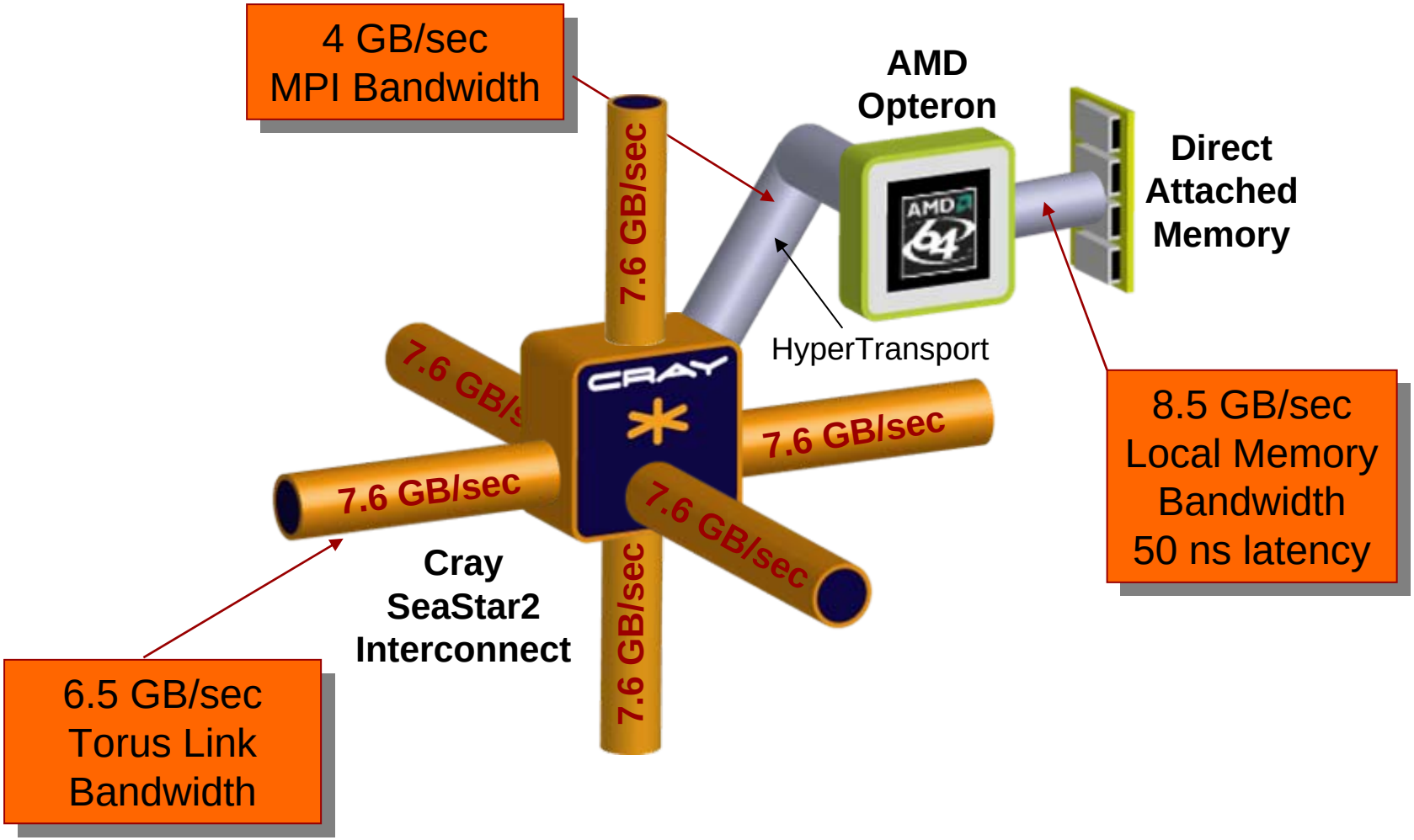
# Optimization for the Cray XT4™ MPP Supercomputer

**John M. Levesque**

**March, 2007**

# The Cray XT4 Processing Element: Providing a bandwidth-rich environment

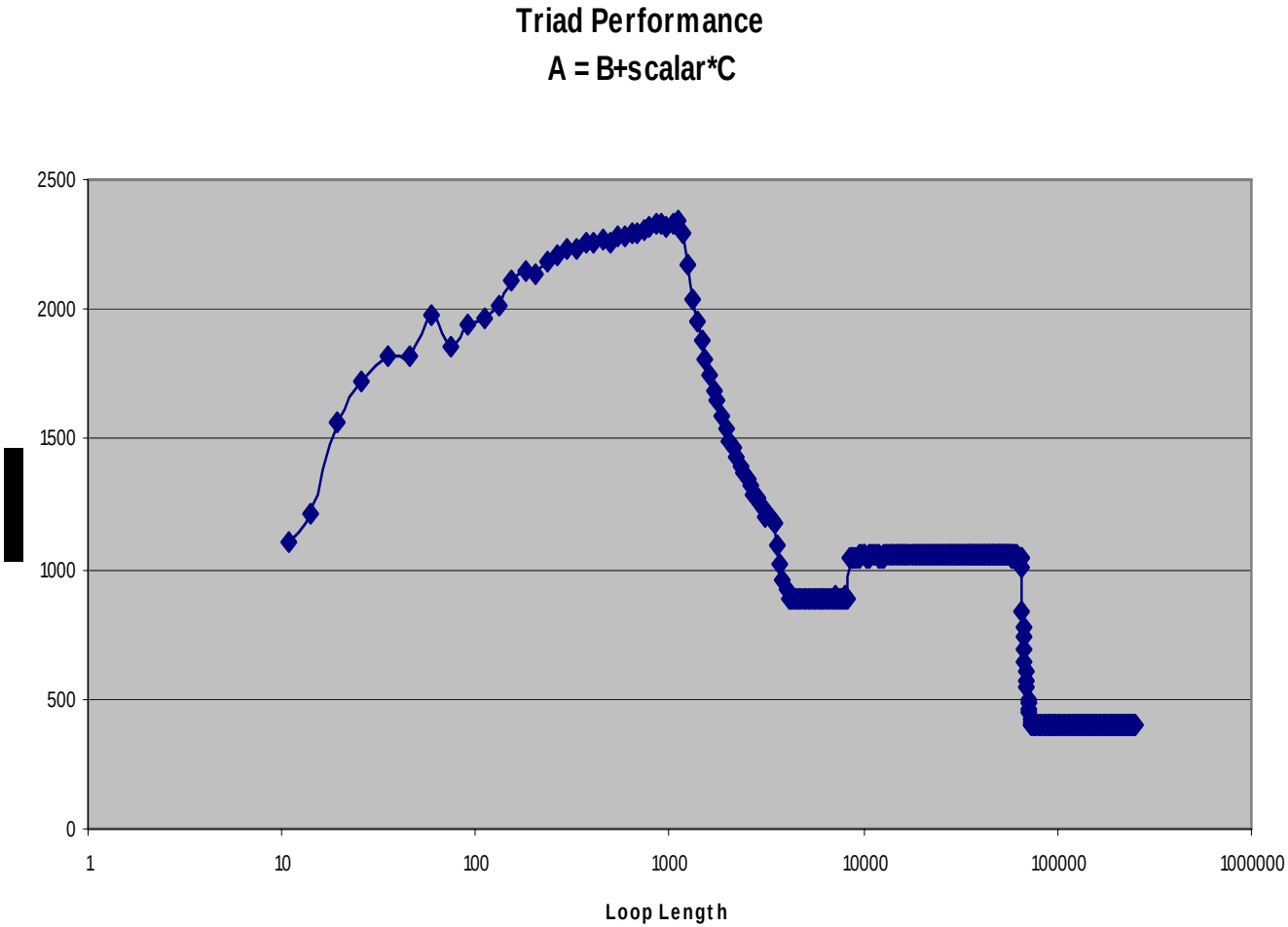


# Opteron Speeds and Feeds

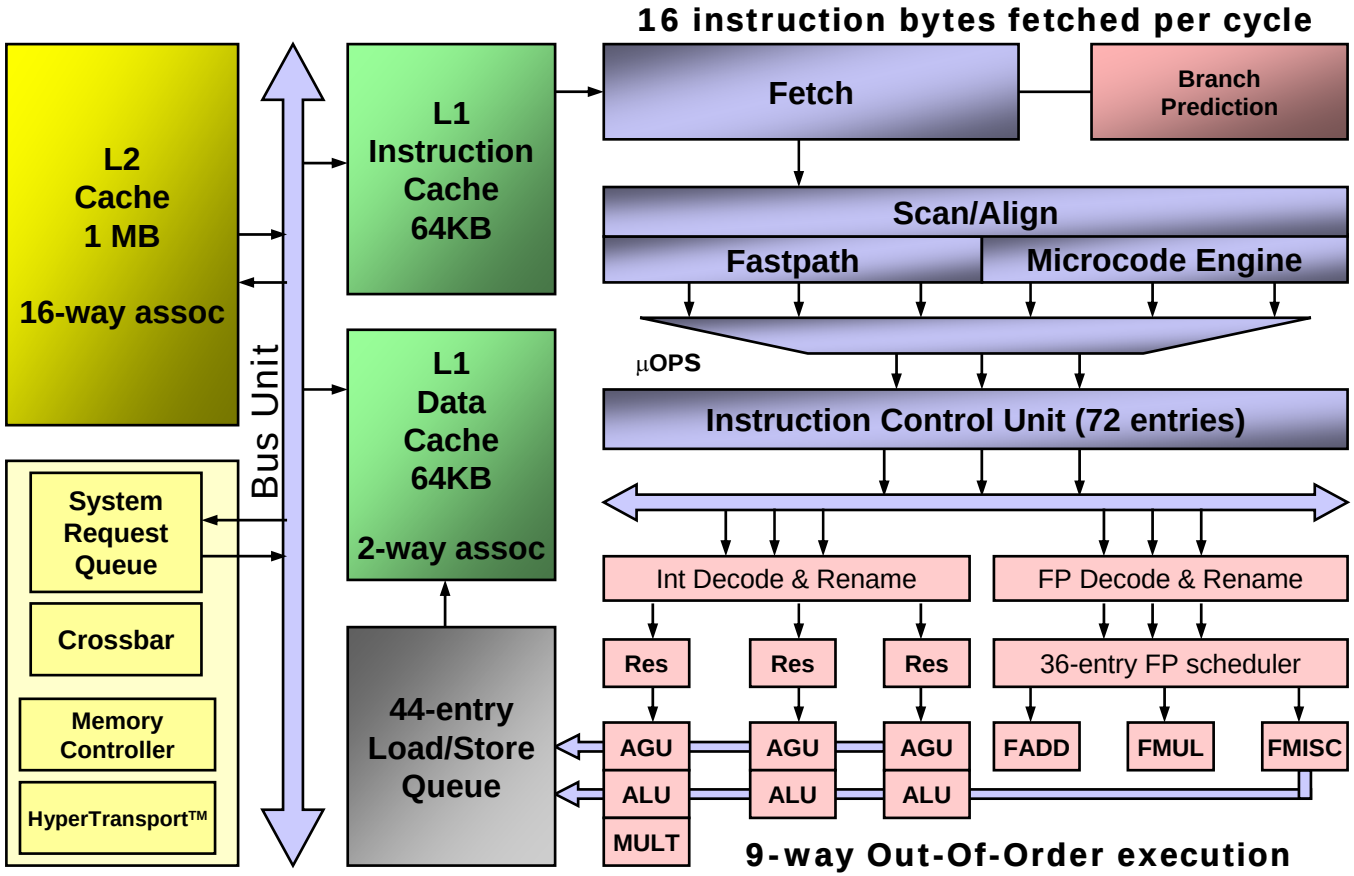
- Core
  - 2.6Ghz clock frequency
  - SSE SIMD FPU (2flops/cycle = 5.2GF peak)
- Cache Hierarchy
  - L1 Dcache/Icache: 64k/core
  - L2 D/I cache: 1M/core
  - 12 HW stream prefetch
  - SW Prefetch and loads to L1
  - Evictions and HW prefetch to L2
- Memory
  - Dual Channel DDR2
  - 10GB/s peak @ 667MHz
  - 8GB/s nominal STREAMs

- **TLB**
  - **Small pages**
    - **4k pages**
    - **512 entries**
    - **covers 2M memory.**
  - **Large pages:**
    - **2MB pages**
    - **8 entries**
    - **covers 16MB memory**
    - **2-pages used by OS (so really only 6 entries covering 12MB)**
- **Shared Resources**
  - **HyperTransport (to Seastar)**
  - **Memory controller**
  - **Otherwise, no other shared resources!!!**

# Performance = F( Cache Utilization )

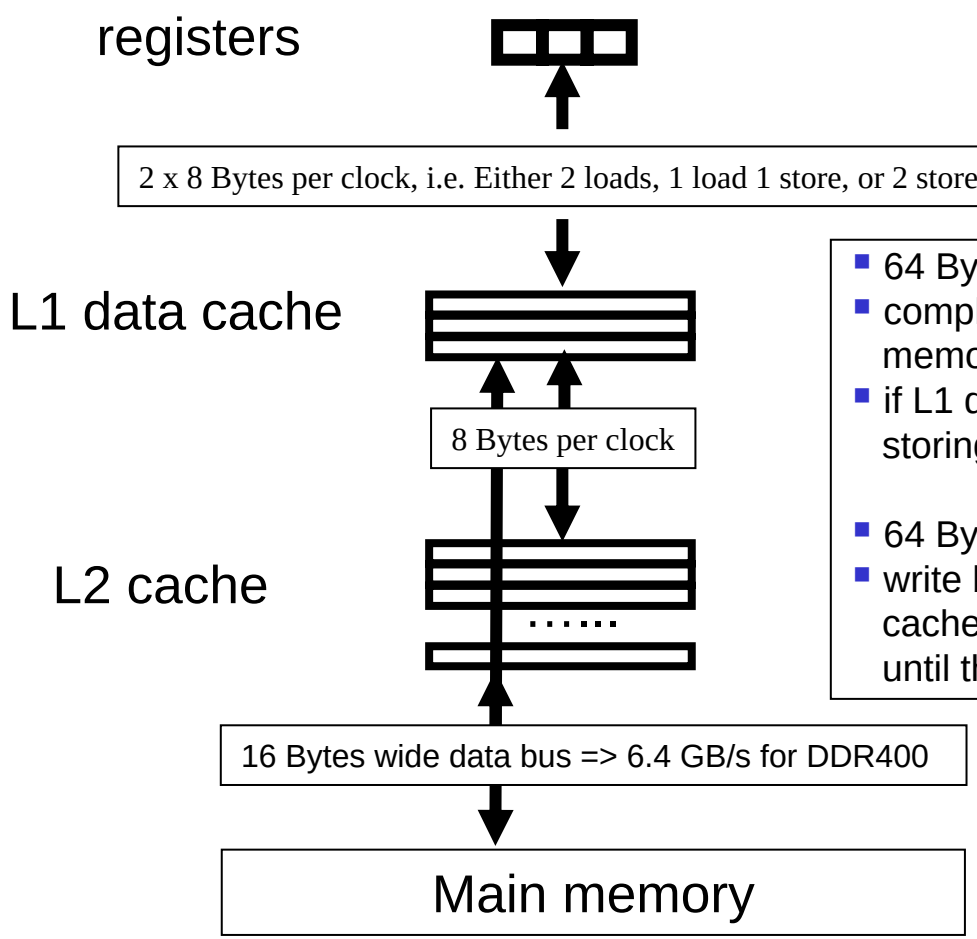


# AMD Opteron Processor




- 36 entry FPU instruction scheduler
- 64-bit/80-bit FP Realized throughput (1 Mul + 1 Add)/cycle: 1.9 FLOPs/cycle
- 32-bit FP Realized throughput (2 Mul + 2 Add)/cycle: 3.4+ FLOPs/cycle

# Simplified memory hierachy on the AMD Opteron

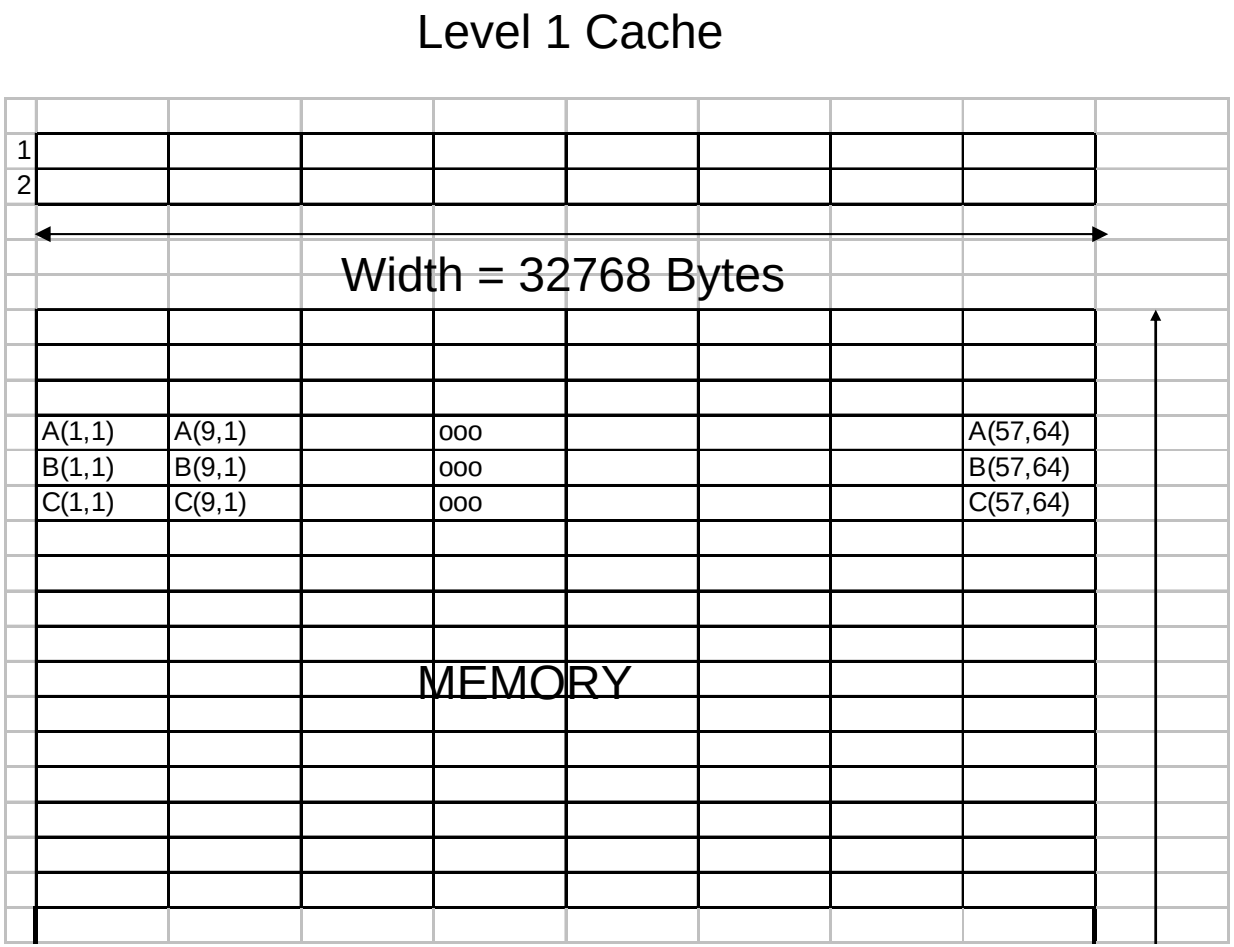



- 64 Byte cache line
- complete data cache lines are loaded from main memory, if not in L2 cache
- if L1 data cache needs to be refilled, then storing back to L2 cache

