The IBM High Performance Computing Toolkit

David Klepacki
Advanced Computing Technology
IBM T.J. Watson Research Center

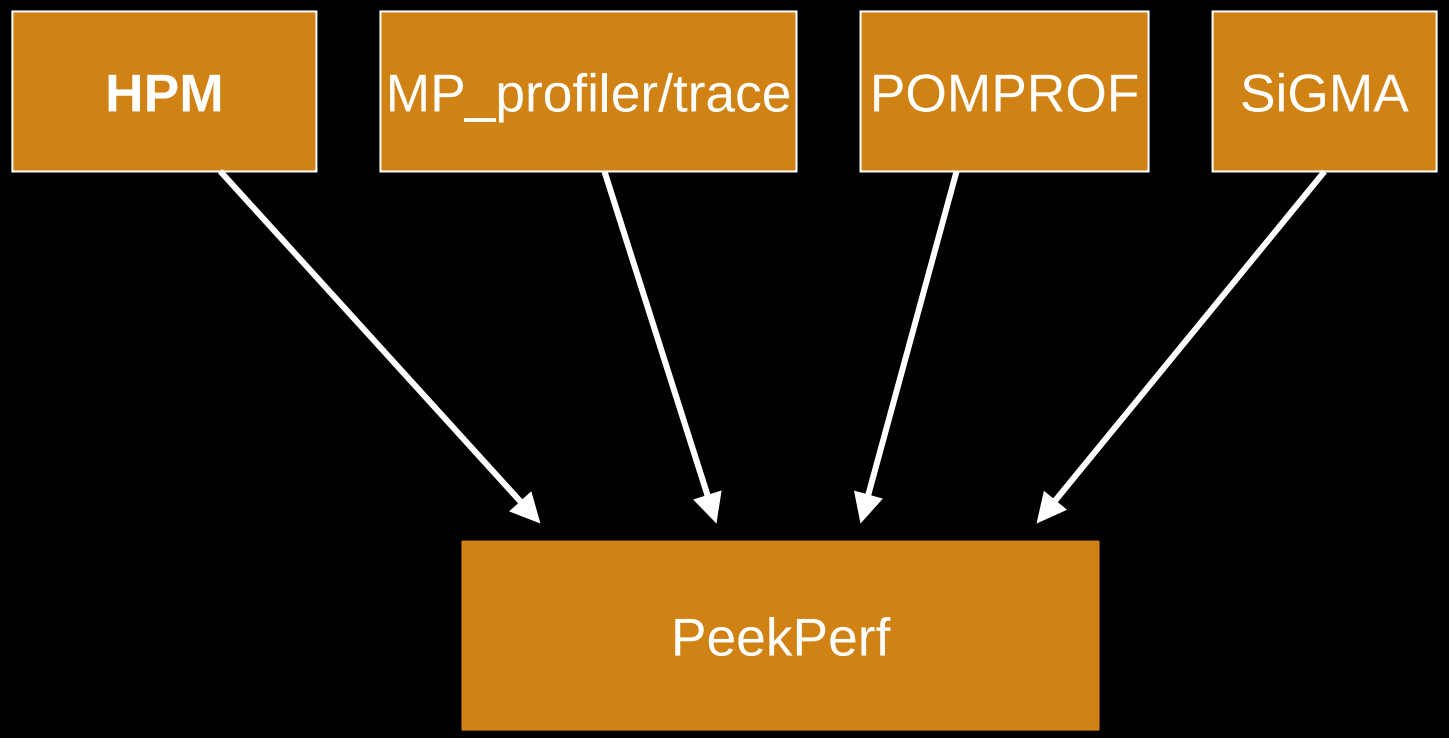
April 2004

IBM High Performance Toolkit Software (IBM pSeries / AIX)

- Hardware Performance
  - HPM Toolkit
    - Catch
    - hpmcount
    - libhpm
    - hpmstat
- Shared Memory Performance
  - DPOMP
    - pomprof
- Message-Passing Performance
  - MP_Profiler
  - MP_Trace
  - TurboMP (SHMEM, MPI2)
- Performance Simulation
  - SiGMA
- Performance Visualization
  - PeekPerf

