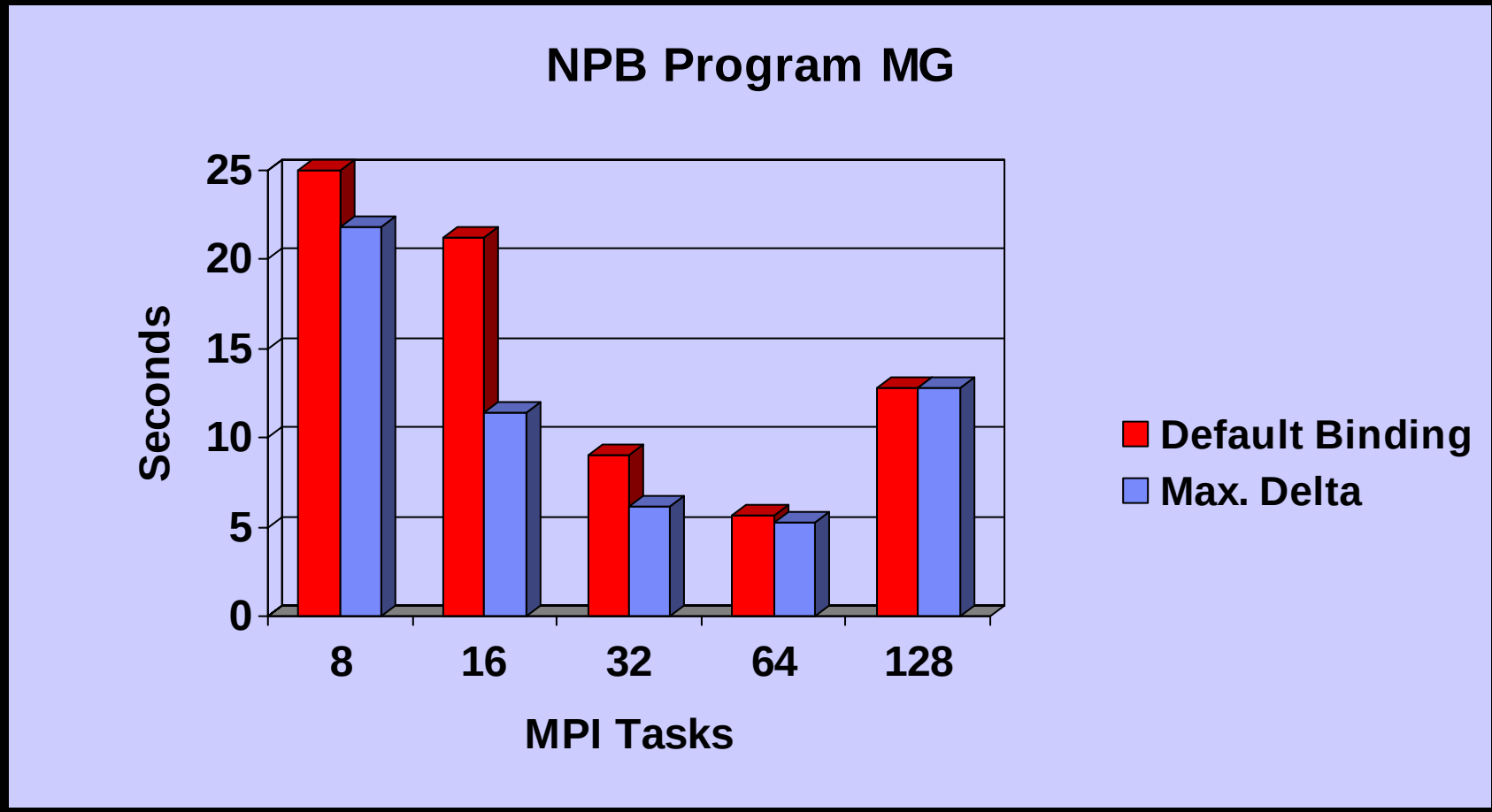

p5-595 1.9 GHz

# Effect of SMT on Tasks



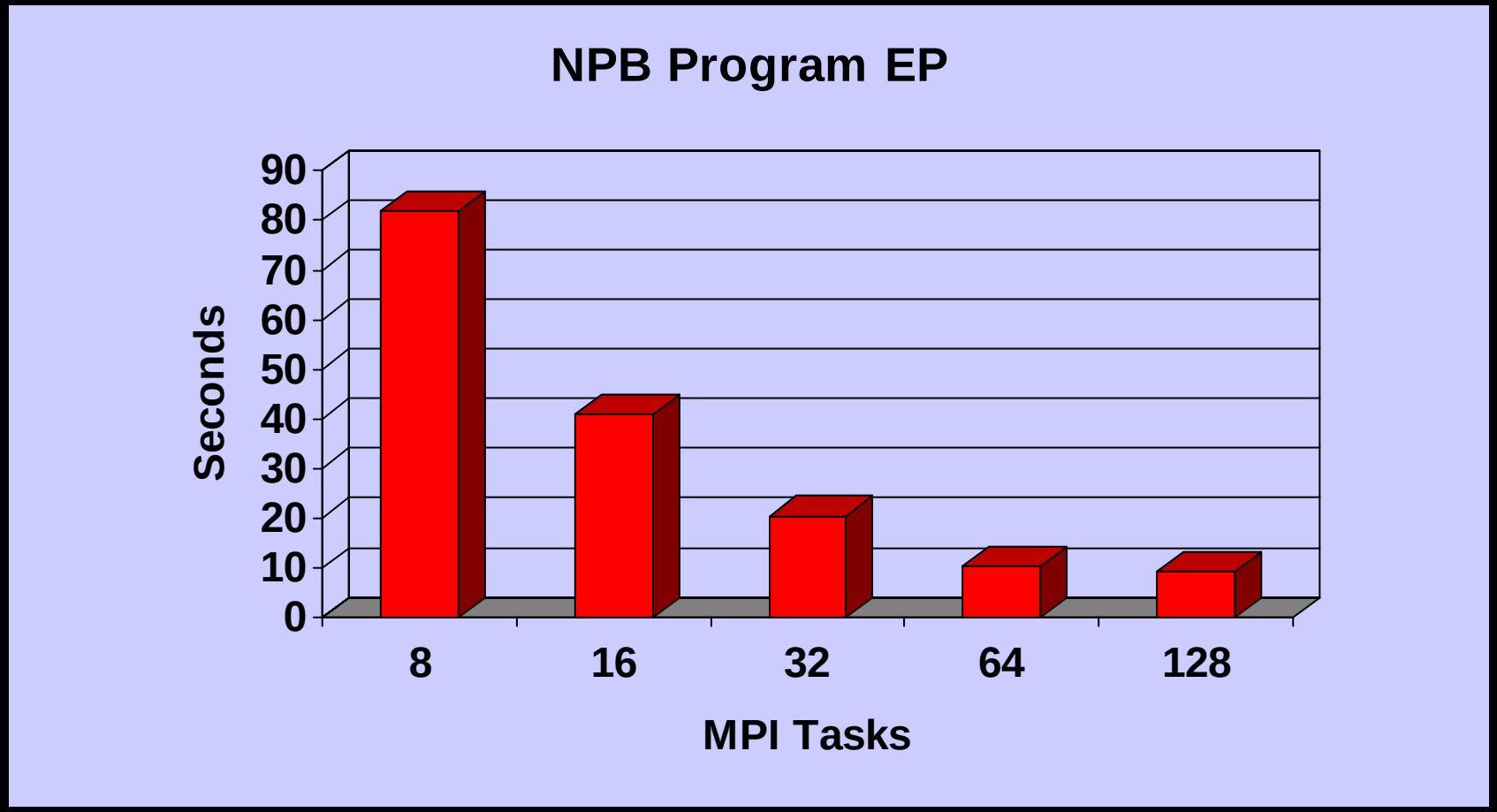

p5-595 1.9 GHz