- 64 Byte cache line
- write back cache: data offloaded from L1 data cache are stored here first until they are flushed out to main memory

```
Real * 8  A(64,64),B(64,64),C(64,64)

DO I = 1,N
      C(I,1) = A(I,1) +B(I,1)
ENDDO
```

# Cache Visualization



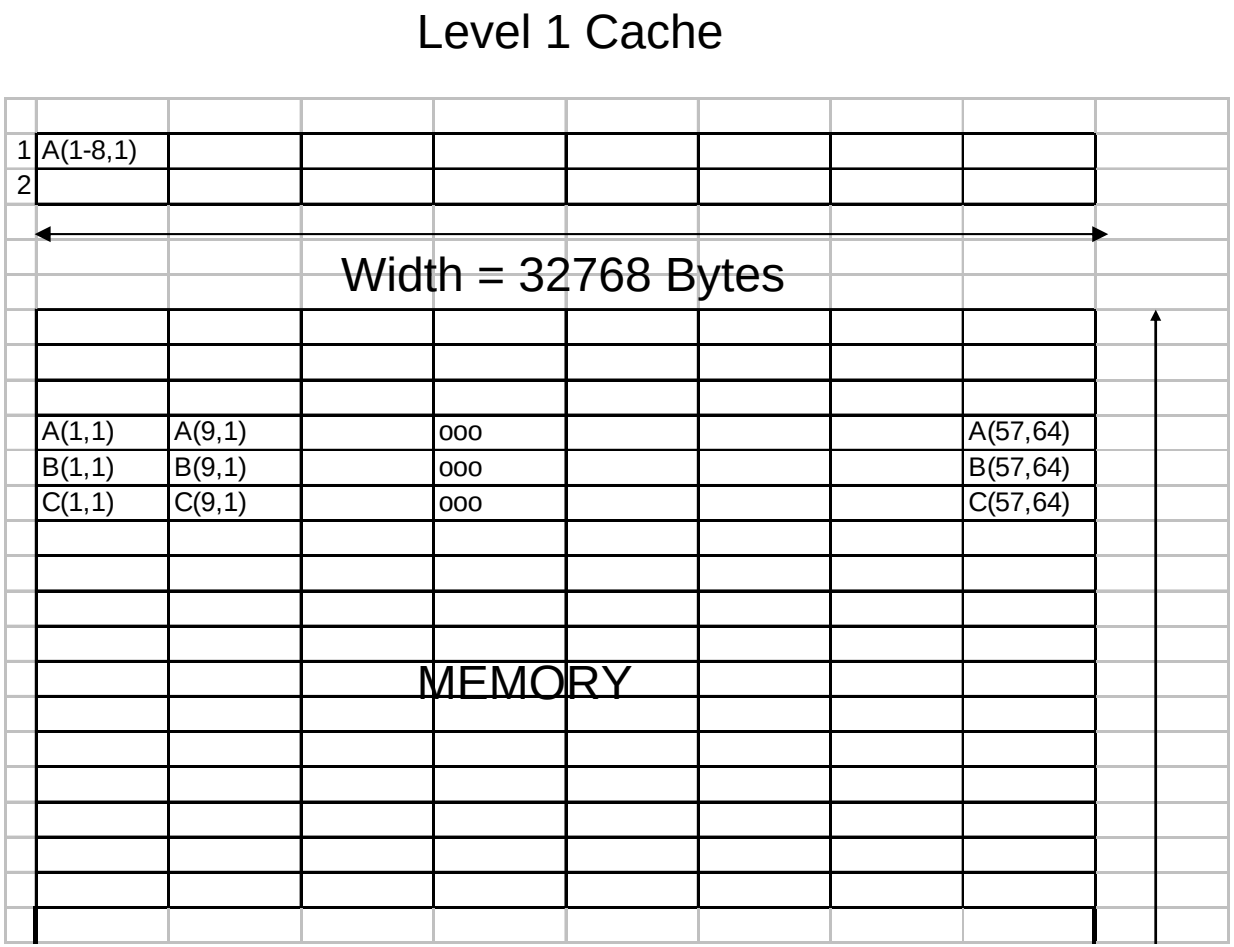

Level 1 Cache
- 65536 B
- 1024 Lines
- 8192 8B Ws
- 16384 4B Ws
- 2 way Assoc

Associativity Class
- 32768 B
- 512 Lines
- 4096 8B Ws
- 8192 4B Ws

64*64*8 = 32768 B

# Consider the following example

| | | | | | |
|---|---|---|---|---|---|
| Real * 8 | A(64,64),B(64,64),C(64,64) | | | | |
| | | | | | |
| | | | | | |
| DO I = 1,N | | | | | |
| C(I,1) = A(I,1) +B(I,1) | | | | | |
| ENDDO | | | | | |
| | | | | | |
| Fetch A(1,1) | | Fetch from M | Uses 1 Associativity Class | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |




Level 1 Cache

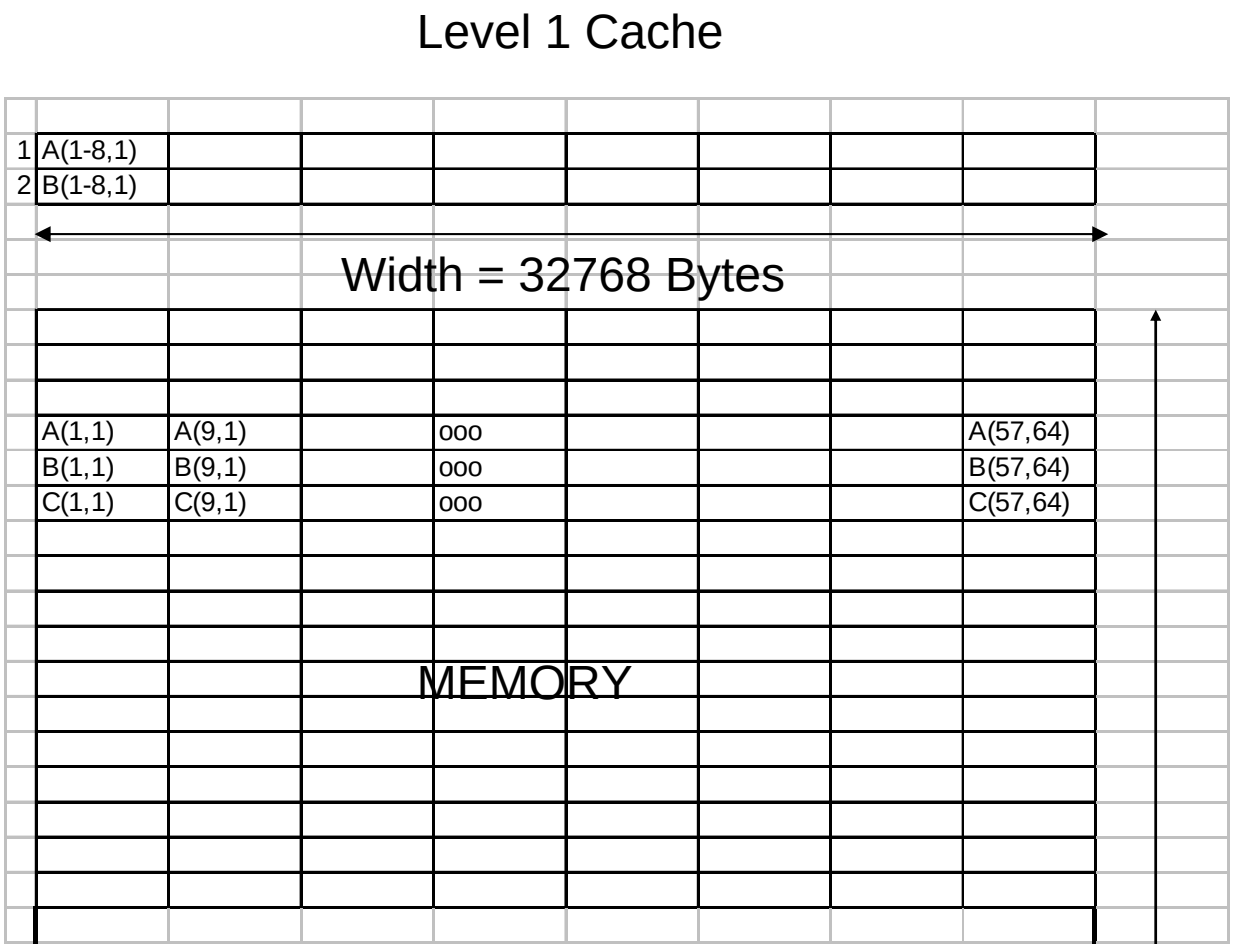
- 65536 B
- 1024 Lines
- 8192 8B Ws
- 16384 4B Ws
- 2 way Assoc

Associativity Class

- 32768 B
- 512 Lines
- 4096 8B Ws
- 8192 4B Ws

64*64*8 = 32768 B

| | | | | | |
|---|---|---|---|---|---|
| Real * 8 | A(64,64),B(64,64),C(64,64) | | | | |
| | | | | | |
| | | | | | |
| DO I = 1,N | | | | | |
| C(I,1) = A(I,1) +B(I,1) | | | | | |
| ENDDO | | | | | |
| | | | | | |
| Fetch A(1,1) | | Fetch from M | Uses 1 Associativity Class | | |
| Fetch B(1,1) | | Fetch from M | Uses 2 Associativity Class | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |




Level 1 Cache

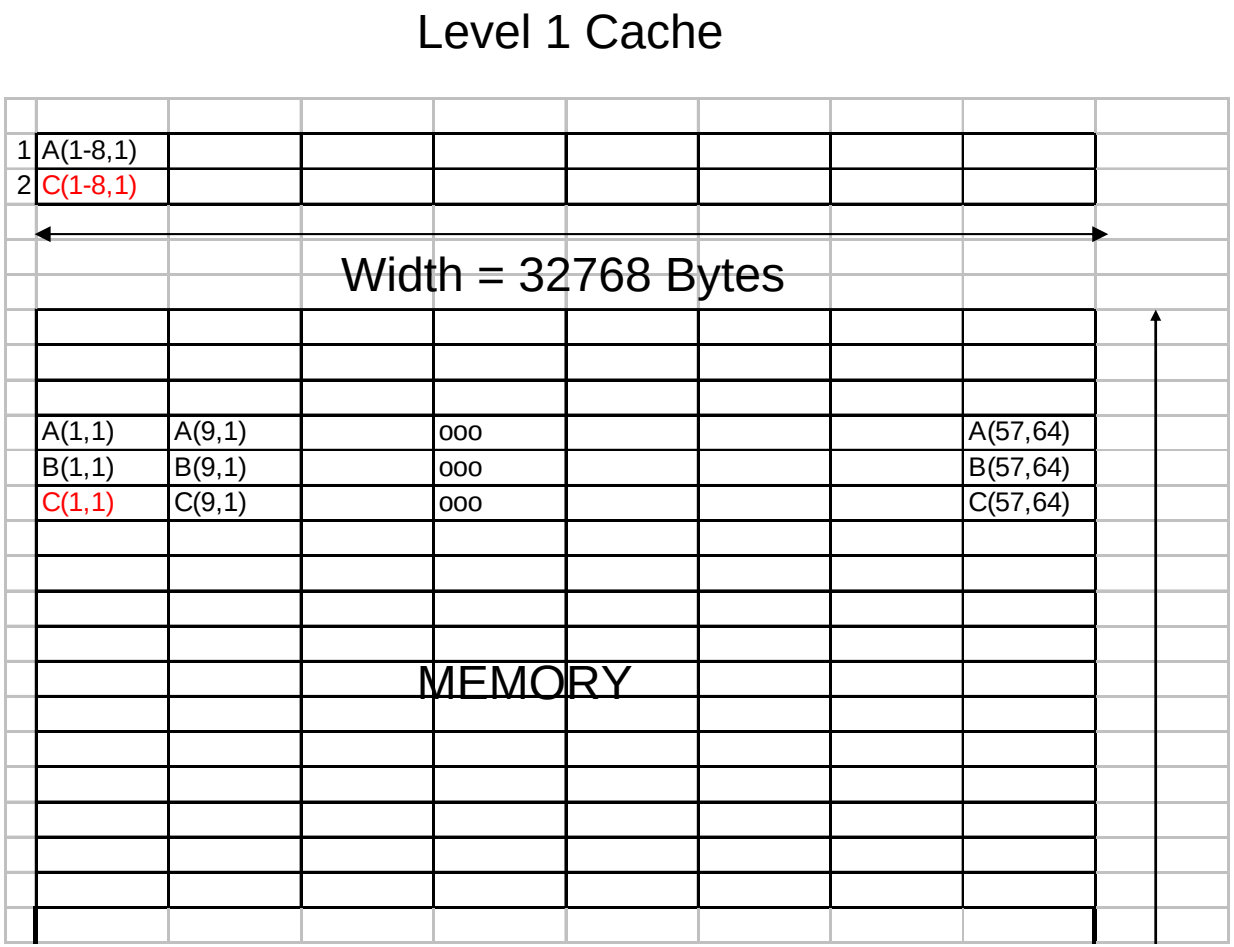
- 65536 B
- 1024 Lines
- 8192 8B Ws
- 16384 4B Ws
- 2 way Assoc

Associativity Class

- 32768 B
- 512 Lines
- 4096 8B Ws
- 8192 4B Ws

64*64*8 = 32768 B

| | | | | | |
|---|---|---|---|---|---|
| Real * 8 | A(64,64),B(64,64),C(64,64) | | | | |
| | | | | | |
| | | | | | |
| DO I = 1,N | | | | | |
| C(I,1) = A(I,1) +B(I,1) | | | | | |
| ENDDO | | | | | |
| | | | | | |
| Fetch A(1,1) | | Fetch from M | Uses 1 Associativity Class | | |
| Fetch B(1,1) | | Fetch from M | Uses 2 Associativity Class | | |
| Add A(1,1) + B(1,1) | | | | | |
| Store C(1,1) | | Fetch from M | Overwrites either 1 or 2 Associativity Class | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |




Level 1 Cache

- 65536 B
- 1024 Lines
- 8192 8B Ws
- 16384 4B Ws
- 2 way Assoc

Associativity Class

- 32768 B
- 512 Lines
- 4096 8B Ws
- 8192 4B Ws

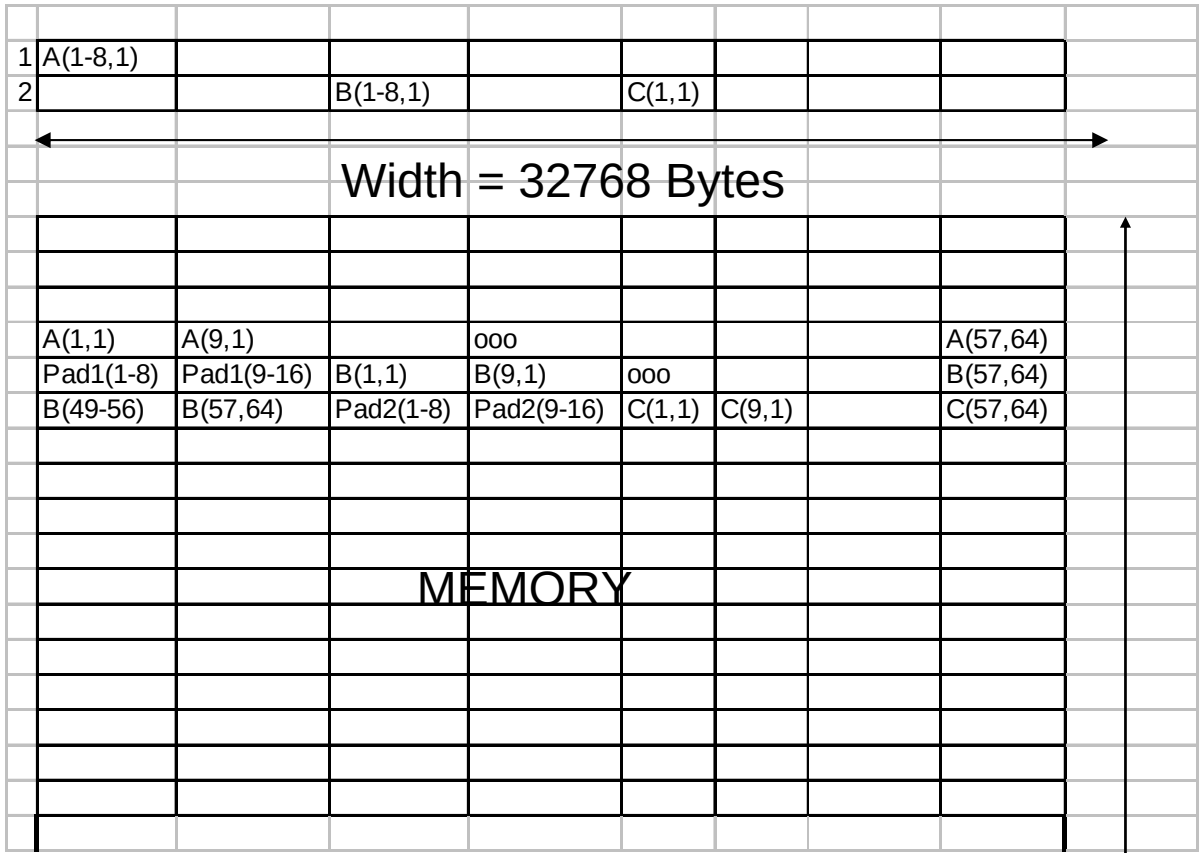
64*64*8 = 32768 B

| | | | | | |
|---|---|---|---|---|---|
| Real * 8 | A(64,64),B(64,64),C(64,64) | | | | |
| | | | | | |
| | | | | | |
| DO I = 1,N | | | | | |
| C(I,1) = A(I,1) +B(I,1) | | | | | |
| ENDDO | | | | | |
| | | | | | |
| Fetch A(1,1) | | Fetch from M | Uses 1 Associativity Class | | |
| Fetch B(1,1) | | Fetch from M | Uses 2 Associativity Class | | |
| Add A(1,1) + B(1,1) | | | | | |
| Store C(1,1) | | Fetch from M | Overwrites either 1 or 2 Associativity Class | | |
| | | | | | |
| Fetch A(2,1) | | Fetch from L2 | Overwrites either 1 or 2 Associativity Class | | |
| Fetch B(2,1) | | Fetch from L2 | Overwrites either 1 or 2 Associativity Class | | |
| Add A(2,1) + B(2,1) | | | | | |
| Store C(2,1) | | Fetch from L2 | Overwrites either 1 or 2 Associativity Class | | |

# Must be a better Way

| | | | | | |
|---|---|---|---|---|---|
| Real * 8 | A(64,64),pad1(16),B(64,64),pad2(16),C(64,64) | | | | |
| | | | | | |
| | | | | | |
| DO I = 1,N | | | | | |
| C(I,1) = A(I,1) +B(I,1) | | | | | |
| ENDDO | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