© 2004 IBM Corporation

Unified GUI with Source Code Traceback



© 2004 IBM Corporation

Typical LIBHPM Usage

- Declaration:

  #include f_hpm.h

- Use:

```
call f_hpminit( 0, “prog” )
call f_hpmstart( 1, “work” )
do
  call do_work()
  call f_hpmstart( 22, “more work” )
        call compute_meaning_of_life()
  call f_hpmstop( 22 )
end do
call f_hpmstop( 1 )
call f_hpmterminate( 0 )
```


© 2004 IBM Corporation

Text LIBHPM Output

```
Instrumented section: 2 - Label: do loop - process: 0
file: peak.f, lines: 71 <--> 82
 Count: 2
 Wall Clock Time: 0.191356 seconds
 Total time in user mode: 0.191085067756377 seconds
 Average duration: 0.095678
 Standard deviation: 0.000235

 PM_FPU_FDIV (FPU executed FDIV instruction)                :              4
 PM_FPU_FMA (FPU executed multiply-add instruction)         :      491532230
 PM_FPU0_FIN (FPU0 produced a result)                       :      245940190
 PM_FPU1_FIN (FPU1 produced a result)                       :      245940117
 PM_CYC (Processor cycles)                                  :      248410547
 PM_FPU_STF (FPU executed store instruction)                :         360000
 PM_INST_CMPL (Instructions completed)                      :      513833515
 PM_LSU_LDF (LSU executed Floating Point load instruction) :         420016

 Utilization rate                          :         99.858 %
 Load and store operations                 :          0.780 M
 Instructions per load/store               :        658.747
 MIPS                                      :       2685.223
 Instructions per cycle                    :          2.068
 HW Float point instructions per Cycle     :          1.980
 Floating point instructions + FMAs (flips) :        983.053 M
 Flip rate (flips / WCT)                   :       5137.297 Mflip/sec
 Flips / user time                         :       5144.581 Mflip/sec
 FMA percentage                            :        100.000 %
 Computation intensity                     :       1260.298
```


© 2004 IBM Corporation

HPM Visualization Using PeekPerf

hpmviz

File

swim_omp

Label	ExcSec	IncSec	Count
Loop 300	4.572	4.572	2398
Loop 200	4.203	4.203	2400
Loop 100	3.071	3.071	2400
Calc3	1.838	6.813	2398
Calc2	1.013	5.632	2400

swim_omp.f | calc1.f | calc2.f | calc3.f

```
     *      VOLD(N1,N2), POLD(N1,N2),
     2      CU(N1,N2), CV(N1,N2),
     *      Z(N1,N2), H(N1,N2), PSI(N1,N2)
C
      COMMON /CONS/ DT,TDT,DX,DY,A,ALPHA,ITMAX,MPRINT,
     1          NP1,EL,PI,TPI,DI,DJ,PCF
      integer ierr
```