# Effect of SMT on Tasks
# MG with binding (Delta default)



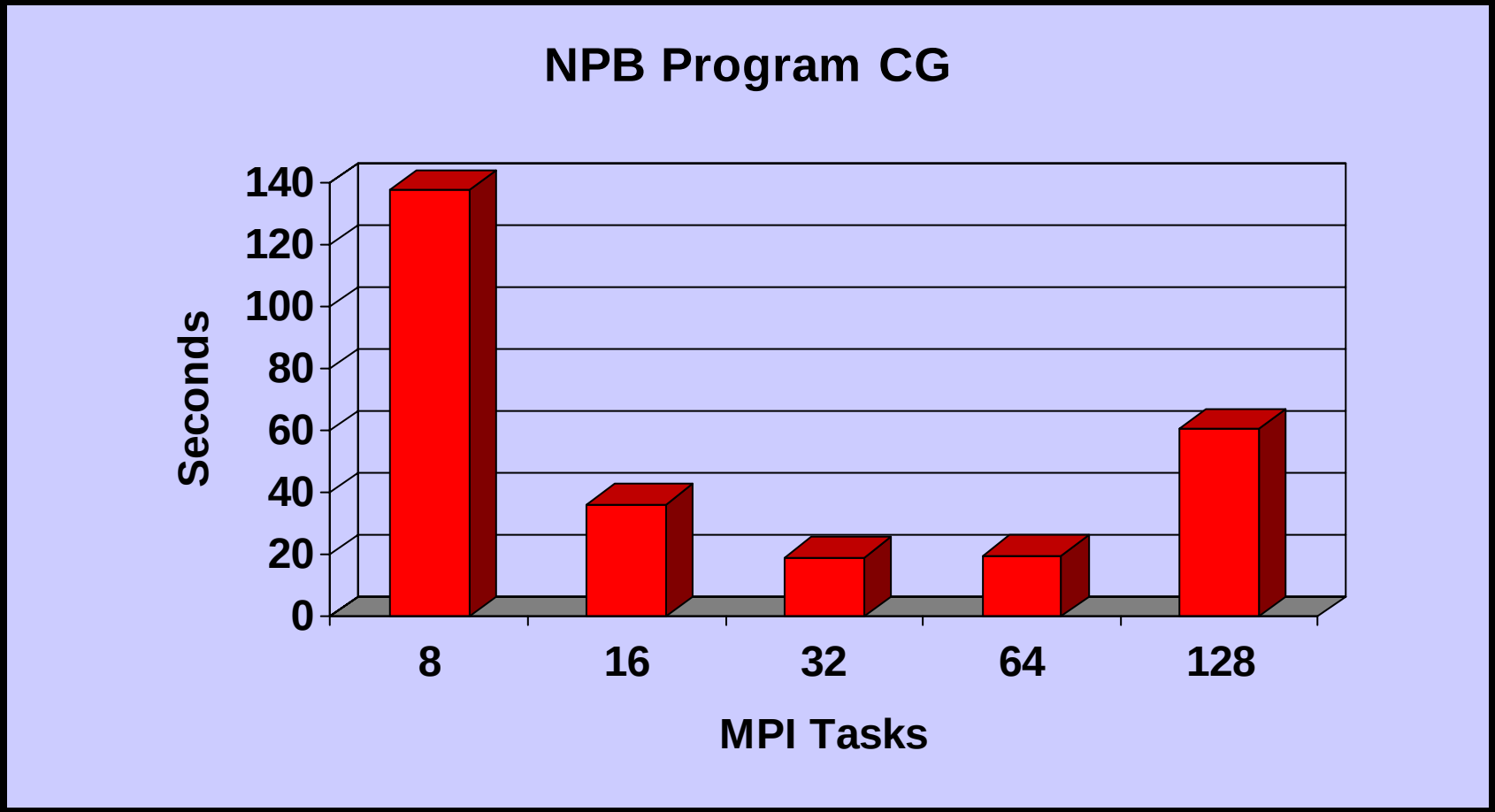

p5-595 1.9 GHz

# Effect of SMT on Tasks




p5-595 1.9 GHz

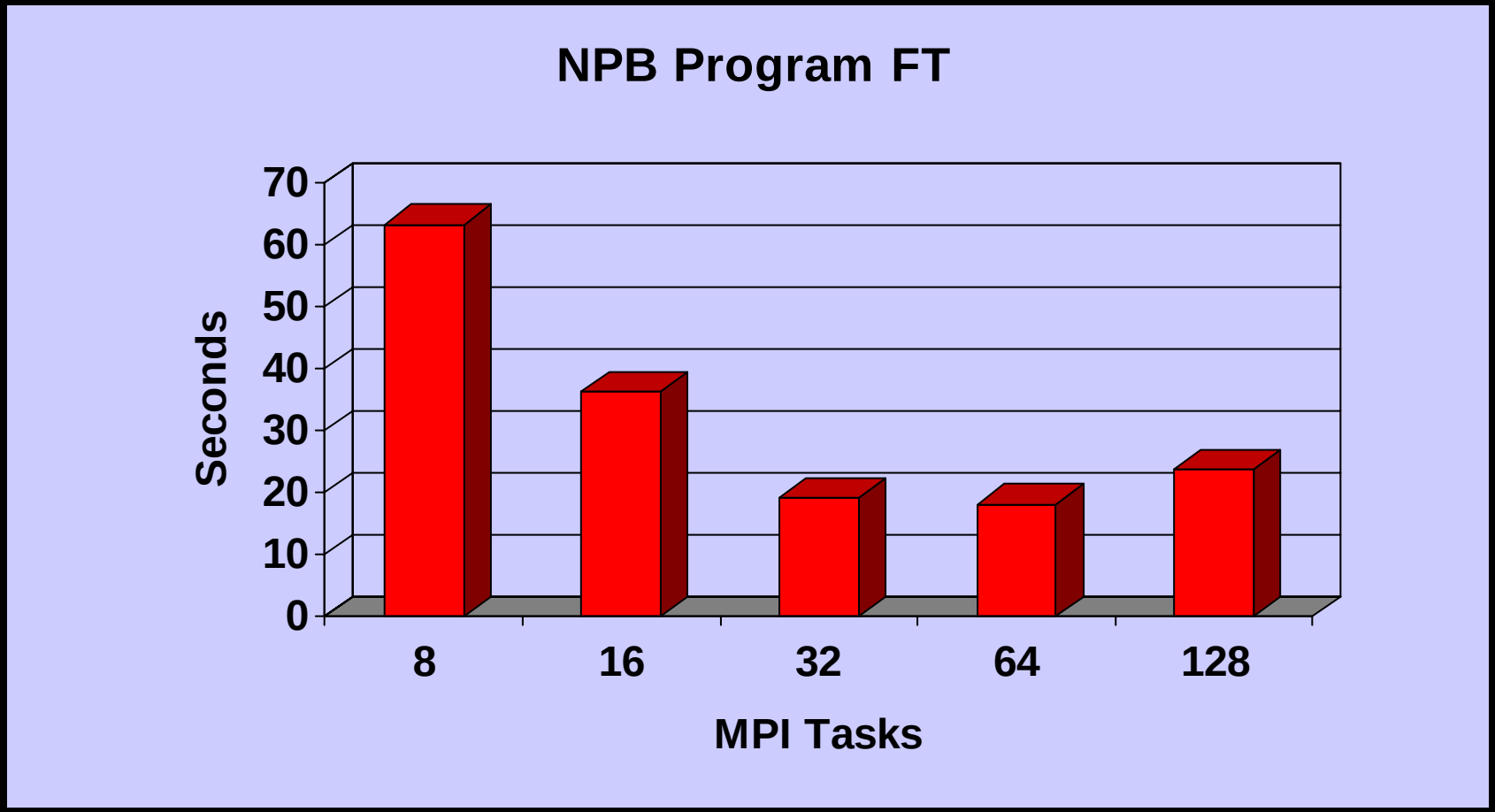