## Level 1 Cache

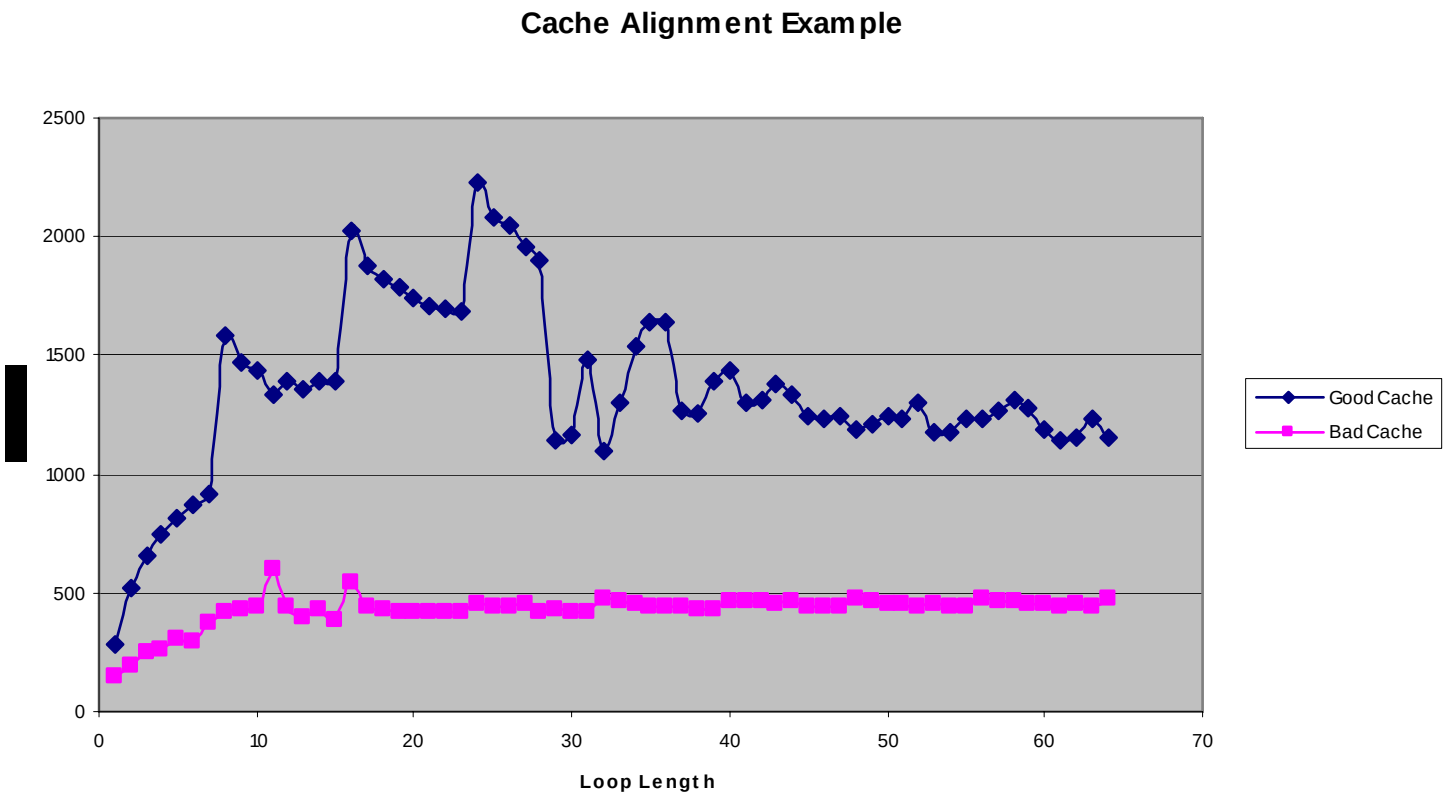



Level 1 Cache
- 65536 B
- 1024 Lines
- 8192 8B Ws
- 16384 4B Ws
- 2 way Assoc

Associativity Class
- 32768 B
- 512 Lines
- 4096 8B Ws
- 8192 4B Ws

64*64*8 = 32768 B

| | | | | | |
|---|---|---|---|---|---|
| Real * 8 | A(64,64),pad1(16),B(64,64),pad2(16),C(64,64) | | | | |
| | | | | | |
| | | | | | |
| DO I = 1,N | | | | | |
| C(I,1) = A(I,1) +B(I,1) | | | | | |
| ENDDO | | | | | |
| | | | | | |
| Fetch A(1) | | Uses 1 Associativity Class | | | |
| Fetch B(1) | | Uses 2 Associativity Class | | | |
| Add A(1) + B(1) | | | | | |
| Store C(1) | | Uses 1 Associativity Class | | | |
| | | | | | |
| Fetch A(2) | | Gets from L1 Cache | | | |
| Fetch B(2) | | Gets from L1 Cache | | | |
| Add A(2) + B(2) | | | | | |
| Store C(2) | | Gets from L1 Cache | | | |


Cache Alignment Example
2500
2000
1500
1000
500
0
0
10
20
30
40
50
60
70
Loop Length
Good Cache
Bad Cache

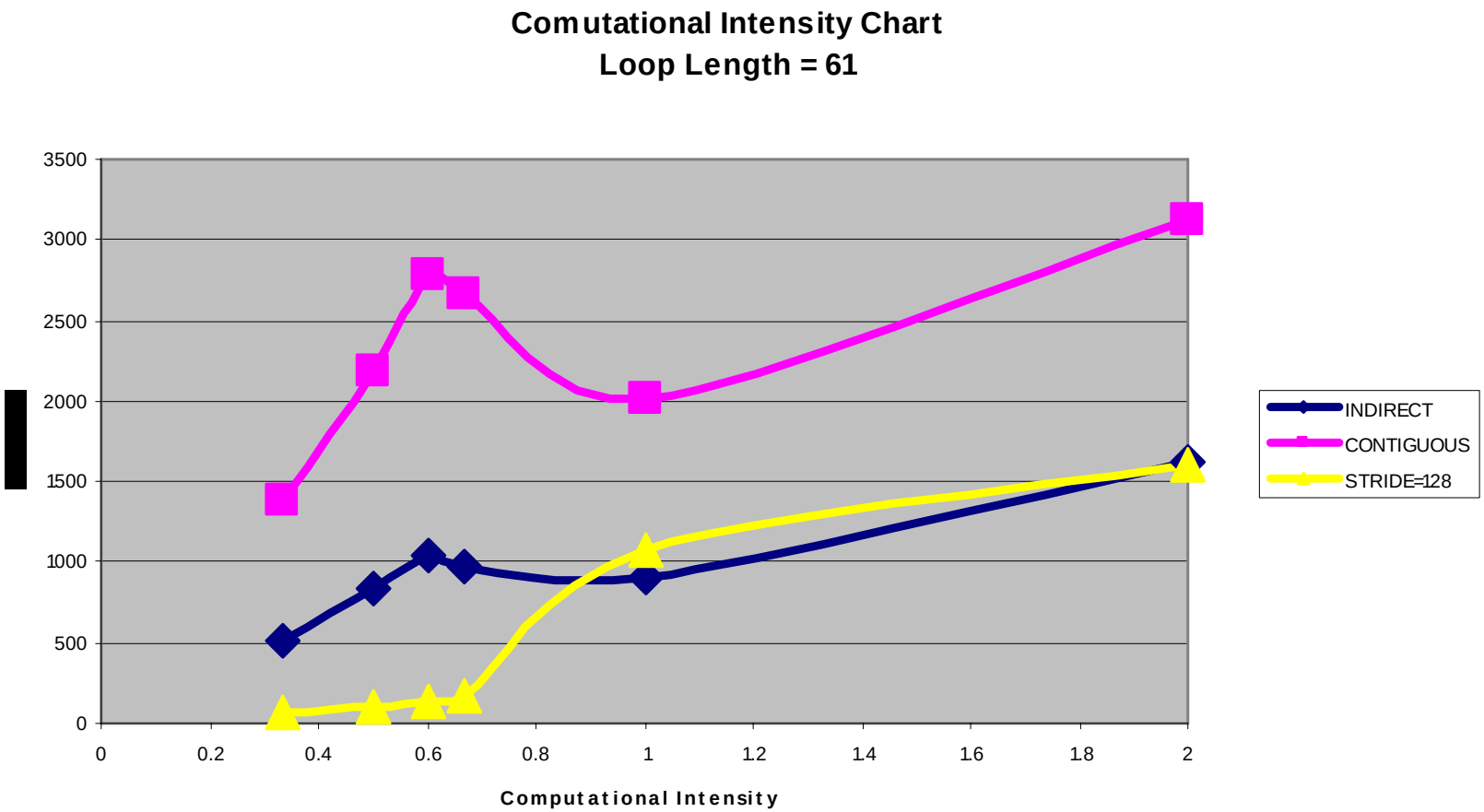
# Bad Cache Alignment

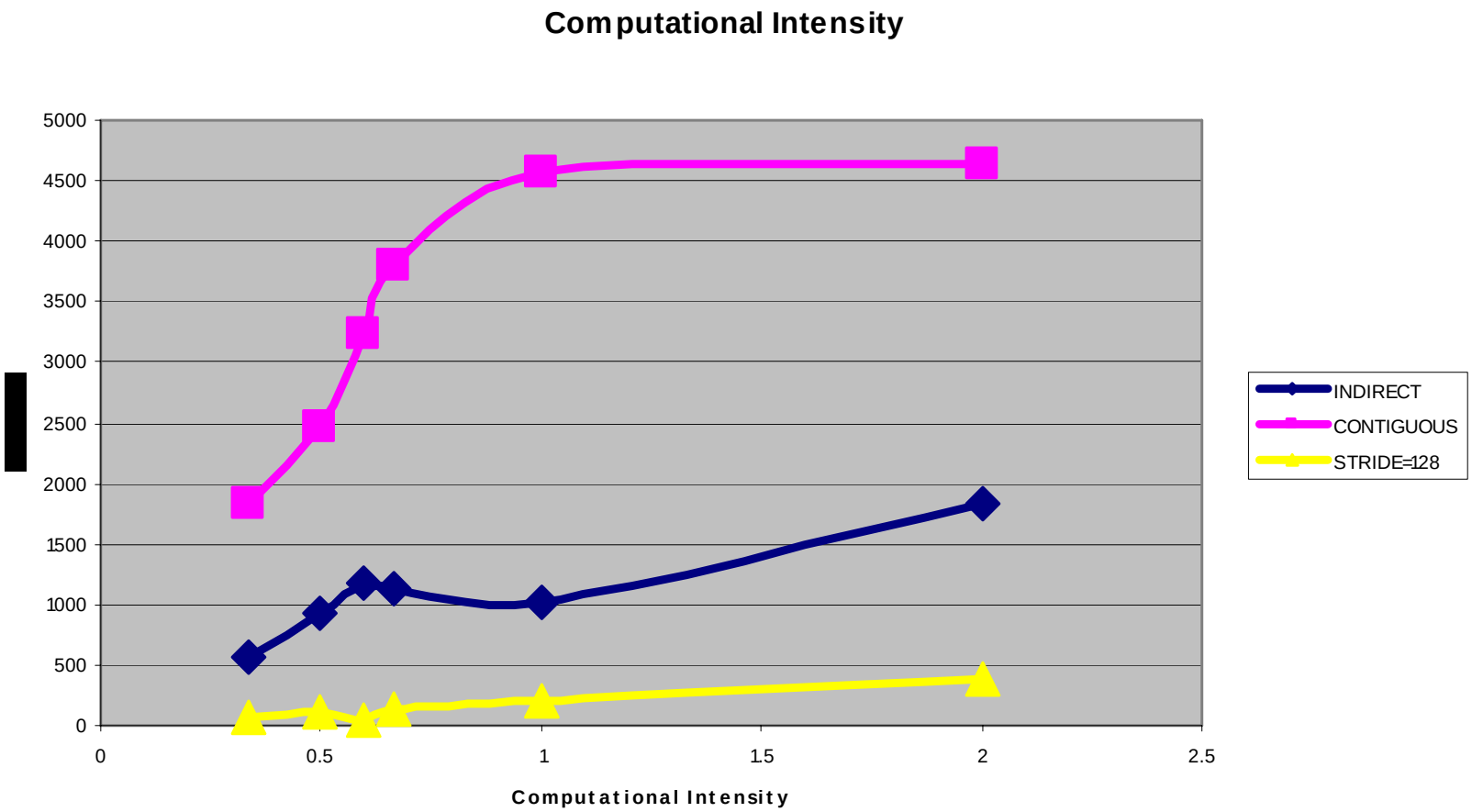
| | | | |
|---|---|---|---|
| Time% | | 0.2% | |
| Time | | 0.000003 | |
| Calls | | 1 | |
| PAPI_L1_DCA | 455.433M/sec | 1367 | ops |
| DC_L2_REFILL_MOESI | 49.641M/sec | 149 | ops |
| DC_SYS_REFILL_MOESI | 0.666M/sec | 2 | ops |
| BU_L2_REQ_DC | 74.628M/sec | 224 | req |
| User time | 0.000 secs | 7804 | cycles |
| Utilization rate | | 97.9% | |
| L1 Data cache misses | 50.308M/sec | 151 | misses |
| LD & ST per D1 miss | | 9.05 | ops/miss |
| D1 cache hit ratio | | 89.0% | |
| LD & ST per D2 miss | | 683.50 | ops/miss |
| D2 cache hit ratio | | 99.1% | |
| L2 cache hit ratio | | 98.7% | |
| Memory to D1 refill | 0.666M/sec | 2 | lines |
| Memory to D1 bandwidth | 40.669MB/sec | 128 | bytes |
| L2 to Dcache bandwidth | 3029.859MB/sec | 9536 | bytes |

# Good Cache Alignment

| | | | |
|---|---|---|---|
| Time% | | 0.1% | |
| Time | | 0.000002 | |
| Calls | | 1 | |
| PAPI_L1_DCA | 689.986M/sec | 1333 | ops |
| DC_L2_REFILL_MOESI | 33.645M/sec | 65 | ops |
| DC_SYS_REFILL_MOESI | | 0 | ops |
| BU_L2_REQ_DC | 34.163M/sec | 66 | req |
| User time | 0.000 secs | 5023 | cycles |
| Utilization rate | | 95.1% | |
| L1 Data cache misses | 33.645M/sec | 65 | misses |
| LD & ST per D1 miss | | 20.51 | ops/miss |
| D1 cache hit ratio | | 95.1% | |
| LD & ST per D2 miss | | 1333.00 | ops/miss |
| D2 cache hit ratio | | 100.0% | |
| L2 cache hit ratio | | 100.0% | |
| Memory to D1 refill | | 0 | lines |
| Memory to D1 bandwidth | | 0 | bytes |
| L2 to Dcache bandwidth | 2053.542MB/sec | 4160 | bytes |

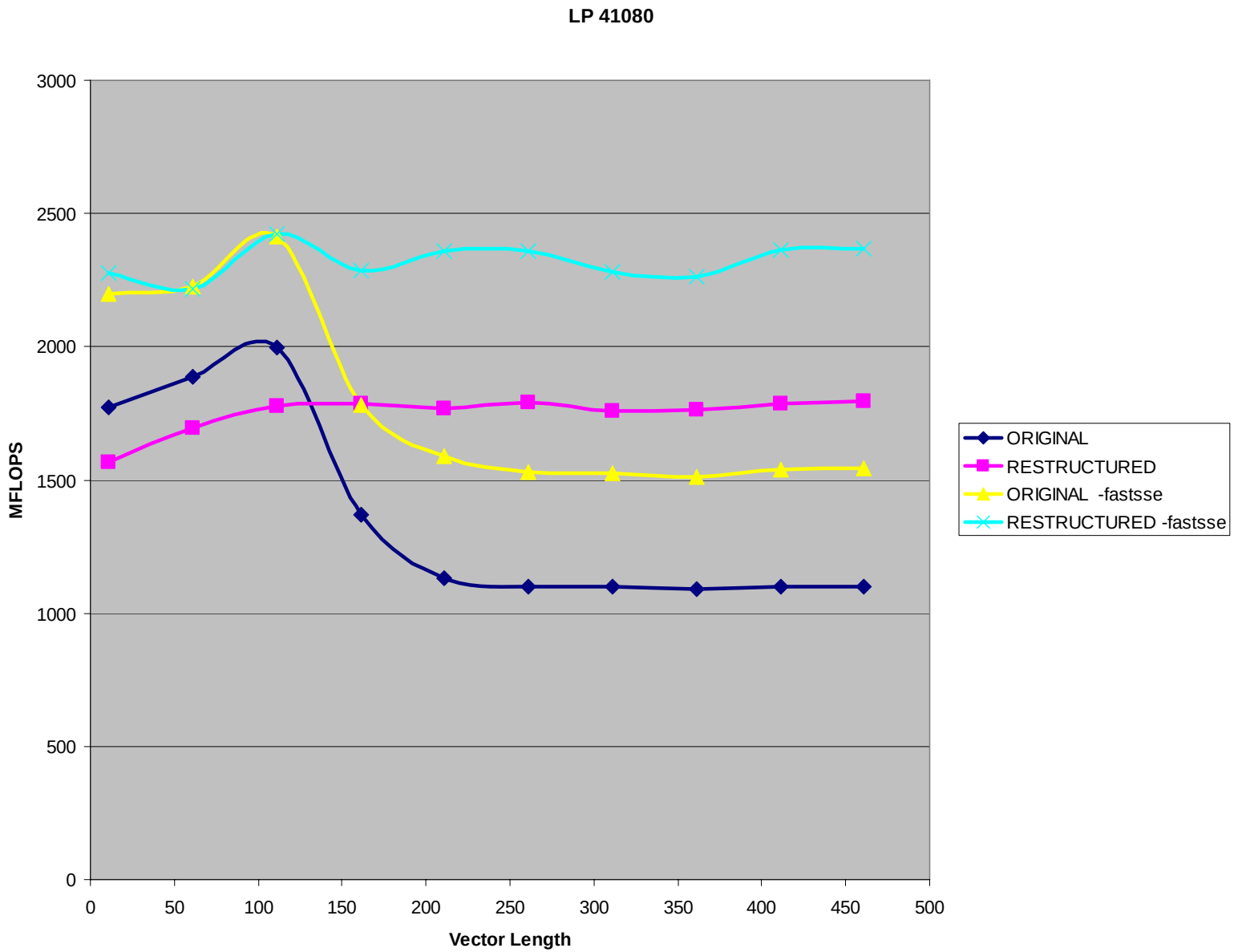
```
C     3 OPERATIONS - 5 OPERANDS     RATIO = 3/5

      DO 41023 I=1, N
       A(I) = B(I) * C(I) + D(I) * E(I)
41023 CONTINUE
```


Comutational Intensity Chart
Loop Length = 61
3500
3000
2500
2000
1500
1000
500
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Computational Intensity
INDIRECT
CONTIGUOUS
STRIDE=128


Computational Intensity
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
0.5
1
1.5
2
2.5
Computational Intensity
INDIRECT
CONTIGUOUS
STRIDE=128

```
C     DIMENSION A(128,N)

      DO 41080  I = 1,N
       A( 1,I) = C1*A(13,I) + C2* A(12,I) + C3*A(11,I) +
     *           C4*A(10,I) + C5* A( 9,I) + C6*A( 8,I) +
     *           C7*A( 7,I) + C0*(A( 5,I) + A( 6,I) )  + A( 3,I)
41080 CONTINUE
```

```
C     DIMENSION B(13,N)

      DO 41081 I = 1,N
       B( 1,I) = C1*B(13,I) + C2* B(12,I) + C3*B(11,I) +
     *           C4*B(10,I) + C5* B( 9,I) + C6*B( 8,I) +
     *           C7*B( 7,I) + C0*(B( 5,I) + B( 6,I) )  + B( 3,I)
41081 CONTINUE
```


LP 41080
MFLOPS
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250
300
350
400
450
500
Vector Length
ORIGINAL
RESTRUCTURED
ORIGINAL -fastsse
RESTRUCTURED -fastsse

```
dimension a(1000,1000,4,4),b(1000,1000,4,4)
real*8 a,b,c,dclock,dtim
b=rand();a=rand()
a=0
ka = 999
ke = 1
ja = 1
je = 1000
dtim = dclock()
ia = 1
ie = 4
      DO 41090 K = KA, KE, -1
        DO 41090 J = JA, JE
          DO 41090 I = IA, IE
            A(K,J,I,3) = A(K,J,I,3) - B(J,K,I,1)*A(K+1,J,I,1)&
         - B(J,K,I,2)*A(K+1,J,I,2) - B(J,K,I,3)*A(K+1,J,I,3)&
         - B(J,K,I,4)*A(K+1,J,I,4) - B(J,K,I,4)*A(K-1,J,I,4)
41090 CONTINUE
dtim = dclock()-dtim
print*,'     MFLOP/SEC',999*1000*4*10/dtim/1e6
end
```