Metric Broswser: Loop 300

Close | Metric Options | Precision

Node	Thread	Count	ExcSec	IncSec	U time	Use rate	(M) LS	MIPS	HW FP/Cyc	Instr/LS	M Flips	IpC	Mflip/s	WFlips	Wflip
0	3	2398	4.539	4.539	3.923	86.425	590.056	291.855	0.116	2.245	589.86	0.26	129.947	589.86	129.9
0	0	2398	4.572	4.572	4.378	95.763	608.414	263.277	0.107	1.978	608.234	0.211	133.037	608.234	133.0
0	2	2398	4.549	4.549	4.366	95.979	590.019	255.241	0.104	1.968	589.838	0.205	129.663	589.838	129.6
0	1	2398	4.547	4.547	4.308	94.759	590.024	259.19	0.105	1.997	589.837	0.21	129.728	589.837	129.7
1	2	2398	4.534	4.534	4.398	96.999	590.044	253.123	0.103	1.945	589.856	0.201	130.088	589.856	130.0
1	1	2398	4.528	4.528	3.942	87.069	589.983	286.058	0.115	2.195	589.807	0.253	130.263	589.807	130.2
1	0	2398	4.547	4.547	3.766	82.828	608.434	308.065	0.124	2.302	608.244	0.286	133.762	608.244	133.7
1	3	2398	4.523	4.523	3.537	78.198	589.962	317.346	0.128	2.433	589.781	0.312	130.4	589.781	13
2	0	2398	4.538	4.538	3.777	83.218	608.448	312.01	0.124	2.327	608.262	0.288	134.029	608.262	134.0
2	2	2398	4.522	4.522	4.313	95.364	590.033	257.962	0.105	1.977	589.86	0.208	130.431	589.86	130.4
2	3	2398	4.52	4.52	4.307	95.285	589.985	258.863	0.105	1.983	589.806	0.209	130.492	589.806	130.4
2	1	2398	4.52	4.52	4.35	96.222	589.943	255.814	0.104	1.96	589.767	0.205	130.466	589.767	130.4
3	3	2398	4.487	4.487	4.193	93.453	571.551	259.827	0.105	2.04	571.374	0.214	127.352	571.374	127.3
3	1	2398	4.502	4.502	4.365	96.953	589.937	254.196	0.104	1.94	589.763	0.202	131.003	589.763	131.0
3	2	2398	4.483	4.483	4.139	92.33	571.556	263.864	0.106	2.07	571.38	0.22	127.445	571.38	127.4
3	0	2398	4.506	4.506	3.927	87.154	590.044	290.852	0.116	2.221	589.856	0.257	130.901	589.856	130.9

```
      V(I,J) = VNEW(I,J)
      P(I,J) = PNEW(I,J)
  300 CONTINUE
      call f_hpmtstop( 30+omp_get_thread_num())
```

Metric Option:

- Count
- ExcSec
- IncSec
- PM_FPU_FDIV
- PM_FPU_FMA
- PM_FPU0_FIN
- PM_FPU1_FIN
- PM_CYC
- PM_FPU_STF
- PM_INST_CMPL
- PM_LSU_LDF
- U time
- Use rate
- (M) LS
- MIPS
- HW FP/Cyc
- Instr/LS
- M Flips
- IpC
- Mflip/s
- WFlips
- Wflip/s
- FMA %
- Comp Int.

© 2004 IBM Corporation

MP_Profiler and MP_Trace Libraries

- MP_Profiler
  - Captures “summary” data for MPI and SHMEM calls with source code traceback
  - No changes to source code, but MUST compile with -g
  - ~1.7 microsecond overhead per MPI / SHMEM call
  - Several libraries needed:
    - mpiprof, smaprof, turbo1prof, turbo2prof
- MP_Trace
  - Captures “timestamped” data for MPI and SHMEM calls with source traceback
  - ~1.4 microsecond overhead per MPI / SHMEM call
  - More libraries needed:
    - mpitrace, smatrace, turbo1trace, turbo2trace

© 2004 IBM Corporation

MP_Profiler Sample Text Output

```
-----------------------------------------------------------
MPI Routine            #calls     avg. bytes      time(sec)
-----------------------------------------------------------
MPI_Comm_size               3            0.0          0.000
MPI_Comm_rank           12994            0.0          0.016
MPI_Send                19575        11166.9         13.490
MPI_Isend              910791         5804.2          9.216
MPI_Recv               138173         2767.9         73.835
MPI_Irecv              784936        15891.6          2.407
MPI_Sendrecv           894809          352.0         88.705
MPI_Wait              1537375            0.0        288.049
MPI_Waitall             44042            0.0         25.312
MPI_Bcast                 464        41936.8          3.272
MPI_Barrier              1312            0.0         34.206
MPI_Gather                 68        16399.1          2.680
MPI_Scatter                 6        17237.3          0.532
-----------------------------------------------------------
total communication time = 770.424 seconds.
total elapsed time       = 1168.662 seconds.
user cpu time            = 1160.960 seconds.
system time              = 0.620 seconds.
maximum memory size      = 68364 KBytes.

To check load balance :  grep "total comm" mpi_profile.*
```