# Effect of SMT on Tasks




p5-595 1.9 GHz

Increase in number of tasks leads to increased communication time

# Effect of SMT on Tasks



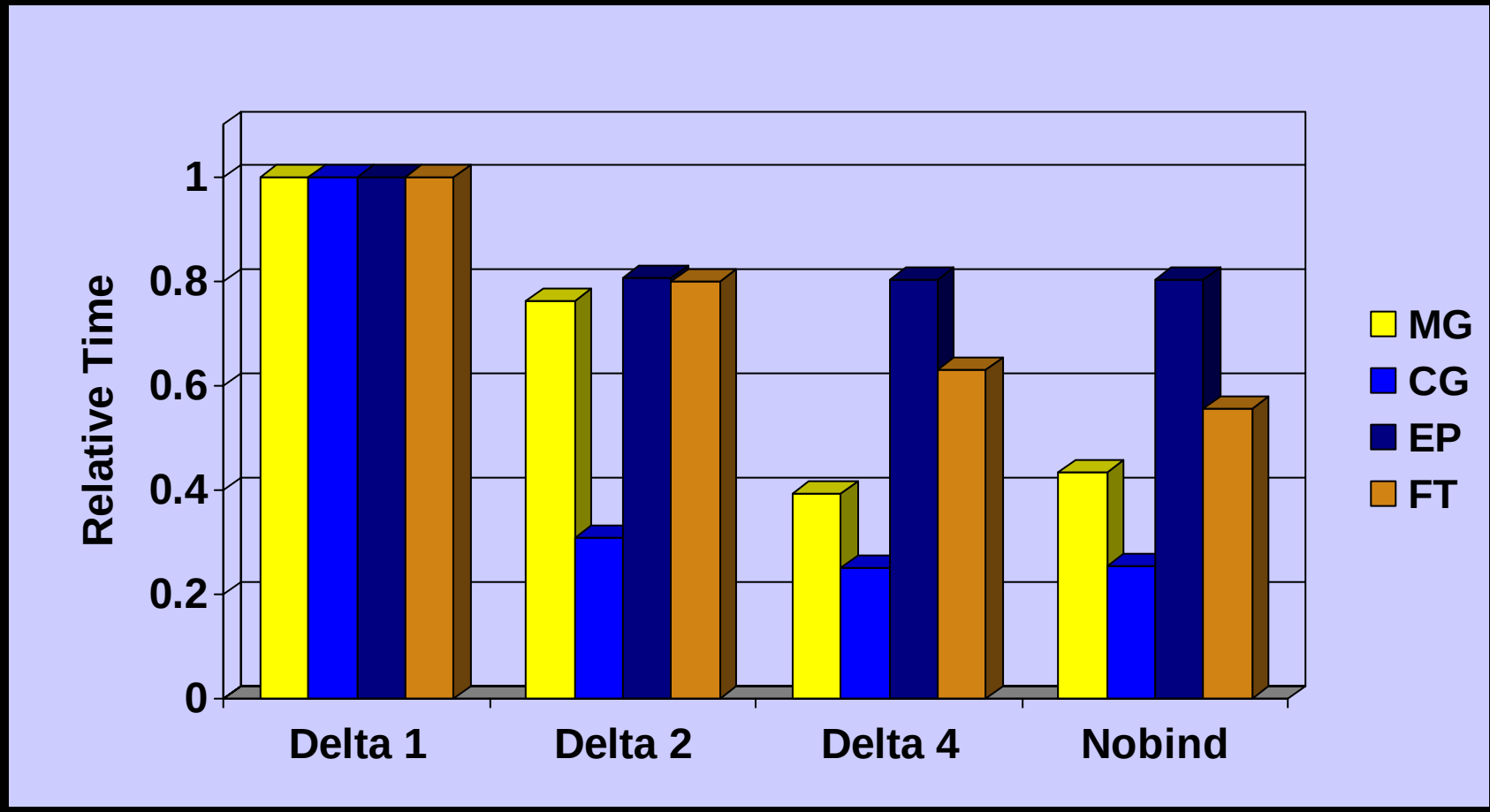

p5-595 1.9 GHz

Increase in number of tasks leads to increased communication time

# Summary

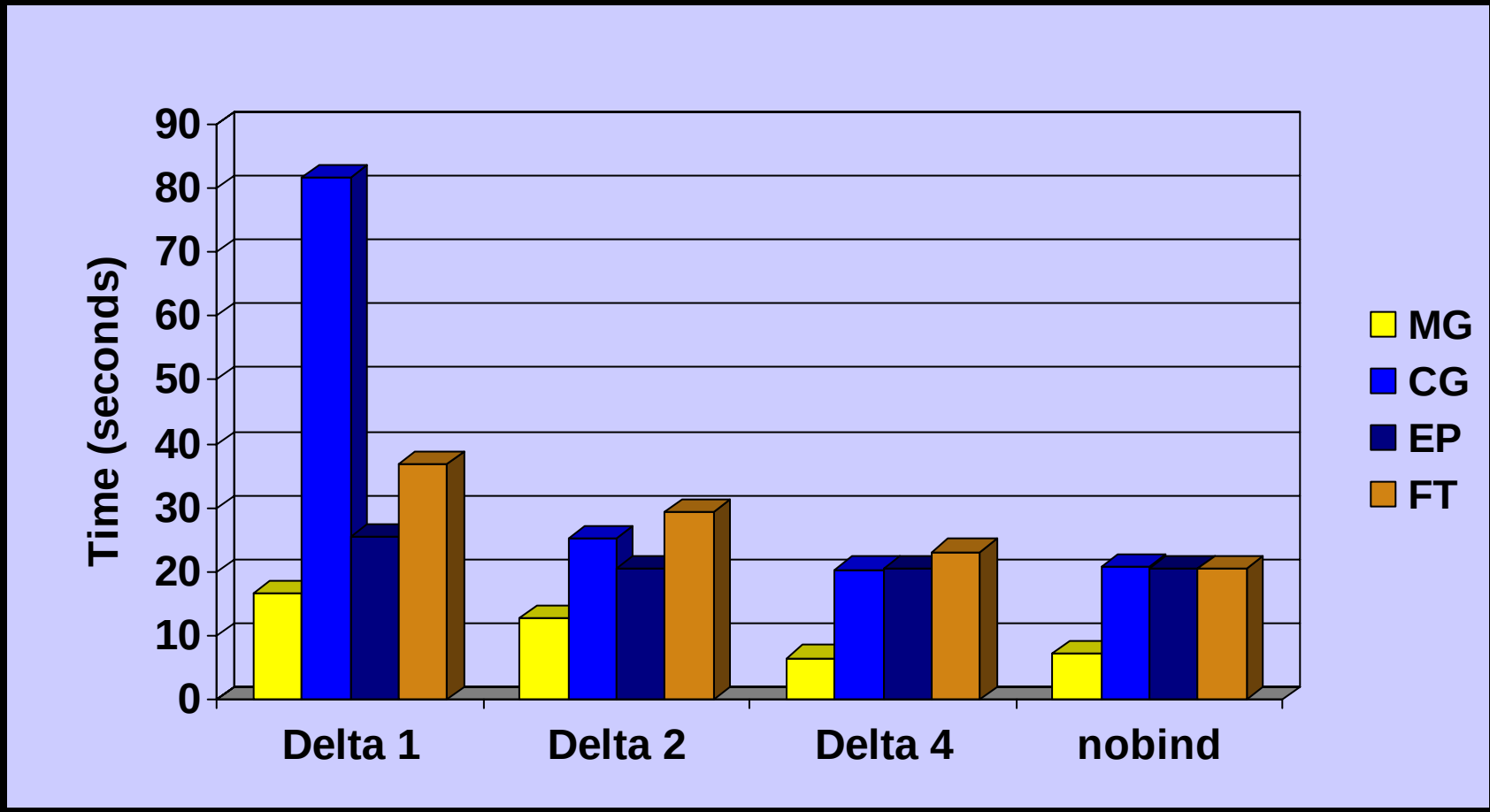
- Large Pages
  - Useful for bandwidth and “gather/scatter”
- Memory affinity:
  - Useful for MPI
  - Not useful for OpenMP
- Process binding
  - Useful for experiments
  - Dangerous for production
- SMT
  - ….?

# SMT Experiment




**p5-595 1.9 GHz**
**NPB MPI Programs**

# SMT Experiment




**p5-595 1.9 GHz**
**NPB MPI Programs**