# Bad TLB results – large pages

USER / MAIN_

---

| | | | |
|---|---|---|---|
| Time% | 99.9% | | |
| Time | 0.513916 | | |
| Calls | 1 | | |
| PAPI_TLB_DM | 61.850M/sec | 31785618 misses | |
| PAPI_L1_DCA | 185.066M/sec | 95108750 ops | |
| PAPI_FP_OPS | 77.883M/sec | 40025479 ops | |
| DC_MISS | 42.790M/sec | 21990585 ops | |
| User time | 0.514 secs | 1336184582 cycles | |
| Utilization rate | 100.0% | | |
| HW FP Ops / Cycles | 0.03 ops/cycle | | |
| HW FP Ops / User time | 77.883M/sec | 40025479 ops | 1.5%peak |
| HW FP Ops / WCT | 77.883M/sec | | |
| Computation intensity | 0.42 ops/ref | | |
| LD & ST per TLB miss | 2.99 ops/miss | | |
| LD & ST per D1 miss | 4.32 ops/miss | | |
| D1 cache hit ratio | 76.9% | | |
| % TLB misses / cycle | 2.4% | | |

# Using small pages

USER / MAIN_

---

| | | | |
|---|---|---|---|
| Time% | 100.0% | | |
| Time | 0.679214 | | |
| Calls | 1 | | |
| PAPI_TLB_DM | 24.426M/sec | 16590471 misses | |
| PAPI_L1_DCA | 145.216M/sec | 98632806 ops | |
| PAPI_FP_OPS | 58.930M/sec | 40026496 ops | |
| DC_MISS | 32.376M/sec | 21990324 ops | |
| User time | 0.679 secs | 1765961050 cycles | |
| Utilization rate | 100.0% | | |
| HW FP Ops / Cycles | 0.02 ops/cycle | | |
| HW FP Ops / User time | 58.930M/sec | 40026496 ops | 1.1%peak |
| HW FP Ops / WCT | 58.930M/sec | | |
| Computation intensity | 0.41 ops/ref | | |
| LD & ST per TLB miss | 5.95 ops/miss | | |
| LD & ST per D1 miss | 4.49 ops/miss | | |
| D1 cache hit ratio | 77.7% | | |
| % TLB misses / cycle | 0.9% | | |

# First Restructuring

```
dimension a(4,4,1000,1000),b(4,4,1000,1000)
real*8 a,b,c,dclock,dtim
b=rand();a=rand()
a=0
ka = 999
ke = 1
ja = 1
je = 1000
dtim = dclock()
ia = 1
ie = 4
      DO 41090 K = KA, KE, -1
        DO 41090 J = JA, JE
          DO 41090 I = IA, IE
            A(I,3,K,J) = A(I,3,K,J) - B(I,1,J,K)*A(I,1,K+1,J)&
         - B(I,2,J,K)*A(I,2,K+1,J) - B(I,3,J,K)*A(I,3,K+1,J)&
         - B(I,4,J,K)*A(I,4,K+1,J) - B(I,4,J,K)*A(I,4,K-1,J)
41090 CONTINUE
dtim = dclock()-dtim
print*,'     MFLOP/SEC',999*1000*4*10/dtim/1e6
end
```

# Using Large pages - default

USER / MAIN_

---

| | | |
|---|---|---|
| Time% | 99.8% | |
| Time | 0.208962 | |
| Calls | 1 | |
| PAPI_TLB_DM | 0.342M/sec | 71366 misses |
| PAPI_L1_DCA | 451.354M/sec | 94316361 ops |
| PAPI_FP_OPS | 191.266M/sec | 39967449 ops |
| DC_MISS | 47.861M/sec | 10001137 ops |
| User time | 0.209 secs | 543304201 cycles |
| Utilization rate | 100.0% | |
| HW FP Ops / Cycles | 0.07 ops/cycle | |
| HW FP Ops / User time | 191.266M/sec | 39967449 ops 3.7%peak |
| HW FP Ops / WCT | 191.266M/sec | |
| Computation intensity | 0.42 ops/ref | |
| LD & ST per TLB miss | 1321.59 ops/miss | |
| LD & ST per D1 miss | 9.43 ops/miss | |
| D1 cache hit ratio | 89.4% | |
| % TLB misses / cycle | 0.0% | |

# Using Small Pages

USER / MAIN_

---

| | | | |
|---|---|---|---|
| Time% | 99.8% | | |
| Time | 0.219233 | | |
| Calls | 1 | | |
| PAPI_TLB_DM | 4.587M/sec | 1005738 misses | |
| PAPI_L1_DCA | 426.675M/sec | 93541922 ops | |
| PAPI_FP_OPS | 182.305M/sec | 39967607 ops | |
| DC_MISS | 45.597M/sec | 9996488 ops | |
| User time | 0.219 secs | 570010039 cycles | |
| Utilization rate | 100.0% | | |
| HW FP Ops / Cycles | 0.07 ops/cycle | | |
| HW FP Ops / User time | 182.305M/sec | 39967607 ops | 3.5%peak |
| HW FP Ops / WCT | 182.305M/sec | | |
| Computation intensity | 0.43 ops/ref | | |
| LD & ST per TLB miss | 93.01 ops/miss | | |
| LD & ST per D1 miss | 9.36 ops/miss | | |
| D1 cache hit ratio | 89.3% | | |
| % TLB misses / cycle | 0.2% | | |

# Restructuring 2

```
dimension a(1000,1000,4,4),b(1000,1000,4,4)
real*8 a,b,c,dclock,dtim,scalar,c0,c1,c2,c3,c4,c5,c6
b=rand();a=rand()
a=0
ka = 999
ke = 1
ja = 1
je = 1000
l = 8
dtim = dclock()
ia = 1
ie = 4
          DO 41090 I = IA, IE
        DO 41090 J = JA, JE
      DO 41090 K = KA, KE, -1
            A(K,J,I,3) = A(K,J,I,3) - B(K,J,I,1)*A(K+1,J,I,1)&
         - B(K,J,I,2)*A(K+1,J,I,2) - B(K,J,I,3)*A(K+1,J,I,3)&
         - B(K,J,I,4)*A(K+1,J,I,4) - B(K,J,I,4)*A(K-1,J,I,4)
41090 CONTINUE
dtim = dclock()-dtim
print*,'     MFLOP/SEC',999*1000*4*10/dtim/1e6
end
```

# Large Pages

USER / MAIN_

```
------------------------------------------------------------------------
  Time%                                  99.8%
  Time                                0.159158
  Calls                                  1
  PAPI_TLB_DM                 1.946M/sec     309788 misses
  PAPI_L1_DCA                641.861M/sec  102157923 ops
  PAPI_FP_OPS                251.545M/sec   40035698 ops
  DC_MISS                     50.356M/sec    8014665 ops
  User time                   0.159 secs  413813233 cycles
  Utilization rate                          100.0%
  HW FP Ops / Cycles                         0.10 ops/cycle
  HW FP Ops / User time       251.545M/sec   40035698 ops       4.8%peak
  HW FP Ops / WCT             251.545M/sec
  Computation intensity                      0.39 ops/ref
  LD & ST per TLB miss                     329.77 ops/miss
  LD & ST per D1 miss                       12.75 ops/miss
  D1 cache hit ratio                        92.2%
  % TLB misses / cycle                       0.1%
========================================================================
```

# Small Pages

USER / MAIN_

```
------------------------------------------------------------------------
  Time%                                  99.8%
  Time                                0.159259
  Calls                                  1
  PAPI_TLB_DM                 0.785M/sec     125077 misses
  PAPI_L1_DCA                611.597M/sec   97403340 ops
  PAPI_FP_OPS                251.382M/sec   40035233 ops
  DC_MISS                     50.323M/sec    8014507 ops
  User time                   0.159 secs  414077811 cycles
  Utilization rate                       100.0%
  HW FP Ops / Cycles                      0.10 ops/cycle
  HW FP Ops / User time       251.382M/sec   40035233 ops       4.8%peak
  HW FP Ops / WCT             251.382M/sec
  Computation intensity                   0.41 ops/ref
  LD & ST per TLB miss                    778.75 ops/miss
  LD & ST per D1 miss                     12.15 ops/miss
  D1 cache hit ratio                     91.8%
  % TLB misses / cycle                    0.0%
========================================================================
```

# Restructuring 3

```
dimension a(1000,1000,4,4),b(1000,1000,4,4)
real*8 a,b,c,dclock,dtim
b=rand();a=rand()
a=0
ka = 999
ke = 1
ja = 1
je = 1000
dtim = dclock()
ia = 1
ie = 4
          DO 41090 I = IA, IE
      DO 41090 K = KA, KE, -1
        DO 41090 J = JA, JE
           A(J,K,I,3) = A(J,K,I,3) - B(J,K,I,1)*A(J,K+1,I,1)&
        - B(J,K,I,2)*A(J,K+1,I,2) - B(J,K,I,3)*A(J,K+1,I,3)&
        - B(J,K,I,4)*A(J,K+1,I,4) - B(J,K,I,4)*A(J,K-1,I,4)
41090 CONTINUE
dtim = dclock()-dtim
print*,'     MFLOP/SEC',999*1000*4*10/dtim/1e6
end
```

# Large Pages

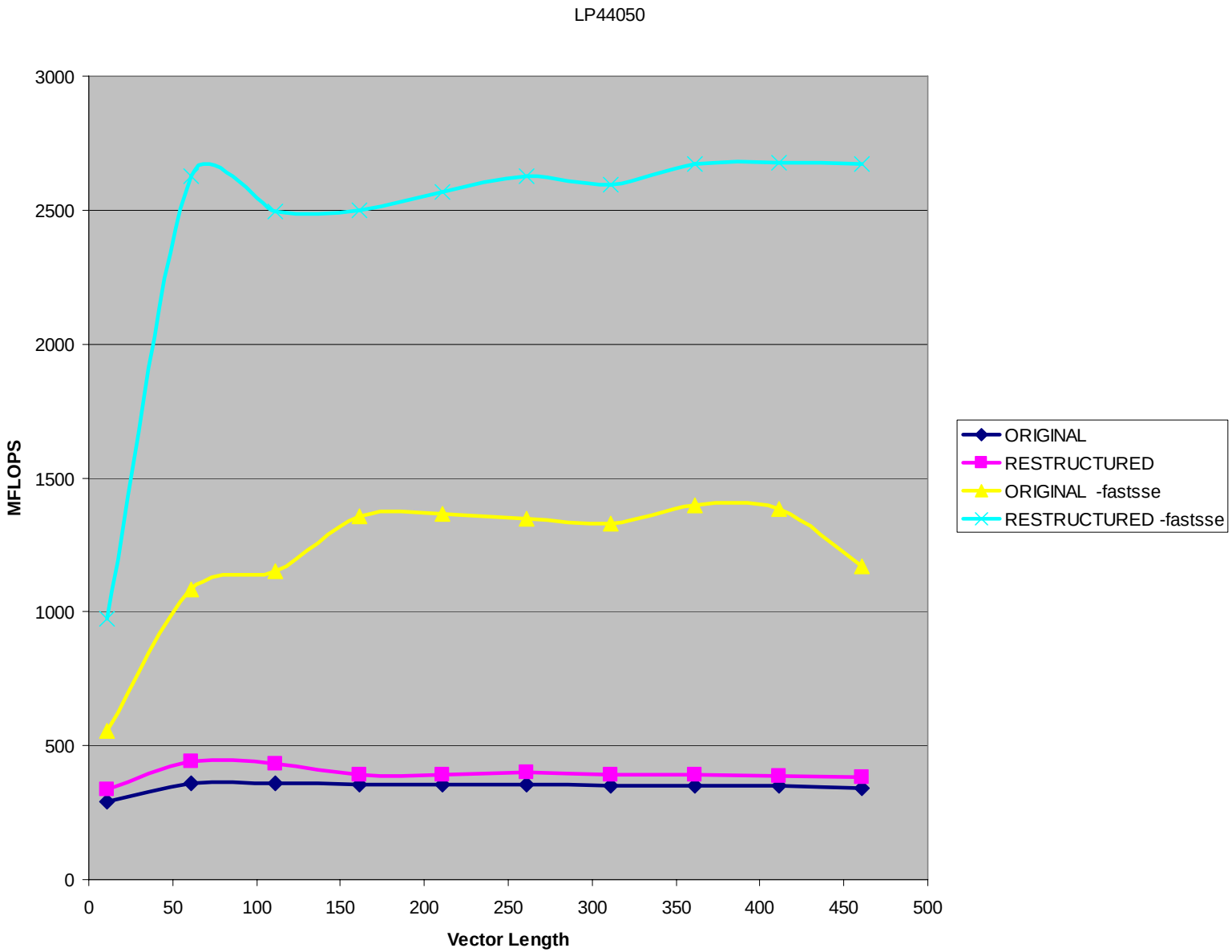
```
================================================================================
USER / MAIN_
--------------------------------------------------------------------------------
  Time%                                    99.7%
  Time                                 0.148981
  Calls                                       1
  PAPI_TLB_DM                1.482M/sec     220852 misses
  PAPI_L1_DCA              675.664M/sec  100662208 ops
  PAPI_FP_OPS              268.738M/sec   40037215 ops
  DC_MISS                   60.473M/sec    9009417 ops
  User time                  0.149 secs  387354592 cycles
  Utilization rate                          100.0%
  HW FP Ops / Cycles                          0.10 ops/cycle
  HW FP Ops / User time    268.738M/sec   40037215 ops
5.2%peak
  HW FP Ops / WCT          268.738M/sec
  Computation intensity                       0.40 ops/ref
  LD & ST per TLB miss                      455.79 ops/miss
  LD & ST per D1 miss                        11.17 ops/miss
  D1 cache hit ratio                         91.0%
  % TLB misses / cycle                        0.1%
================================================================================
```

# Small Pages

```
USER / MAIN_
------------------------------------------------------------------------
  Time%                                              99.8%
  Time                                           0.154248
  Calls                                                 1
  PAPI_TLB_DM                 0.831M/sec          128183 misses
  PAPI_L1_DCA               666.774M/sec       102849427 ops
  PAPI_FP_OPS               259.572M/sec        40038736 ops
  DC_MISS                    58.415M/sec         9010497 ops
  User time                   0.154 secs       401047898 cycles
  Utilization rate                                100.0%
  HW FP Ops / Cycles                                0.10 ops/cycle
  HW FP Ops / User time     259.572M/sec        40038736 ops
5.0%peak
  HW FP Ops / WCT           259.572M/sec
  Computation intensity                             0.39 ops/ref
  LD & ST per TLB miss                            802.36 ops/miss
  LD & ST per D1 miss                              11.41 ops/miss
  D1 cache hit ratio                               91.2%
  % TLB misses / cycle                              0.0%
========================================================================
```

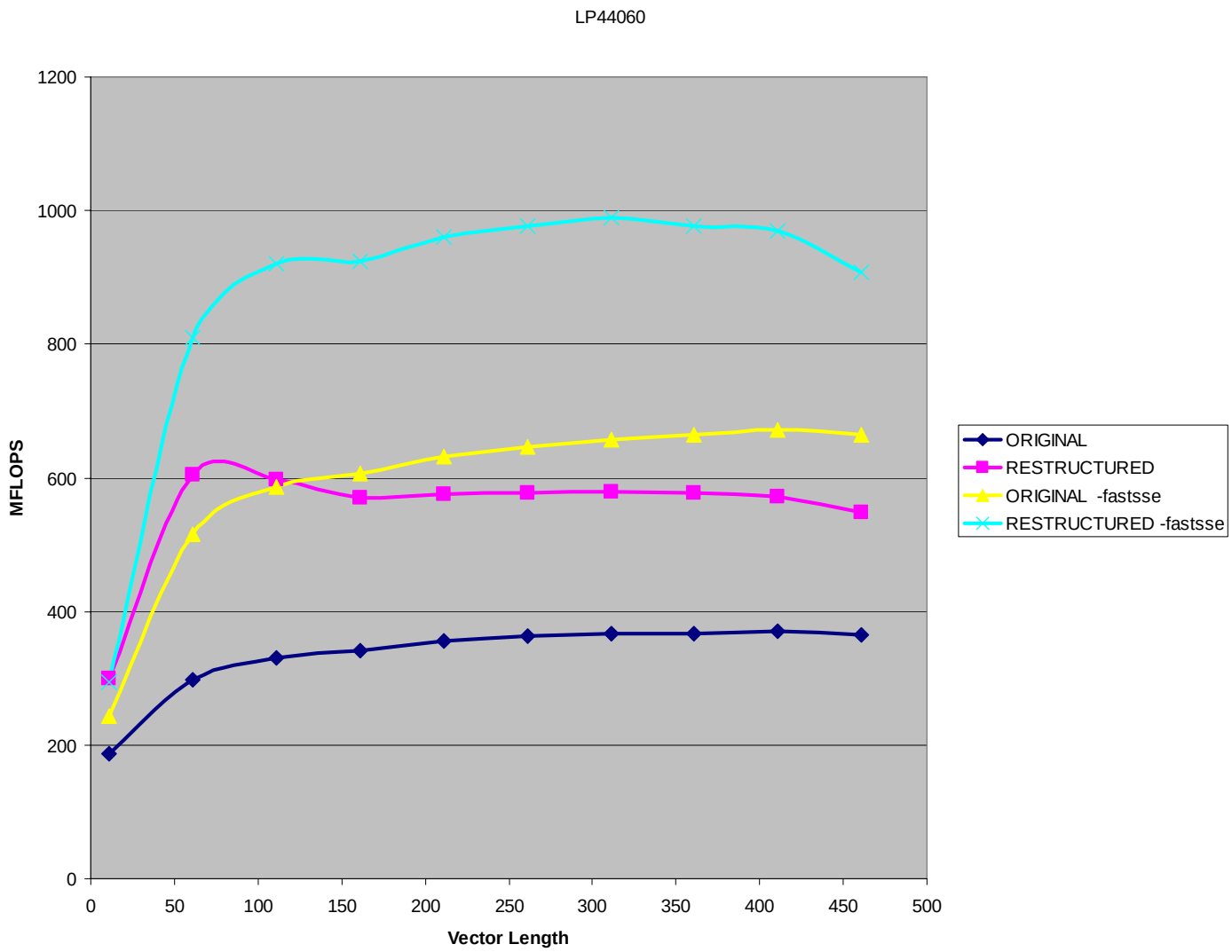
```
 DO 44050 I = 1, N
       DO 44050 J = 1, N
        A(I,J) = 0.0
        DO 44050 K = 1, N
         A(I,J) = A(I,J) + B(I,K) * C(K,J)
44050 CONTINUE
```

```
      DO 44051 J = 1, N
       DO 44051 I = 1, N
        A(I,J) = 0.0
44051 CONTINUE

      DO 44052 K = 1, N
       DO 44052 J = 1, N
        DO 44052 I = 1, N
         A(I,J) = A(I,J) + B(I,K) * C(K,J)
44052 CONTINUE
```


LP44050
MFLOPS
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250
300
350
400
450
500
Vector Length
ORIGINAL
RESTRUCTURED
ORIGINAL -fastsse
RESTRUCTURED -fastsse