© 2004 IBM Corporation

MP_Profiler Sample Call Graph Output

```
communication time = 143.940 sec, parent = gwwrloc
   MPI Routine          #calls        time(sec)
   MPI_Barrier            2311          143.734
   MPI_Gatherv            2311            0.206
communication time = 137.823 sec, parent = f2drecv
   MPI Routine          #calls        time(sec)
   MPI_Recv              91959          137.823
communication time = 108.960 sec, parent = puttsf
   MPI Routine          #calls        time(sec)
   MPI_Barrier           23607          106.821
   MPI_Gatherv           23607            2.139
communication time = 94.435 sec, parent = fft_tri_recv
   MPI Routine          #calls        time(sec)
   MPI_Recv               6378           94.435
communication time = 83.836 sec, parent = fft2drecv
   MPI Routine          #calls        time(sec)
   MPI_Recv              93003           83.836
```


© 2004 IBM Corporation

MP_Profiler Visualization Using PeekPerf

PeekPerf Main Window

File Tools Options

GAMESS | MPI_Application

Label	Call Count [Ma
MPI_Allreduce_807	16
MPI_Allreduce_868	38
MPI_Barrier_1496	1
MPI_Barrier_248	1
MPI_Barrier_765	4
MPI_Bcast_924	285
MPI_Comm_rank_973	5
MPI_Comm_size_972	5
MPI_Iprobe_1160	
MPI_Recv_1027	
MPI_Recv_1135	
MPI_Ssend_1002	
Summary_MPI_All	
Summary_MPI_Bar	
Summary_MPI_Bcas	
Summary_MPI_Comm	
Summary_MPI_Comm	
Summary_MPI_Iprc	
Summary_MPI_Recv	
Summary_MPI_Sser	

gamess.f | ddi.f

```
C  PROCESSES ONLY, NOT THE TOTAL NUMBER OF
PROCESSES.
C
-----------------------------------------------
C
      IMPLICIT NONE
      INTEGER DDI_NP, DDI_ME
C
      INCLUDE 'mpif.h'
```

Metric Broswser: MPI_Bcast_924

Close | Metric Options | Precision

Task	Message Size	WallClock	Count	Call Count [Max]	WallClock [Max]	Transfered Bytes
0	(2) 5 ... 16	0.049632	133	133	0.049632	1064
0	(3) 17 ... 64	0.000144	2	2	0.000144	64
0	(4) 65 ... 256	0.001124	16	16	0.001124	1408
0	(5) 257 ... 1K	0.030647	163	163	0.030647	47408
0	(6) 1K ... 4K	0.011084	134	134	0.011084	198472
0	(7) 4K ... 16K	0.024244	285	285	0.024244	1.69084e+06
0	(8) 16K ... 64K	0.001328	16	16	0.001328	227200
0	(9) 64K ... 256K	0.078051	121	121	0.078051	1.09938e+07
1	(2) 5 ... 16	0.185036	133	133	0.185036	1064
1	(3) 17 ... 64	0.000183	2	2	0.000183	64
1	(4) 65 ... 256	0.009773	16	16	0.009773	1408
1	(5) 257 ... 1K	0.018897	163	163	0.018897	47408
1	(6) 1K ... 4K	0.019423	134	134	0.019423	198472
1	(7) 4K ... 16K	0.251916	285	285	0.251916	1.69084e+06
1	(8) 16K ... 64K	0.406296	16	16	0.406296	227200
1	(9) 64K ... 256K	0.087307	121	121	0.087307	1.09938e+07

© 2004 IBM Corporation

MP_Profiler and HPM Integration

PeekPerf Main Window

File Tools Options

GAMESS | MPI_Application

Label	Count	ExcSec	IncSe
BRNCHX	1	62.664	62.66
Gamess Main	1	0.037	62.71
LMOINP	1	0	0
LMOX	1	0	0
TRUNC	1	0.013	0.013
VALFM (END)	1	0	0
VALFM (INIT)	1	0	0

gamess.f | ddi.f

```
        call f_hpmstop( 13)
      ENDIF
      IF(RPAC    .AND.  EXETYP.NE.CHECK) THEN
        call f_hpmstart( 14, "RPACX")
        CALL RPACX
        call f_hpmstop( 14)
      ENDIF
      IF(FRIEND.NE.BLANK) CALL ZEALX
C
C        ----- ALL DONE -----
C
      call f_hpmstart( 9, "VALFM (END)")
      CALL VALFM(LASTFM)
      call f_hpmstop( 9)
      IF(LASTFM.NE.INITFM  .AND.  MASWRK) WRITE(IW,
```