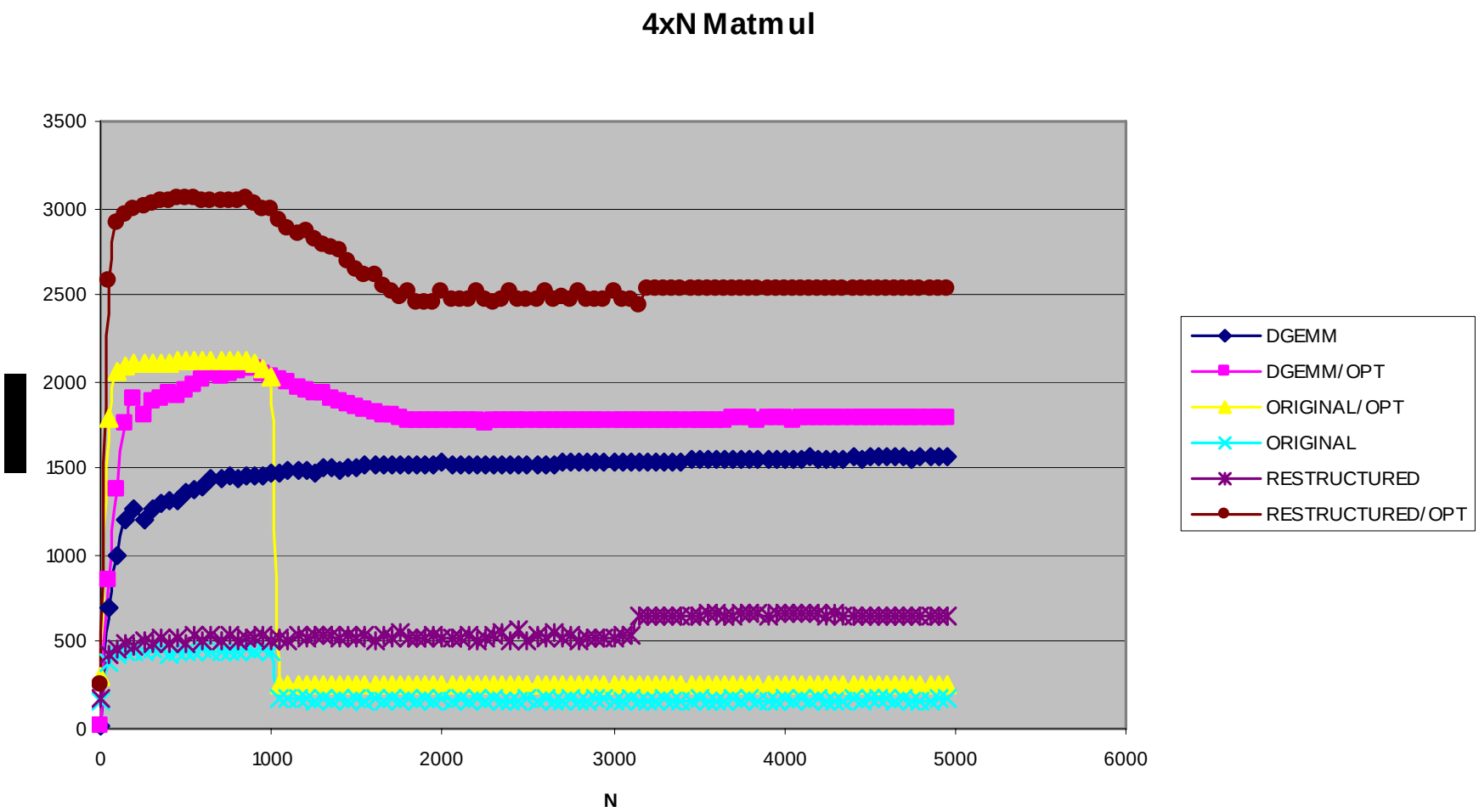
```
      DO 44060 I = 1, N
      A(I) = 0.0
      DO 44060 J = 1, I
       A(I) = A(I) + B(I,J) * C(J,I)
44060 CONTINUE
```

```
      DO 44061 I = 1, N
       A(I) = 0.0
44061 CONTINUE

      DO 44062 J = 1, N
       DO 44062 I = J, N
        A(I) = A(I) + B(I,J) * C(J,I)
44062 CONTINUE
```


LP44060
MFLOPS
Vector Length
0
200
400
600
800
1000
1200
0
50
100
150
200
250
300
350
400
450
500
ORIGINAL
RESTRUCTURED
ORIGINAL -fastsse
RESTRUCTURED -fastsse

```
C      THE ORIGINAL
      DO 46011 J = 1, 4
       DO 46010 I = 1, N
        C(J,I)=0.0
46010 CONTINUE
        DO 46011 K = 1,4
        DO 46011 I = 1,N
         C(J,I) = C(J,I) + A(J,K) * B(K,I)
46011 CONTINUE
```

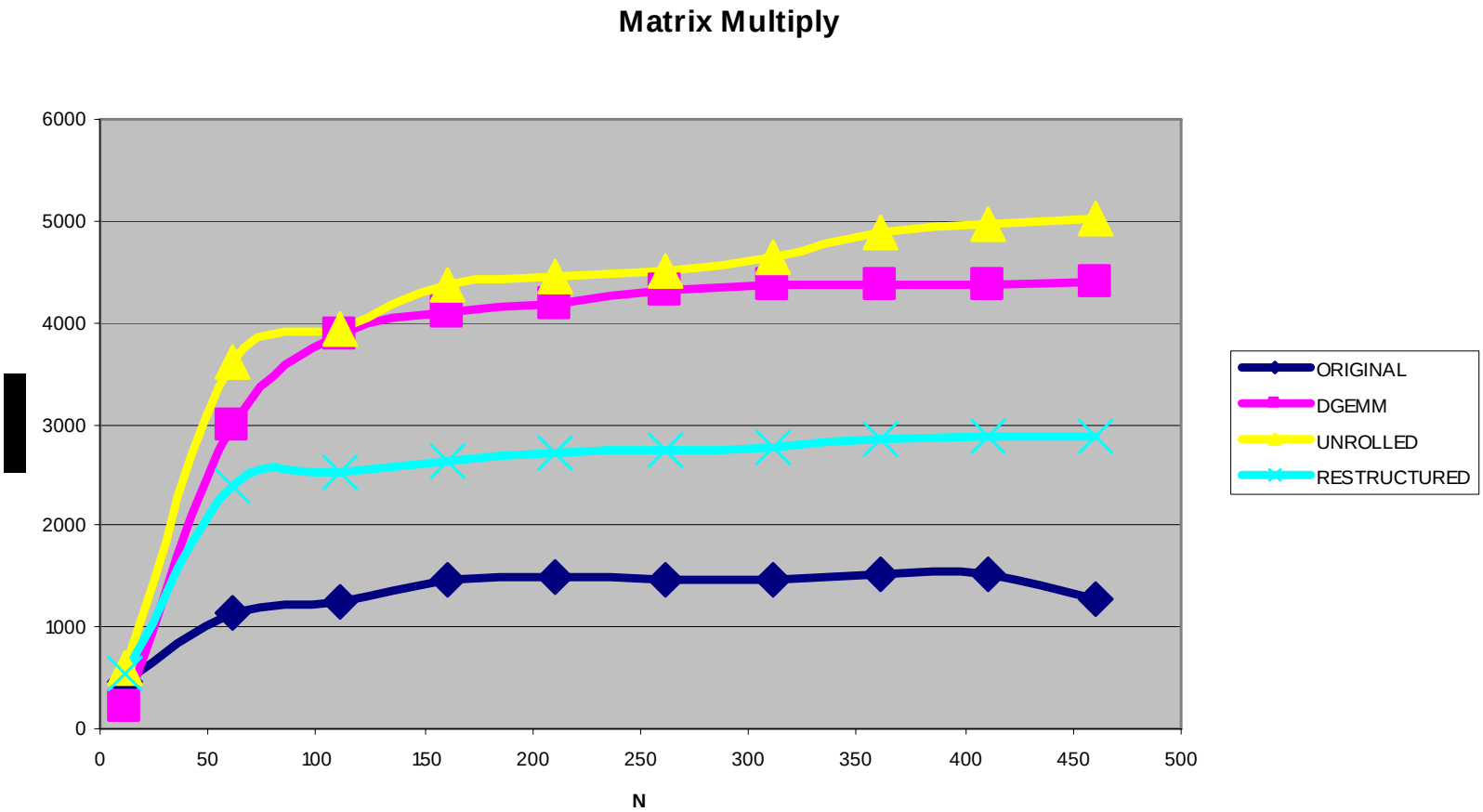
```
C       THE RESTRUCTURED
       DO 46012 I = 1, N
         C(1,I) = A(1,1) * B(1,I) + A(1,2) * B(2,I)
     *          + A(1,3) * B(3,I) + A(1,4) * B(4,I)
         C(2,I) = A(2,1) * B(1,I) + A(2,2) * B(2,I)
     *          + A(2,3) * B(3,I) + A(2,4) * B(4,I)
         C(3,I) = A(3,1) * B(1,I) + A(3,2) * B(2,I)
     *          + A(3,3) * B(3,I) + A(3,4) * B(4,I)
         C(4,I) = A(4,1) * B(1,I) + A(4,2) * B(2,I)
     *          + A(4,3) * B(3,I) + A(4,4) * B(4,I)
46012 CONTINUE
```




OPT have non-power of two as first dimension

```
      DO 46030 J  = 1, N
       DO 46030 I = 1, N
        A(I,J) = 0.
46030 CONTINUE

      DO 46031   K = 1, N
       DO 46031  J = 1, N
        DO 46031 I = 1, N
         A(I,J) = A(I,J) + B(I,K) * C(K,J)
46031 CONTINUE
```

```
C       THE RESTRUCTURED

      DO 46032  J = 1, N
       DO 46032 I = 1, N
        A(I,J)=0.
46032 CONTINUE
C
      DO 46033   K = 1, N-5, 6
       DO 46033  J = 1, N
        DO 46033 I = 1, N
         A(I,J) = A(I,J) + B(I,K  ) * C(K  ,J)
     *                   + B(I,K+1) * C(K+1,J)
     *                   + B(I,K+2) * C(K+2,J)
     *                   + B(I,K+3) * C(K+3,J)
     *                   + B(I,K+4) * C(K+4,J)
     *                   + B(I,K+5) * C(K+5,J)
46033 CONTINUE
C
      DO 46034  KK = K, N
       DO 46034  J = 1, N
        DO 46034 I = 1, N
         A(I,J) = A(I,J) + B(I,KK) * C(KK ,J)
46034 CONTINUE
```


Matrix Multiply
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
300
350
400
450
500
N
ORIGINAL
DGEMM
UNROLLED
RESTRUCTURED

USER / #1.inner product

---

| | | |
|---|---|---|
| Time% | 73.0% | |
| Time | 0.226803 | |
| Calls | 1 | |
| PAPI_TLB_DM | 22 /sec | 5 misses |
| PAPI_L1_DCA | 947.759M/sec | 214953166 ops |
| PAPI_FP_OPS | 1495.678M/sec | 339222112 ops |
| DC_MISS | 177.035M/sec | 40151838 ops |
| User time | 0.227 secs | 589683955 cycles |
| Utilization rate | 100.0% | |
| HW FP Ops / Cycles | 0.58 ops/cycle | |
| HW FP Ops / User time | 1495.678M/sec | 339222112 ops 28.8%peak |
| HW FP Ops / WCT | 1495.671M/sec | |
| Computation intensity | 1.58 ops/ref | |
| LD & ST per TLB miss | 42990633.20 ops/miss | |
| LD & ST per D1 miss | 5.35 ops/miss | |
| D1 cache hit ratio | 81.3% | |
| % TLB misses / cycle | 0.0% | |

USER / #2.unrolled product

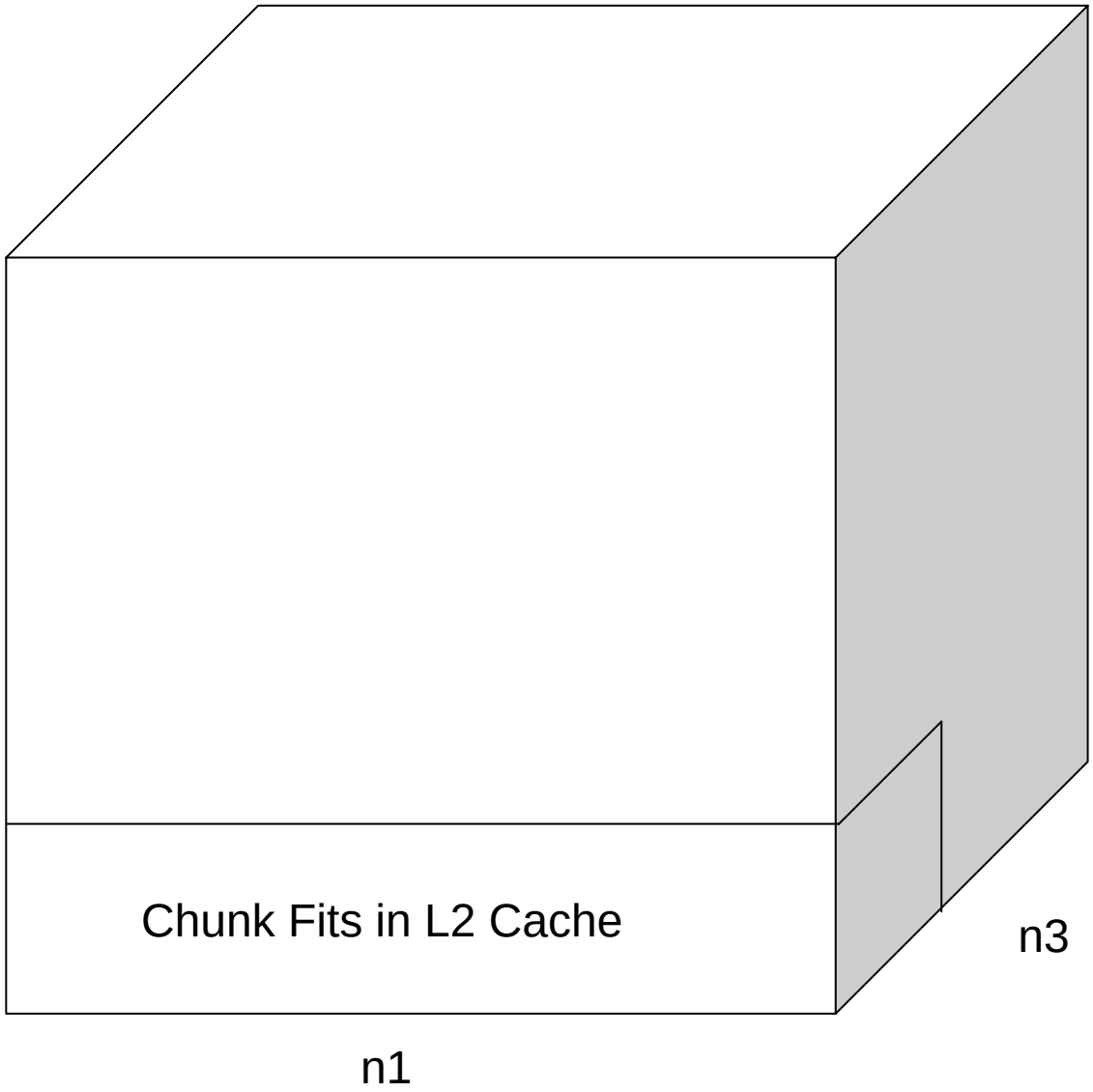
---

| | | | |
|---|---|---|---|
| Time% | 17.9% | | |
| Time | 0.055725 | | |
| Calls | 1 | | |
| PAPI_TLB_DM | 71 /sec | 4 misses | |
| PAPI_L1_DCA | 1967.956M/sec | 109667050 ops | |
| PAPI_FP_OPS | 3062.605M/sec | 170667843 ops | |
| DC_MISS | 25.496M/sec | 1420773 ops | |
| User time | 0.056 secs | 144888568 cycles | |
| Utilization rate | 100.0% | | |
| HW FP Ops / Cycles | 1.18 ops/cycle | | |
| HW FP Ops / User time | 3062.605M/sec | 170667843 ops | 58.9%peak |
| HW FP Ops / WCT | 3062.605M/sec | | |
| Computation intensity | 1.56 ops/ref | | |
| LD & ST per TLB miss | 27416762.50 ops/miss | | |
| LD & ST per D1 miss | 77.19 ops/miss | | |
| D1 cache hit ratio | 98.7% | | |
| % TLB misses / cycle | 0.0% | | |

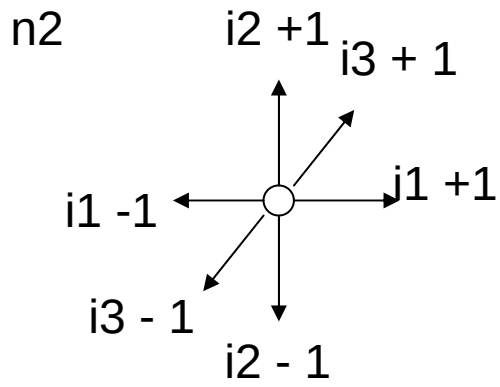
# NPB MG routine RESID

```
      do i3=2,n3-1
           do i2=2,n2-1
              do i1=1,n1
                 u1(i1) = u(i1,i2-1,i3) + u(i1,i2+1,i3)
     >                   + u(i1,i2,i3-1) + u(i1,i2,i3+1)
                 u2(i1) = u(i1,i2-1,i3-1) + u(i1,i2+1,i3-1)
     >                   + u(i1,i2-1,i3+1) + u(i1,i2+1,i3+1)
              enddo
              do i1=2,n1-1
                 r(i1,i2,i3) = v(i1,i2,i3)
     >                         - a(0) * u(i1,i2,i3)
     >                         - a(2) * ( u2(i1) + u1(i1-1) + u1(i1+1) )
     >                         - a(3) * ( u2(i1-1) + u2(i1+1) )
              enddo
           enddo
      enddo
```

```
==========================================================================
USER / resid_
--------------------------------------------------------------------------
  Time%                                              42.4%
  Time                                           12.397761
  Imb.Time                                        0.000370
  Imb.Time%                                           0.0%
  Calls                                               340
  PAPI_L1_DCA                 2719.188M/sec    33711498004 ops
  DC_L2_REFILL_MOESI            79.644M/sec      987402929 ops
  DC_SYS_REFILL_MOESI            4.059M/sec       50318116 ops
  BU_L2_REQ_DC                 129.172M/sec     1601429574 req
  User time                     12.398 secs    32233848320 cycles
  Utilization rate                                  100.0%
  L1 Data cache misses          83.703M/sec     1037721045 misses
  LD & ST per D1 miss                                32.49 ops/miss
  D1 cache hit ratio                                 96.9%
  LD & ST per D2 miss                               669.97 ops/miss
  D2 cache hit ratio                                 96.9%
  L2 cache hit ratio                                 95.2%
  Memory to D1 refill            4.059M/sec       50318116 lines
  Memory to D1 bandwidth       247.723MB/sec     3220359424 bytes
  L2 to Dcache bandwidth      4861.112MB/sec    63193787456 bytes
```




Entire Cube does not fit in L2 Cache
256*256*256*3 arrays
= 402 MBytes




Take data in chunks that Fit in L2 Cache
256*16*32*3 arrays
= 1 MBytes

# Tiling for better Cache utilization

```
do i3block=2,n3-1,BLOCK3
     do i2block=2,n2-1,BLOCK2
     do i3=i3block,min(n3-1,i3block+BLOCK3-1)
        do i2=i2block,min(n2-1,i2block+BLOCK2-1)
           do i1=1, n1
              u1(i1) = u(i1,i2-1,i3) + u(i1,i2+1,i3)
    >                + u(i1,i2,i3-1) + u(i1,i2,i3+1)
              u2(i1) = u(i1,i2-1,i3-1) + u(i1,i2+1,i3-1)
    >                + u(i1,i2-1,i3+1) + u(i1,i2+1,i3+1)
           enddo
           do i1=1, n1
              r(i1,i2,i3) = v(i1,i2,i3)
    >                     - a(0) * u(i1,i2,i3)
    >                     - a(2) * ( u2(i1) + u1(i1-1) + u1(i1+1) )
    >                     - a(3) * ( u2(i1-1) + u2(i1+1) )
           enddo
        enddo
     enddo
     enddo
     enddo
```

```
==============================================================================
USER / resid_
------------------------------------------------------------------------------
  Time%                                              36.3%
  Time                                            8.753226
  Imb.Time                                        0.000596
  Imb.Time%                                           0.0%
  Calls                                                340
  PAPI_L1_DCA                  3861.533M/sec    33800955933 ops
  DC_L2_REFILL_MOESI            116.399M/sec     1018867620 ops
  DC_SYS_REFILL_MOESI             2.755M/sec       24114222 ops
  BU_L2_REQ_DC                  161.490M/sec     1413560527 req
  User time                       8.753 secs    22758444048 cycles
  Utilization rate                                   100.0%
  L1 Data cache misses          119.154M/sec     1042981842 misses
  LD & ST per D1 miss                                 32.41 ops/miss
  D1 cache hit ratio                                  96.9%
  LD & ST per D2 miss                               1401.70 ops/miss
  D2 cache hit ratio                                  98.3%
  L2 cache hit ratio                                  97.7%
  Memory to D1 refill             2.755M/sec       24114222 lines
  Memory to D1 bandwidth        168.145MB/sec     1543310208 bytes
  L2 to Dcache bandwidth       7104.420MB/sec    65207527680 bytes
```

```
 do i3block=2,n3-1,BLOCK3
 do i2block=2,n2-1,BLOCK2
 do i3=i3block,min(n3-1,i3block+BLOCK3-1)
    do i2=i2block,min(n2-1,i2block+BLOCK2-1)
       do i1=1,n1
          u1(i1) = u(i1,i2-1,i3) + u(i1,i2+1,i3)
>                + u(i1,i2,i3-1) + u(i1,i2,i3+1)
          u2(i1) = u(i1,i2-1,i3-1) + u(i1,i2+1,i3-1)
>                + u(i1,i2-1,i3+1) + u(i1,i2+1,i3+1)
       enddo
       do i1=2,n1-1
          r(i1,i2,i3) = v(i1,i2,i3)
>                     - a(0) * u(i1,i2,i3)
>                     - a(2) * ( u2(i1) + u1(i1-1) + u1(i1+1) )
>                     - a(3) * ( u2(i1-1) + u2(i1+1) )
       enddo
    enddo
 enddo
 enddo
 enddo
```

```
do i3block=2,n3-1,BLOCK3
 do i2block=2,n2-1,BLOCK2
 do i3=i3block,min(n3-1,i3block+BLOCK3-1)
    do i2=i2block,min(n2-1,i2block+BLOCK2-1)
       do i1=2,n1-1
          u21 = u(i1,i2-1,i3-1) + u(i1,i2+1,i3-1)
>               + u(i1,i2-1,i3+1) + u(i1,i2+1,i3+1)
          u21p1 = u(i1+1,i2-1,i3-1) + u(i1+1,i2+1,i3-1)
>               + u(i1+1,i2-1,i3+1) + u(i1+1,i2+1,i3+1)
          u21m1 = u(i1-1,i2-1,i3-1) + u(i1-1,i2+1,i3-1)
>               + u(i1-1,i2-1,i3+1) + u(i1-1,i2+1,i3+1)
          u11p1 = u(i1+1,i2-1,i3) + u(i1+1,i2+1,i3)
>               + u(i1+1,i2,i3-1) + u(i1+1,i2,i3+1)
          u11m1 = u(i1-1,i2-1,i3) + u(i1-1,i2+1,i3)
>               + u(i1-1,i2,i3-1) + u(i1-1,i2,i3+1)
          r(i1,i2,i3) = v(i1,i2,i3)
>                     - a(0) * u(i1,i2,i3)
>                     - a(2) * ( u21 + u11m1 + u11p1 )
>                     - a(3) * ( u21m1 + u21p1 )
       enddo
    enddo
 enddo
 enddo
 enddo
```

```
USER / resid_
------------------------------------------------------------------------
  Time%                                              37.7%
  Time                                           9.132935
  Imb.Time                                       0.003440
  Imb.Time%                                          0.1%
  Calls                                               340
  PAPI_TLB_DM                  0.139M/sec         1270096 misses
  PAPI_L1_DCA                3694.219M/sec     33739238309 ops
  PAPI_FP_OPS                2601.948M/sec     23763548027 ops
  DC_MISS                     111.833M/sec      1021371774 ops
  User time                     9.133 secs     23745753175 cycles
  Utilization rate                                  100.0%
  HW FP Ops / Cycles                                  1.00 ops/cycle
  HW FP Ops / User time      2601.948M/sec     23763548027 ops
25.0%peak
  HW FP Ops / WCT            2601.948M/sec
  Computation intensity                               0.70 ops/ref
  LD & ST per TLB miss                            26564.32 ops/miss
  LD & ST per D1 miss                                33.03 ops/miss
  D1 cache hit ratio                                 97.0%
  % TLB misses / cycle                                0.0%
```

```
USER / resid_
------------------------------------------------------------------------
  Time%                                              39.6%
  Time                                           9.752716
  Imb.Time                                       0.002081
  Imb.Time%                                          0.0%
  Calls                                               340
  PAPI_TLB_DM                    0.115M/sec       1119418 misses
  PAPI_L1_DCA                 2792.319M/sec   27232706384 ops
  PAPI_FP_OPS                 3488.881M/sec   34026076279 ops
  DC_MISS                      104.718M/sec    1021283533 ops
  User time                      9.753 secs   25357072370 cycles
  Utilization rate                                  100.0%
  HW FP Ops / Cycles                                  1.34 ops/cycle
  HW FP Ops / User time       3488.881M/sec   34026076279 ops     33.5%peak
  HW FP Ops / WCT             3488.881M/sec
  Computation intensity                               1.25 ops/ref
  LD & ST per TLB miss                            24327.56 ops/miss
  LD & ST per D1 miss                                26.67 ops/miss
  D1 cache hit ratio                                 96.2%
  % TLB misses / cycle                                0.0%
```

```
USER / resid_
------------------------------------------------------------------------
  Time%                                              38.3%
  Time                                           9.162149
  Imb.Time                                       0.006363
  Imb.Time%                                          0.1%
  Calls                                              340
  PAPI_L1_DCA                 3682.405M/sec   33739250204 ops
  DC_L2_REFILL_MOESI           111.475M/sec    1021369289 ops
  DC_SYS_REFILL_MOESI            2.964M/sec      27157915 ops
  BU_L2_REQ_DC                 157.164M/sec    1439982850 req
  User time                      9.162 secs   23821945786 cycles
  Utilization rate                                 100.0%
  L1 Data cache misses         114.439M/sec    1048527204 misses
  LD & ST per D1 miss                               32.18 ops/miss
  D1 cache hit ratio                                96.9%
  LD & ST per D2 miss                             1242.34 ops/miss
  D2 cache hit ratio                                98.1%
  L2 cache hit ratio                                97.4%
  Memory to D1 refill            2.964M/sec      27157915 lines
  Memory to D1 bandwidth       180.914MB/sec    1738106560 bytes
  L2 to Dcache bandwidth      6803.916MB/sec   65367634496 bytes
```

```
USER / resid_
------------------------------------------------------------------------
  Time%                                                39.4%
  Time                                              9.699533
  Imb.Time                                          0.003564
  Imb.Time%                                             0.1%
  Calls                                                  340
  PAPI_L1_DCA                  2807.643M/sec    27232738768 ops
  DC_L2_REFILL_MOESI            105.292M/sec     1021281565 ops
  DC_SYS_REFILL_MOESI             2.366M/sec       22945693 ops
  BU_L2_REQ_DC                  114.970M/sec     1115152062 req
  User time                       9.700 secs    25218702347 cycles
  Utilization rate                                    100.0%
  L1 Data cache misses          107.658M/sec     1044227258 misses
  LD & ST per D1 miss                                  26.08 ops/miss
  D1 cache hit ratio                                   96.2%
  LD & ST per D2 miss                                1186.83 ops/miss
  D2 cache hit ratio                                   97.9%
  L2 cache hit ratio                                   97.8%
  Memory to D1 refill             2.366M/sec       22945693 lines
  Memory to D1 bandwidth        144.388MB/sec     1468524352 bytes
  L2 to Dcache bandwidth       6426.524MB/sec    65362020160 bytes
```

```
USER / psinv_
-----------------------------------------------------------------------
  Time%                                              15.4%
  Time                                           3.737211
  Imb.Time                                       0.003707
  Imb.Time%                                          0.2%
  Calls                                               336
  PAPI_TLB_DM                  0.021M/sec           80087 misses
  PAPI_L1_DCA                4833.682M/sec     18064847976 ops
  PAPI_FP_OPS                3462.803M/sec     12941480645 ops
  DC_MISS                     114.023M/sec       426137415 ops
  User time                     3.737 secs      9716940584 cycles
  Utilization rate                                  100.0%
  HW FP Ops / Cycles                                  1.33 ops/cycle
  HW FP Ops / User time      3462.803M/sec     12941480645 ops
33.3%peak
  HW FP Ops / WCT            3462.803M/sec
  Computation intensity                               0.72 ops/ref
  LD & ST per TLB miss                           225565.30 ops/miss
  LD & ST per D1 miss                                42.39 ops/miss
  D1 cache hit ratio                                 97.6%
  % TLB misses / cycle                                0.0%
```

```
USER / psinv_
-----------------------------------------------------------------
---------
  Time%                                              14.8%
  Time                                             3.630726
  Imb.Time                                         0.000418
  Imb.Time%                                            0.0%
  Calls                                                 336
  PAPI_TLB_DM                    0.023M/sec          81706 misses
  PAPI_L1_DCA                 3413.252M/sec     12392623798 ops
  PAPI_FP_OPS                 4339.828M/sec     15756780901 ops
  DC_MISS                      117.264M/sec       425753159 ops
  User time                      3.631 secs      9439920160 cycles
  Utilization rate                                   100.0%
  HW FP Ops / Cycles                                   1.67 ops/cycle
  HW FP Ops / User time       4339.828M/sec     15756780901 ops
41.7%peak
  HW FP Ops / WCT             4339.828M/sec
  Computation intensity                                1.27 ops/ref
  LD & ST per TLB miss                            151673.36 ops/miss
  LD & ST per D1 miss                                 29.11 ops/miss
  D1 cache hit ratio                                  96.6%
  % TLB misses / cycle                                 0.0%
```

# How to Use MPI on the Cray XT

Howard Pritchard
Mark Pagel
Cray Inc.

# Outline

- XT MPI implementation overview
- Using MPI on the XT
- Recently added performance improvements
- Additional Documentation

# XT MPI implementation overview

- Portals

- MPI implementation

# Portals API

- API designed to fit MPI message matching rules
- Emphasis on application bypass, off loading of message passing work from application process
- Emphasis on scalability
- Similar in concept to Quadrics t-ports

# XT MPI

- Based on MPICH2
- Cray developed a Portals ADI3 device for MPICH2
  - Portions of design come from earlier MPICH1 portals ADI device
  - Portions from CH3 ADI3 device in MPICH2
- Supports MPI-2 RMA (one-sided)
- Full MPI-IO support
- Does not support MPI-2 dynamic process management (chapter 5 of MPI-2 standard)

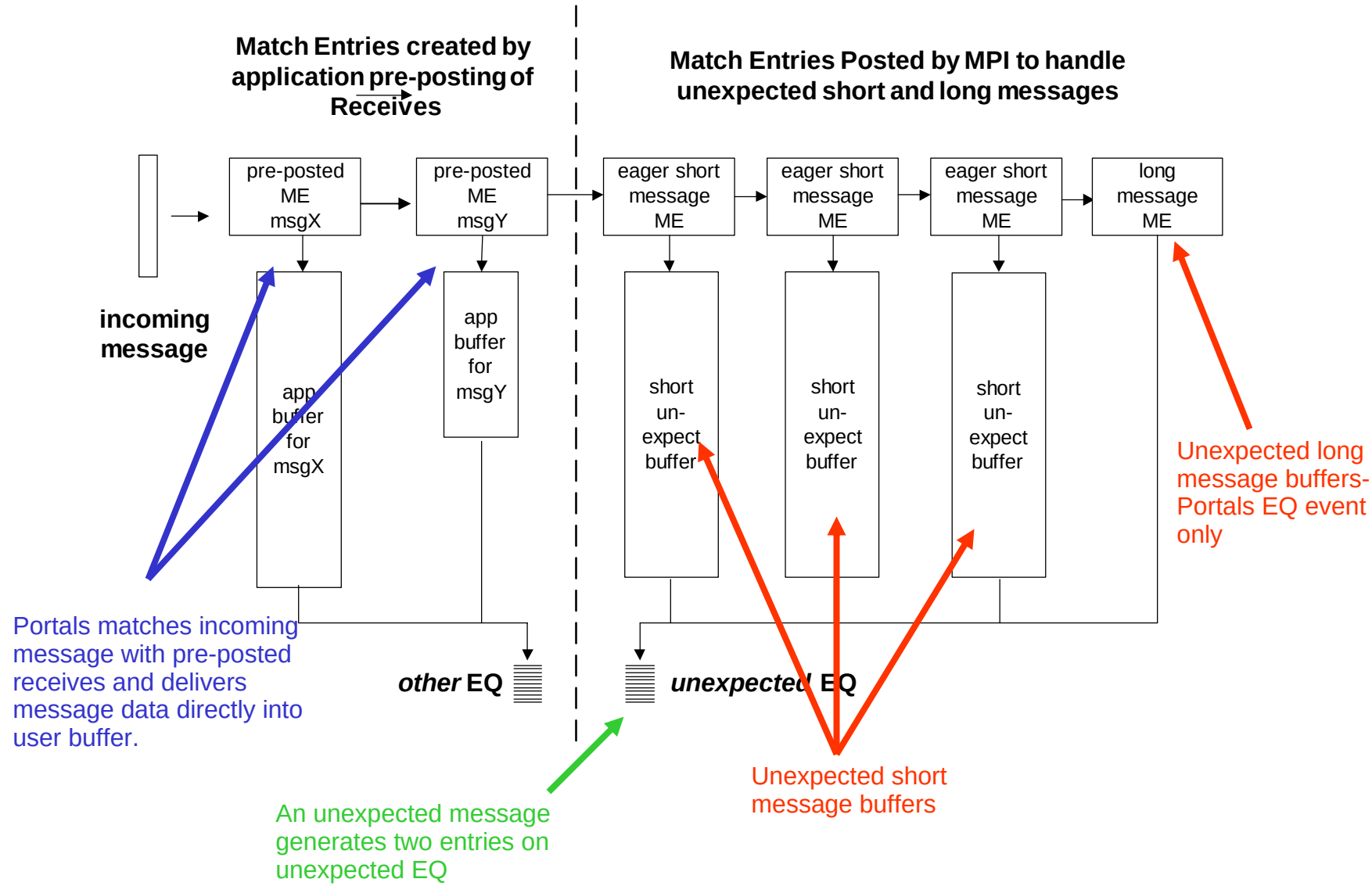
# XT MPI Implementation

**Two protocols are used for handling basic MPI-1 send/recv style messaging:**

- Eager protocol for short messages
- Two protocols for long messages

**But first we will talk about the receive side, since that is where Portal's important features with respect to MPI are most evident…**

# XT MPI – Receive Side



# XT MPI Short Message Protocol- Sending side

- If message to send has no more than 128000 bytes of data, the short message, eager protocol is used by sender.
- Sender assumes the receiver has space for the message in an MPI-internal receive buffer and 2 slots in the *unexpected Event Queue.*
- For very short messages (1024 bytes or shorter), sender copies data to internal buffer before sending *PtlPut* request to Portals

# XT MPI Short Message Protocol- Receive side

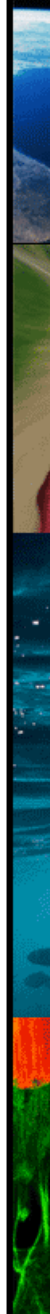
**Two things can happen:**

- No matching receive is posted when message arrives. In this case the message goes into one of the unexpected buffers, later to be copied into application buffer. Two events are delivered to the *unexpected* event queue.
- A matching receive was posted. In this case the incoming message data goes directly into

# XT MPI Long Message Protocol

**Two Protocols:**

- Receiver-pull based method (default)
- Eager method



# XT MPI Long Message Protocol-Receiver Pull method

- Sender encodes information in a Portals Header with information for receiver to get the data from the application buffer
- Sender sends just this Portals header.
- Receiver picks up the Portals header and decodes it. If matching receive already posted, GET the data using PtlGet to store directly in receiver application buffer.
- If no buffer posted, just leave on internal unexpected list till matching receive is

# XT MPI Long Message Protocol-Eager Send Method(1)

- Sender sends the message data much as with the short protocol. It is assumed that there are sufficient resources at the receiver (slots in *unexpected Event Queue*). Sender requests a PTL_ACK_REQ.

- If there is no matching posted receive, the message header is matched to the long protocol match entry.

# XT3 MPI Long Message Protocol-Eager Send Method(2)

- If there is a matching posted receive Portals delivers the message data directly into the application buffer and replies to the sender with a PTL_EVENT_ACK. This tells the sender that the send is complete.

- Otherwise, when the matching receive is posted by the application, the receiver GETs the data from the source, much like in the *receiver-pull* method. The PtlGET generates a *PTL_EVENT_REPLY_END* at the sender. This tells the sender the send is complete.

# XT MPI Long Message Protocol-Why Two Methods?

- COP (CAM OVERFLOW PROTECTION) causes significant delays in returns from blocking sends, causing noticeable bandwidth drops using *eager long* method

- Applications that insure receives are pre-posted should use the *eager* method. This is done by setting the MPICH_PTLS_EAGER_LONG environment variable.

# Using MPI on XT

- Optimizing MPI point-to-point calls for XT
- MPI derived datatypes
- Collective Operations
- MPI-2 RMA
- Odds and ends
- Environment variable summary
- “What does this mean?”

# Optimizing MPI Point-to-point calls(1)

- Use non-blocking send/recvs when it is possible to overlap communication with computation
- If possible, pre-post receives before sender posts the matching send
- Don’t go crazy pre-posting receives though. May hit Portals internal resource limitations.

# Optimizing MPI Point-to-point calls(2)

- Normally best to avoid MPI_(I)probe. Eliminates many of the advantages of the Portals network protocol stack.
- No significant performance advantages associated with persistent requests.
- For many very small messages, it may be better to aggregate data to reduce the number of messages
- But don't aggregate too much. Portals/Seastar ~1/2 of asymptotic bandwidth at ~4-8 KB.