Metric Broswser: VALFM (END)

Close | Metric Options | Precision

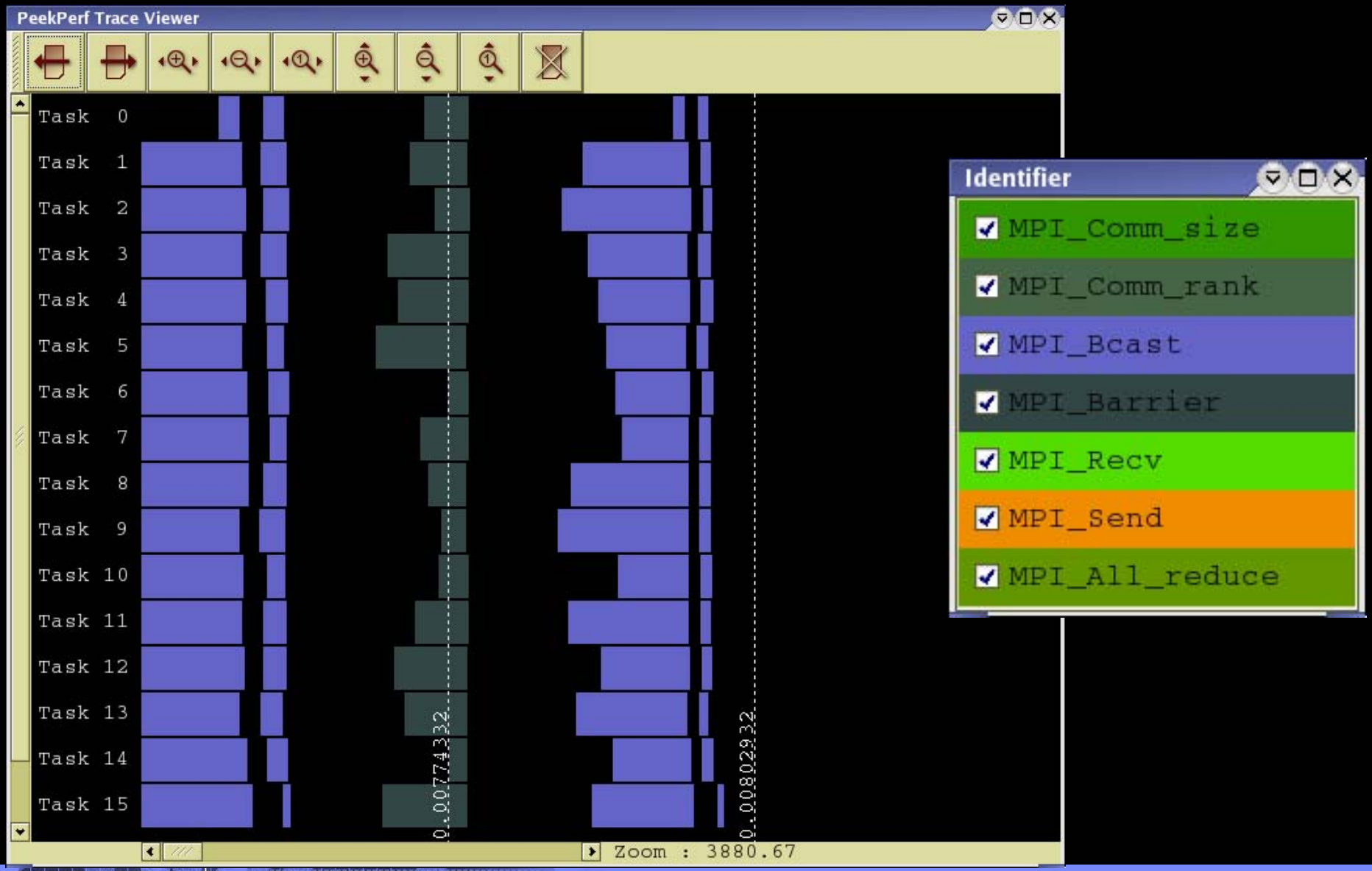
Task	thread	Count	Use rt	(M)LS	I/LS	MIPS	IpC	MFlips	Mflip/s	MFlip/Us	FMA %
0	0	1	4.909	0	0	2.471	0.035	0	0	0	0
1	0	1	1.403	0	0	2.8	0.137	0	0	0	0

```
,3A8)
ATED NORMAI
EW INPUT CA

2 DEC 2003
NIVERSITY'
```