# MPI derived datatypes

- XT MPI uses MPICH2 *dataloop* representation of derived data types, shown to be superior to MPICH1, at least for micro-processors
- However, XT hardware not designed to handle non-contiguous data transfers efficiently, still better to use contiguous data types if possible
  - MPI packs data on sender side
  - MPI allocates temporary buffer on receive side and then unpacks data into application receive buffer
  - Opteron more active in sending/receiving

# Collective Operations

- XT MPI uses MPICH2 default collectives
  with some optimized algorithms enabled by message size (more on this later)
- Environment variables available for additional optimized algorithms
- In some cases it may be better to replace collective operations with point to point communications to overlap communication with computation

# XT MPI-2 RMA

- XT MPI supports all RMA operations
- Based on MPICH2 CH3 device RMA
  - Layered on top of internal send/recv protocol
- Designed for functionality, not performance.
- Little opportunity for overlapping of communication with computation when using MPI-2 RMA on XT.
- Almost all communication occurs at end of exposure epochs or in *MPI_Win_free.*

# Odds and Ends

- MPI_Wtime is not global
- MPI_LONG_DOUBLE datatype is not supported
- MPI_Send to self will cause application to abort for any message size *(if a matching receive is not pre-posted).*
- Topology-related functions (***MPI_Cart_create***, etc.) are not optimized in current releases

# XT3 MPI environment variables(1)

| environment variable | description | default |
|---|---|---|
| MPICH_MAX_SHORT_MSG_SIZE | Sets the maximum size of a message in bytes that can be sent via the short(eager) protocol. | 128000 bytes |
| MPICH_UNEX_BUFFER_SIZE | Overrides the size of the buffers allocated to the MPI unexpected receive queue. | 60 MB |
| MPICH_PTLS_EAGER_LONG | Enables eager long path for message delivery. | disabled |

# XT MPI environment variables(2)

| environment variable | description | default |
|---|---|---|
| MPICH_PTL_UNEX_EVENTS | Specifies size of event queue associated with unexpected messages. Bear in mind that each unexpected message generates 2 events on this queue. | 20480 events |
| MPICH_PTL_OTHER_EVENTS | Specifies size of event queue associated with handling of Portals events not associated with unexpected messages. | 2048 events |
| MPICH_MAX_VSHORT_MSG_SIZE | Specifies in bytes the maximum size message to be considered for the vshort path. | 1024 bytes |

# XT MPI environment variables(3)

| environment variable | description | default |
|---|---|---|
| MPICH_VSHORT_BUFFERS | Specifies the number of 16384 byte buffers to be pre-allocated for the send side buffering of messages for the vshort protocol. | 32 buffers |
| MPICH_DBMASK | Set this variable to 0x200 to get a coredump and traceback when MPI encounters errors either from incorrect arguments to MPI calls, or internal resource limits being hit. | not enabled |
| MPICH_PTL_SEND_CREDITS | Sets the number of send credits from one process to another (send credits are processed by the MPI progress engine of the receiving process). Note: The value -1 sets the number of send credits equal to the size of the unexpected event queue divided by the number of processes in the job, which should prevent queue overflow in any situation. | 0 (disabled) |

# XT MPI Environment variables(4)

| environment variable | description | default |
|---|---|---|
| MPICH_ALLTOALL_SHORT_MSG | Adjusts the cut-off point for which the store and forward Alltoall algorithm is used for short messages | 512 bytes |
| MPICH_BCAST_ONLY_TREE | Setting to 1 or 0, respectively disables or enables the ring algorithm in the implementation for MPI_Bcast for communicators of nonpower of two size. | 1 |
| MPICH_REDUCE_SHORT_MSG | Adjusts the cut-off point for which a reduce-scatter algorithm is used. A binomial tree algorithm is used for smaller values. | 64K bytes |
| MPICH_ALLTOALLVW_SENDRECV | Disables the flow-controlled Alltoall algorithm. When disabled, the pair-wise sendrecv algorithm is used which is the default for messages larger than 32K bytes. | not enabled |

# What does this mean? (1)

**If you see this error message:**

```
internal ABORT - process 0: Other MPI error, error stack:
MPIDI_PortalsU_Request_PUPE(317): exhausted unexpected receive queue

buffering increase via env. var. MPICH_UNEX_BUFFER_SIZE
```

**It means:**

**The application is sending too many short, unexpected messages to a particular receiver.**

**Try doing this to work around the problem:**

**Increase the amount of memory for MPI buffering using the MPICH_UNEX_BUFFER_SIZE variable(default is 60 MB) and/or decrease the short message threshold using the MPICH_MAX_SHORT_MSG_SIZE (default is 128000 bytes) variable. May want to set MPICH_DBMASK to 0x200 to get a traceback/coredump to learn where in application this problem is occurring.**

# What does this mean? (2)

**If you see this error message:**

```
Assertion failed in file
/notbackedup/users/rsrel/rs64.REL_1_4_06.060419.Wed/pe/computelibs/m
pich2/src/mpid/portals32/src/portals_init.c at line 193:
MPIDI_Portals_unex_block_size > MPIDI_Portals_short_size
```

**It means:**

**The appearance of this assertion means that the size of the unexpected buffer space is too small to contain even 1 unexpected short message.**

**Try doing this to work around the problem:**

**User needs to check their MPICH environment settings to make sure there are no conflicts between the setting of the MPICH_UNEX_BUFFER_SIZE variable and the setting for MPICH_MAX_SHORT_MSG_SIZE. Note setting MPICH_UNEX_BUFFER_SIZE too large ( > 2 GB) may confuse MPICH and also lead to this message.**

# What does this mean? (3)

## If you see this error message:

```
[0] MPIDI_PortalsU_Request_FDU_or_AEP: dropped event on unexpected
receive queue, increase

[0] queue size by setting the environment variable MPICH_PTL_UNEX_EVENTS
```

**It means:**

**You have exhausted the event queue entries associated with the unexpected queue. The default size is 20480.**

**Try doing this to work around the problem:**

**You can increase the size of this queue by setting the environment variable MPICH_PTL_UNEX_EVENTS to some value higher than 20480.**

# What does this mean? (4)

## If you see this error message:

```
[0] MPIDI_Portals_Progress: dropped event on "other" queue, increase
[0] queue size by setting the environment variable MPICH_PTL_OTHER_EVENTS

aborting job: Dropped Portals event
```

## It means:

**You have exhausted the event queue entries associated with the "other" queue. This can happen if the application is posting many non-blocking sends, or a large number of pre-posted receives are being posted, or many MPI-2 RMA operations are posted in a single epoch. The default size of the other EQ is 2048.**

## Try doing this to work around the problem:

**You can increase the size of this queue by setting the environment variable MPICH_PTL_OTHER_EVENTS to some value higher than the 2048 default.**

# What does this mean? (5)

## If you see this error message:

```
0:(/notbackedup/users/rsrel/rs64.REL_1_3_12.051214.Wed/pe/compute
libs/mpich2/src/mpid/portals32/src/portals_progress.c:642)

PtlEQAlloc failed : PTL_NO_SPACE
```

## It means:

**You have requested so much EQ space for MPI (and possibly SHMEM if using both in same application) that there are not sufficient Portals resources to satisfy the request.**

## Try doing this to work around the problem:

**You can decrease the size of the event queues by setting the environment variable MPICH_PTL_UNEXPECTED_EVENTS and**

**MPICH_PTL_OTHER_EVENTS to smaller values.**

# What does this mean? (6)

## If you see this error message:

```
aborting job: Fatal error in MPI_Init: Other MPI error, error
stack: MPIR_Init_thread(195): Initialization failed
MPID_Init(170): failure during portals initialization
MPIDI_Portals_Init(321): progress_init failed

MPIDI_PortalsI_Progress_init(653): Out of memory
```

## It means:

**There is not enough memory on the nodes for the program plus MPI buffers to fit.**

## Try doing this to work around the problem:

**You can decrease the amount of memory that MPI is using for buffers by using MPICH_UNEX_BUFFER_SIZE environment variable.**

# Recently added XT MPI Performance Improvements

- Portals improvements (In 1.5.07, 1.4.28)
  - Send to self short-circuit optimizations
  - Symmetric portals syscall optimizations
  - Portals API extended (PtlMEMDPost)
- MPI use of PtlMEMDPost (In 1.5.07, 1.4.28)
- New MPI env variables
  - MPICH_RANK_REORDER_METHOD
  - MPI_COLL_OPT_ON
  - MPICH_FAST_MEMCPY

# New MPI env variables

- MPICH_RANK_REORDER_METHOD env variable to control rank placement (In 1.5.08 and 1.4.30)
  - yod default placement:

| NODE | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| RANK | 0&4 | 1&5 | 2&6 | 3&7 |

  - Setting env to “1” causes SMP style placement

| NODE | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| RANK | 0&1 | 2&3 | 4&5 | 6&7 |

Note that aprun(CNL) uses SMP-style placement by default.

# MPICH_RANK_REORDER_METHOD (cont.)

- Setting env to “2” causes folded rank placement

| NODE | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| RANK | 0&7 | 1&6 | 2&5 | 3&4 |

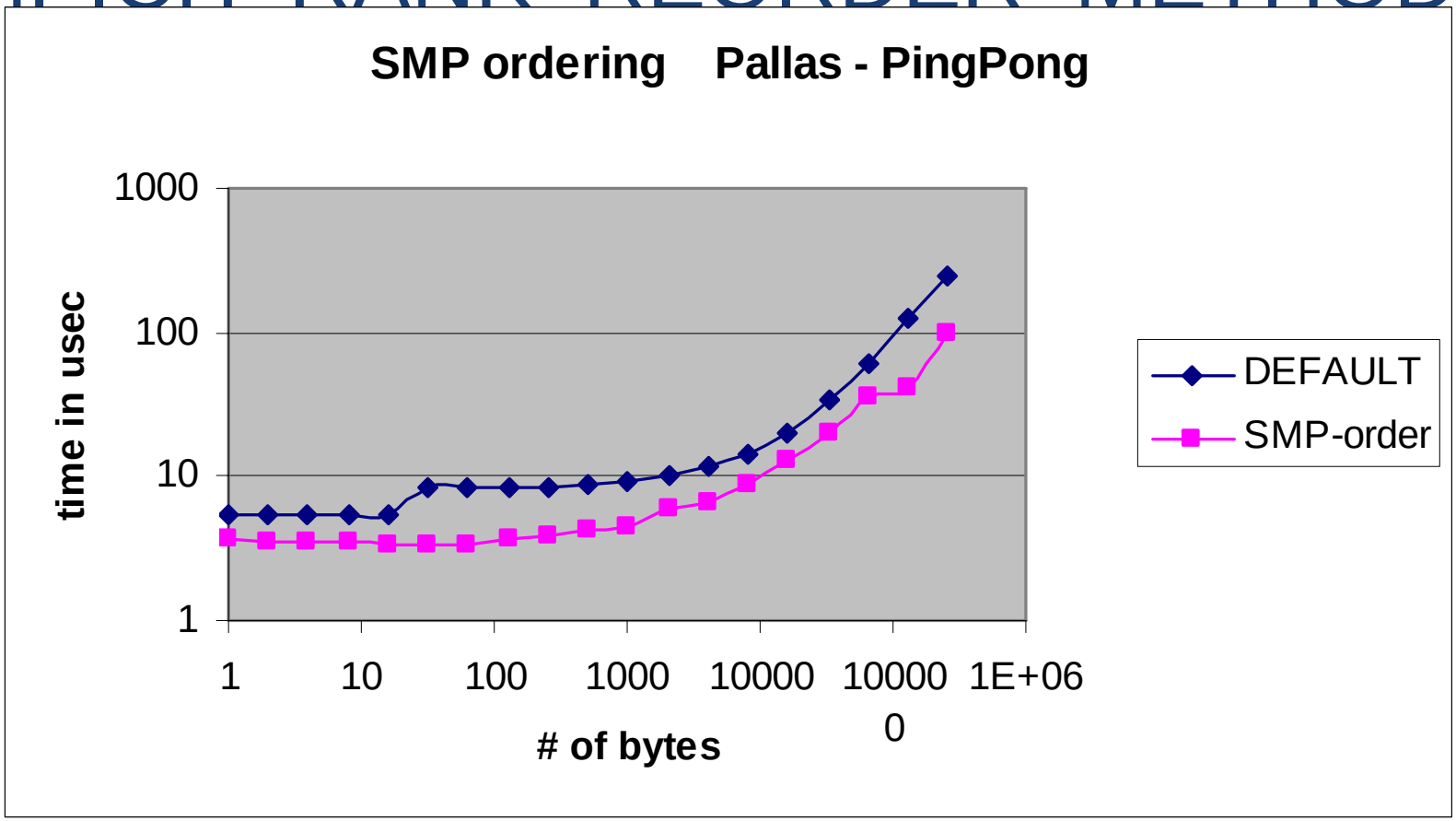
- Setting env to “3” causes custom rank placement using “MPICH_RANK_ORDER” file. For example:

| | |
|---|---|
| 0-15 | Places the ranks in SMP-style order |
| 15-0 | Places ranks 15&14 on the first node, 13&12 on next, etc. |
| 0,4,1,5,2,6,3,7 | Places ranks 0&4 on the first node, 1&5 on the next, 2&6 together, and 3&7 together. |

  - MPICH_RANK_FILE_BACKOFF
    Specifies the number of milliseconds for backoff.
  - MPICH_RANK_FILE_GROUPSIZE
    Specifies the number of ranks in the group size.

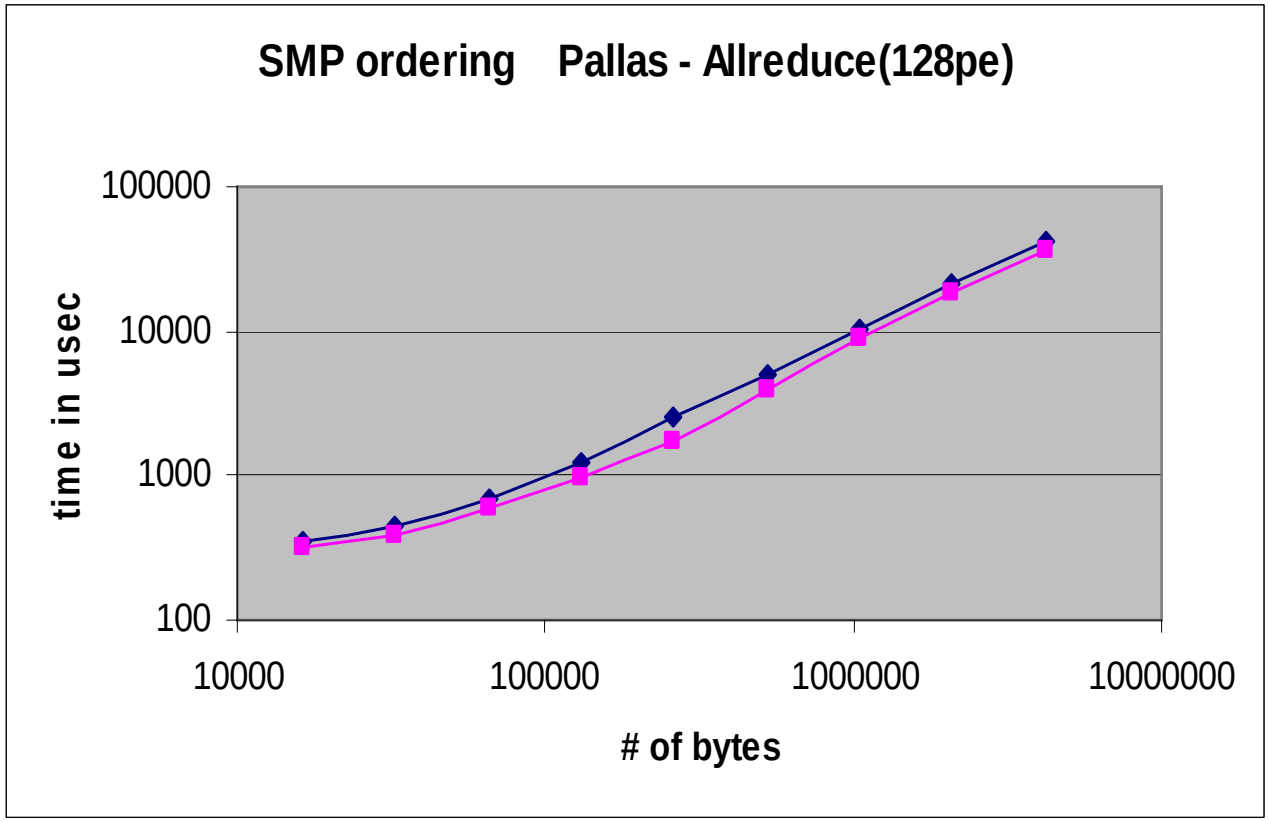
NOTE: Setting PMI_DEBUG will display rank information to stdout

# SMP Rank placement speedups (MPICH_RANK_REORDER_METHOD=1)



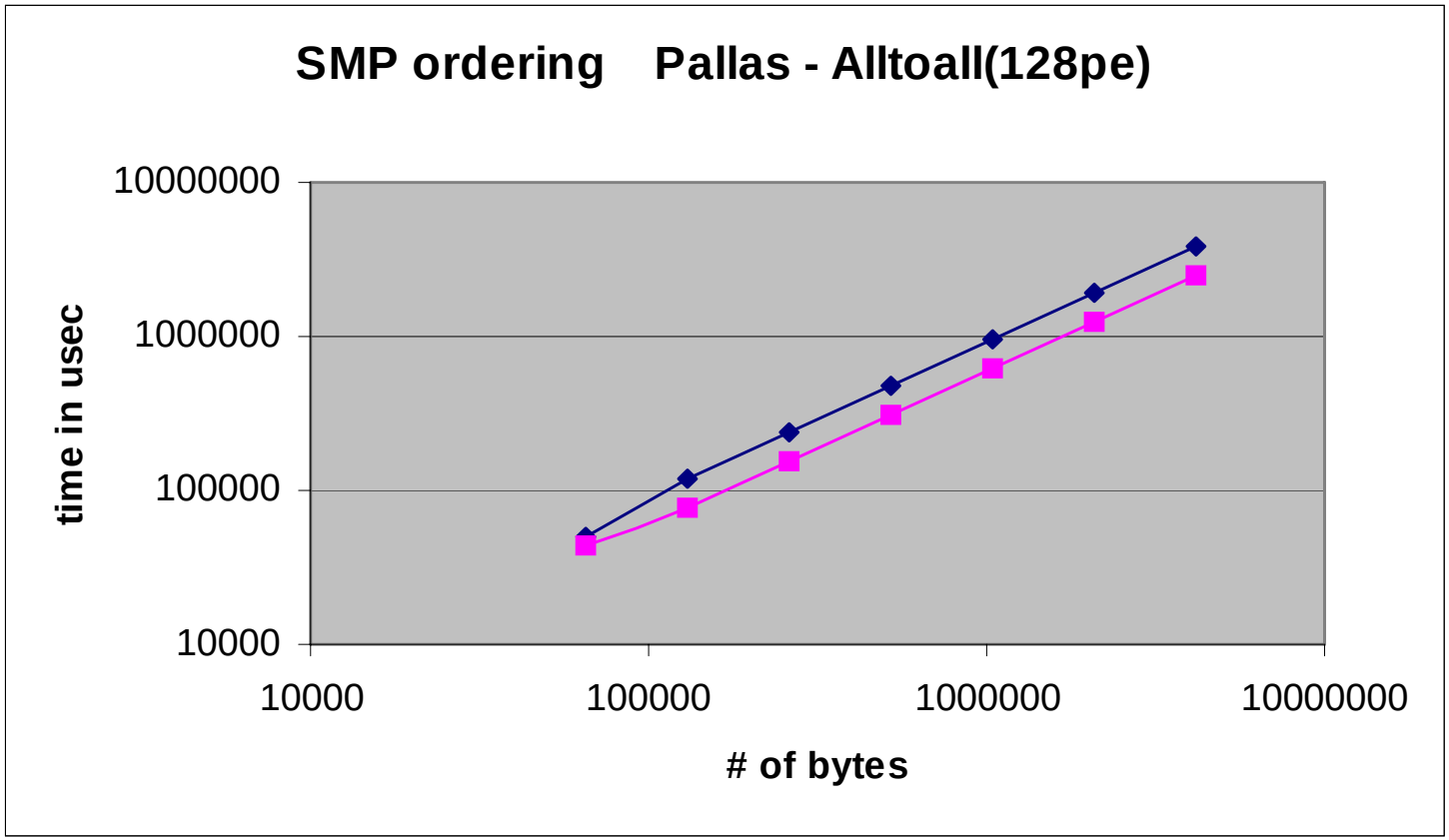

**pt2pt faster by 35% at 8 byte to 60% at 256K bytes**

# SMP Rank placement speedups (MPICH_RANK_REORDER_METHOD=1)



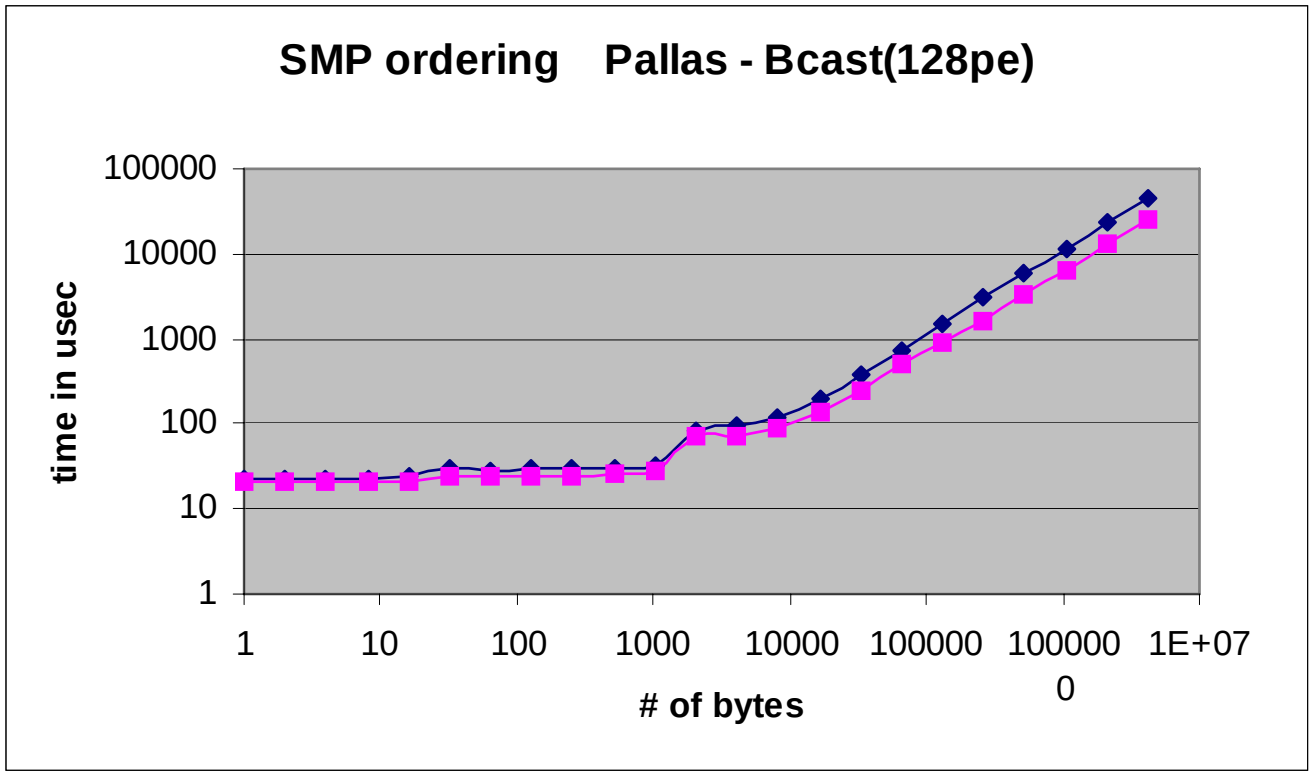

**Allreduce faster by 7% to 32% above 16K bytes**

# SMP Rank placement speedups (MPICH_RANK_REORDER_METHOD=1)




**Alltoall faster by 15% to 36% above 65K message size**

# SMP Rank placement speedups (MPICH_RANK_REORDER_METHOD=1)

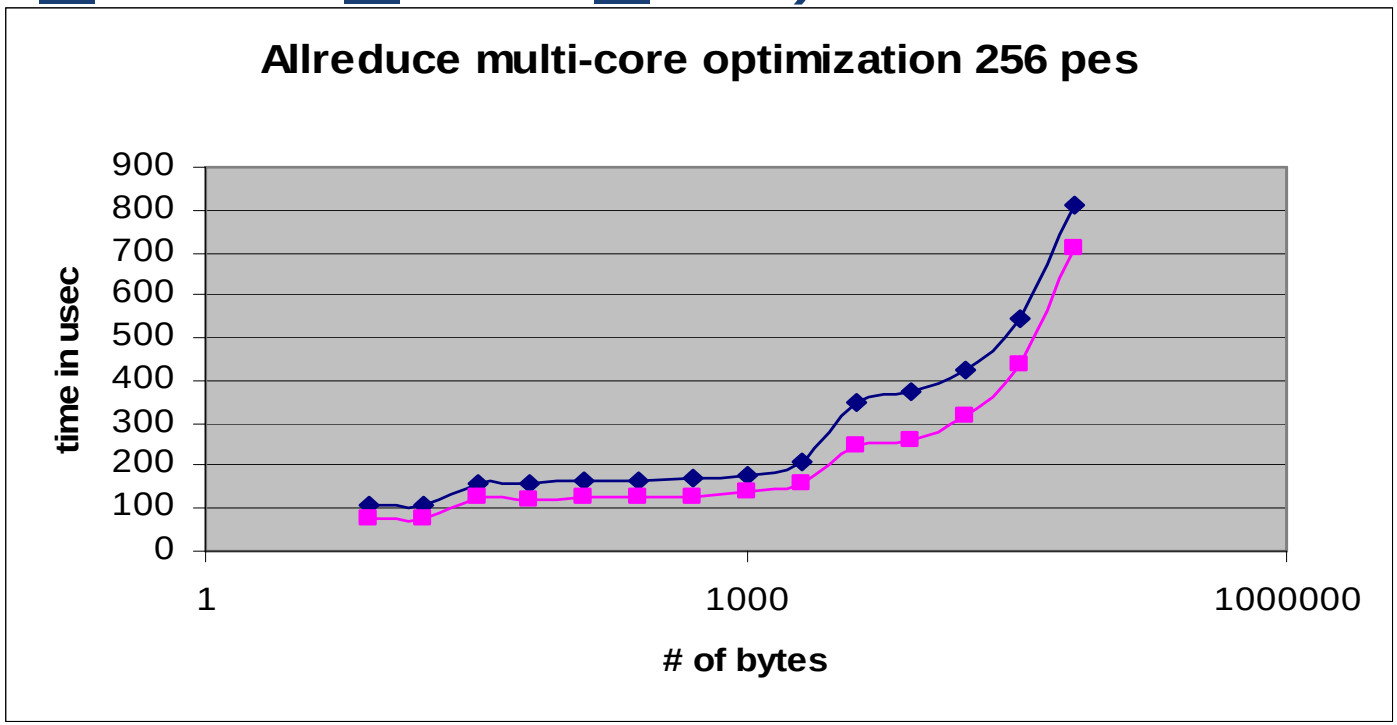



**Bcast faster by 12% at 8 bytes to 45% at 1M bytes**

# New MPI env variables (cont.)

- MPI_COLL_OPT_ON multi-node collective optimizations (In 1.5.11 and 1.4.32)
  - MPI_Allreduce 30% faster for 16K bytes or less (Pallas 256pes)
  - MPI_Barrier - 25% faster (Pallas 256pes)

# Multi-core optimization speedup (MPI_COLL_OPT_ON)




**Allreduce faster by 4% at 1M bytes to 42% at 8 bytes**

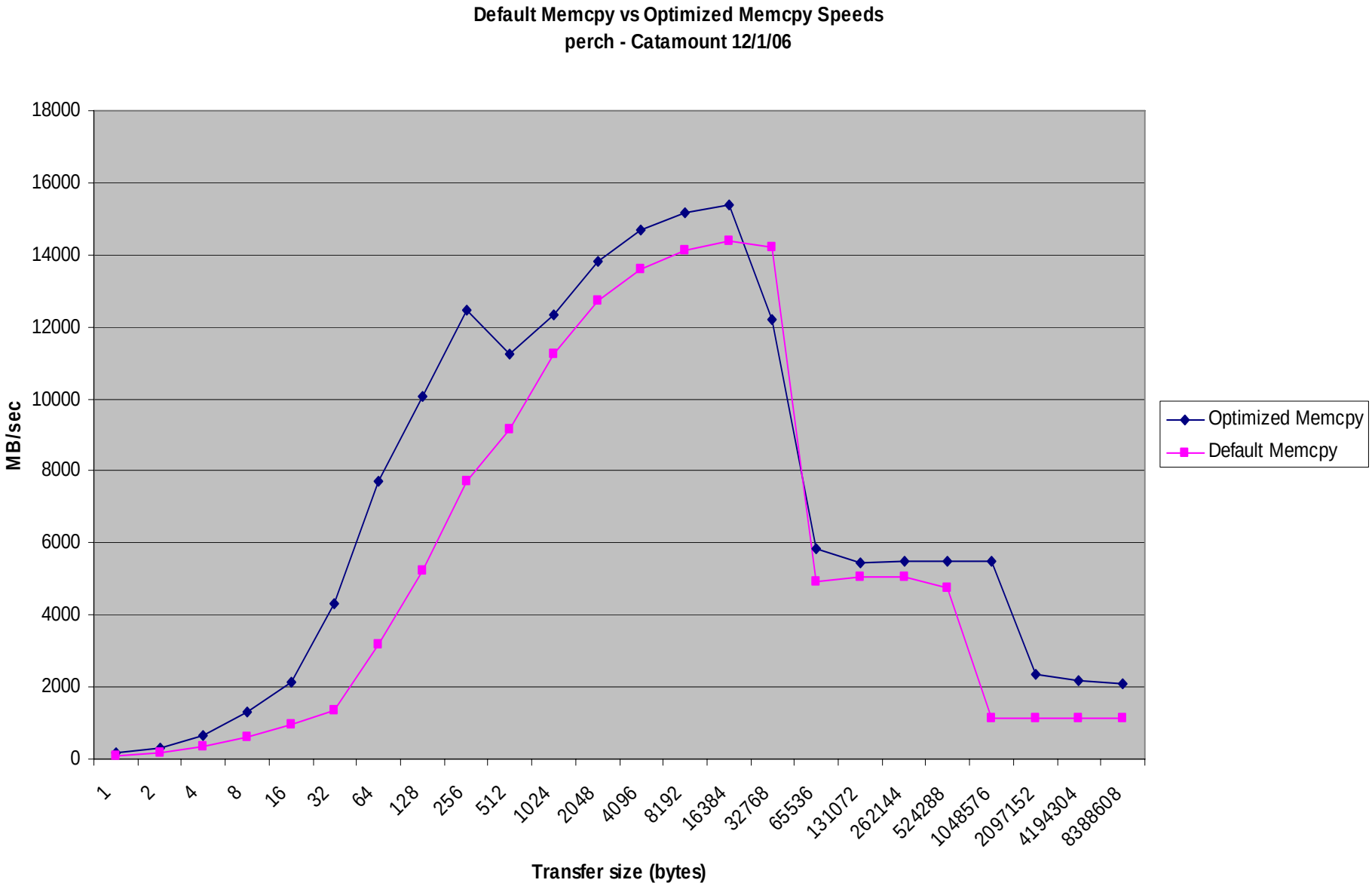
# New MPI env variables (cont.)

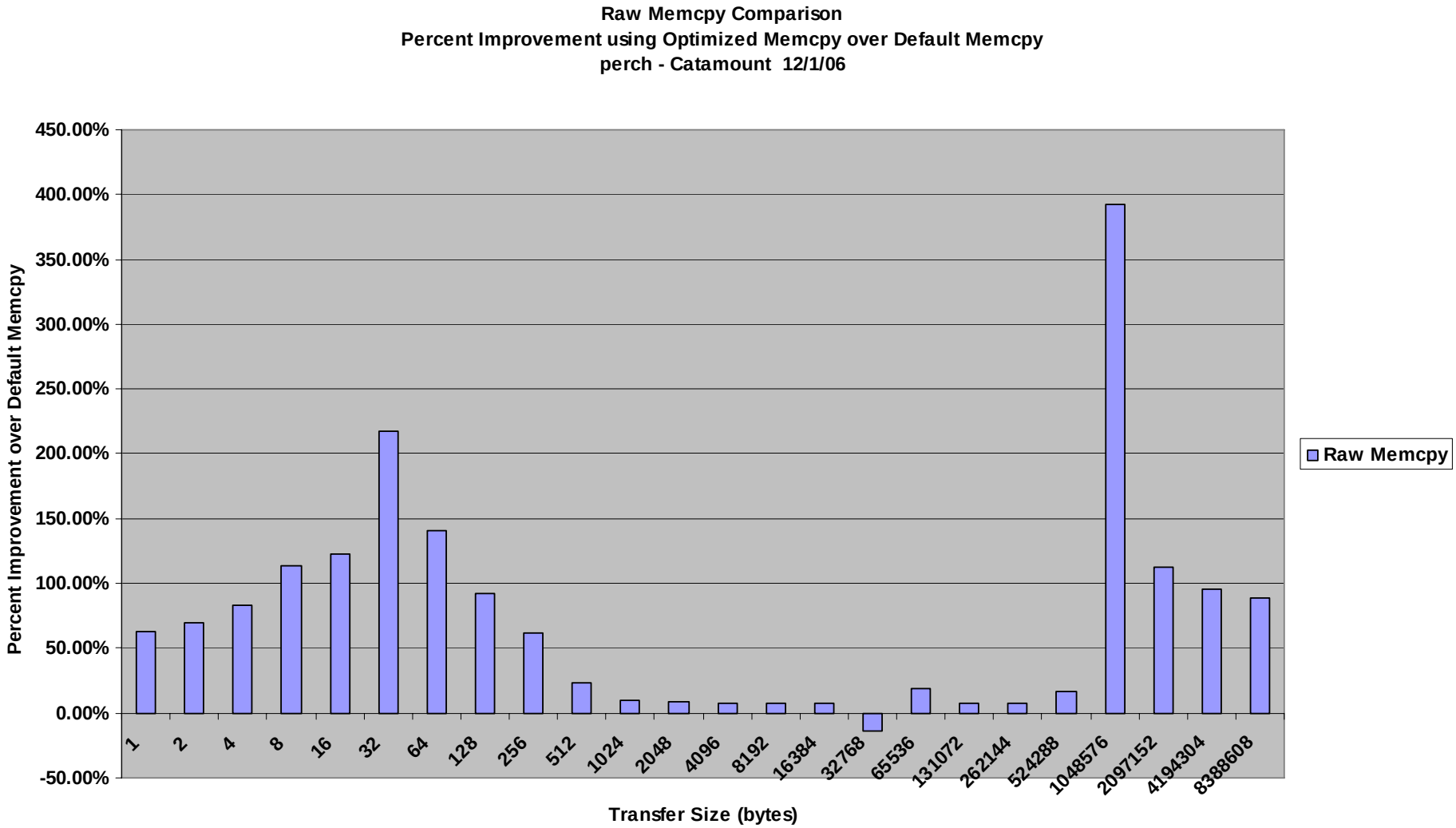
- Added fast memcpy (MPICH_FAST_MEMCPY)

  New improved memcpy used within MPI for local copies for pt2pt and collectives.

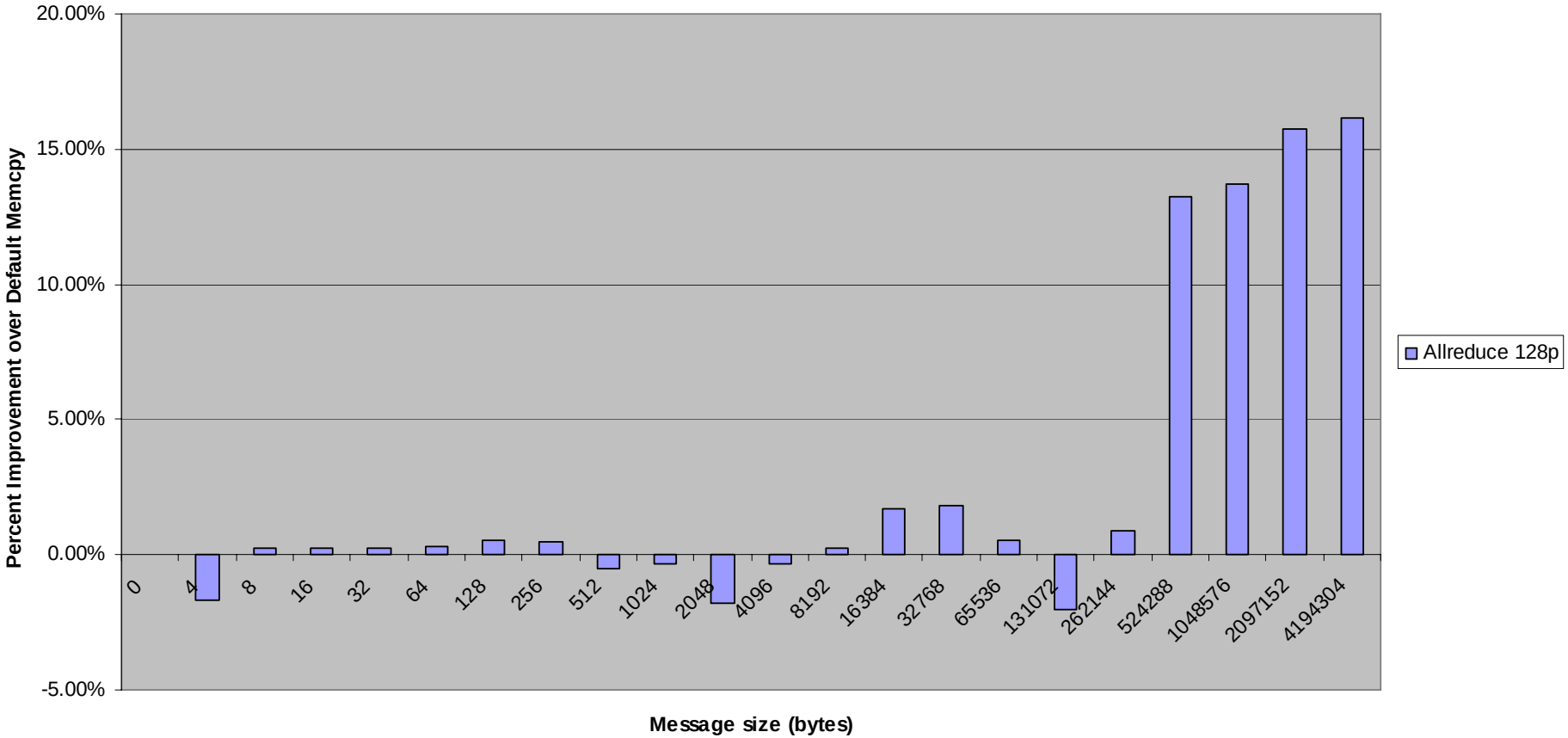
  (In 1.5.30 and 1.4.46)

  - Many collectives 8-20% faster above 256K bytes


Default Memcpy vs Optimized Memcpy Speeds
perch - Catamount 12/1/06
MB/sec
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
8388608
Transfer size (bytes)
Optimized Memcpy
Default Memcpy


Raw Memcpy Comparison
Percent Improvement using Optimized Memcpy over Default Memcpy
perch - Catamount 12/1/06
Percent Improvement over Default Memcpy
450.00%
400.00%
350.00%
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
-50.00%
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
8388608
Transfer Size (bytes)
Raw Memcpy

**Allreduce 128p**
**Percent Improvement using Optimized Memcpy over Default Memcpy**
**perch - Catamount 12/1/06**
**(non-dedicated system)**



# XT Specific MPI documentation

- Man pages
  - intro_mpi
  - yod
- Cray XT Programming Environment User's Guide

# Portals related documentation

- Paper by Brightwell (Sandia), et al. about Portals on XT3 (Red Storm)
  - http://gaston.sandia.gov/cfupload/ccim_pubs_prod/Brightwell_paper.pdf
- Portals project on source forge
  - http://sourceforge.net/projects/sandiaportals/