© 2004 IBM Corporation

MP_Trace Visualization Using PeekPerf



© 2004 IBM Corporation

TurboMP Libraries (latest version: 3.0.2)

- TurboMPI-1 Libraries
  - Collective Communications Enhanced for shared memory
    - AllReduce, Alltoall, Alltoallv, Bcast, Barrier, Reduce
  - No LAPI dependency (to accommodate pre-Federation switched systems)
  - 32-bit and 64-bit support, Fortran/C/C++
  - Syntax: -lturbo1
- TurboMPI-2 Libraries
  - All of MPI-1 above, plus MPI-2 RMA operations
    - Put, Get, Accumulate, Fence, Lock/Unlock, Start/Post/Wait/Complete, Test
  - Syntax: -lturbo2
- SHMEM Libraries
  - “Complete” implementation (400+ routines)

© 2004 IBM Corporation

SHMEM Implementation Status

- Fortran77, Fortran90/95, and C/C++
- Threadsafe libraries
- Check/Start Safety under study
- Uses both LAPI and MPI for best performance
- TurboMPI only supports **contiguous datatypes**

  No user-defined datatypes or reduction operations
- SHMEM is “complete” except for:

  shmem_ptr() , and,

  Functions which don’t make sense on IBM:

  shmem_stack(), shmem_cache functions, etc

  -- Implemented as no-ops

© 2004 IBM Corporation

SHMEM Profiling Capability

PeekPerf Main Window

File Tools Options

SHMEM_Application

Label	Call Count
my_pe_945	2
num_pes_944	2
shmem_barrier_all_788	431
shmem_barrier_all_799	431
shmem_barrier_all_859	186
shmem_barrier_all_871	186
shmem_barrier_all_915	114
shmem_barrier_all_927	114
shmem_broadcast8_796	273
shmem_int8_sum_to_all_923	114

ddishm.f

```
      WPE_START    = PE_START
      WLOG_STRIDE = LOG_STRIDE
      WPE_SIZE     = PE_SIZE
C
      CALL SHMEM_BROADCAST8( TARGET, SOURCE, WLENGTH
     *                       WPE_START, WLOG_STRIDE, W
      CALL SHMEM_QUIET()
      CALL SHMEM_BARRIER_ALL()
      IF ( MYNODE .NE. PE_ROOT )
     *  CALL ICOPY( LENGTH, TARGET, 1, BUFF(LOCBUF), 1
      LOCBUF = LOCBUF + LENGTH
```

Metric Broswser: shmem_broadcast8_796

Close | Metric Options | Precision

Task	Message Size	Call Count [Max]	WallClock [Max]	Transfered Bytes	Count	WallClock
0	(2) 5 ... 16	49	0.040212	392	49	0.040212
0	(3) 17 ... 64	7	0.002289	168	7	0.002289
0	(4) 65 ... 256	48	0.015485	3192	48	0.015485
0	(5) 257 ... 1K	273	0.105899	76440	273	0.105899
0	(6) 1K ... 4K	29	0.019601	20888	29	0.019601
0	(7) 4K ... 16K	25	0.008123	162816	25	0.008123
1	(2) 5 ... 16	49	0.020304	392	49	0.020304
1	(3) 17 ... 64	7	0.003646	168	7	0.003646
1	(4) 65 ... 256	48	0.029487	3192	48	0.029487
1	(5) 257 ... 1K	273	0.142042	76440	273	0.142042
1	(6) 1K ... 4K	29	0.184179	20888	29	0.184179
1	(7) 4K ... 16K	25	0.015682	162816	25	0.015682

© 2004 IBM Corporation

SHMEM Notes

- Needed to add 4 “new” functions
  shmem_init(), shmem_finalize()
  shpalloc_init() and shmalloc_init()

- lxput/lxget functions modified to accommodate both 32-bit and 64-bit index vectors (see README)
  -- User needs to ensure 32-bit integer parameters in 64-bit mode

© 2004 IBM Corporation

TurboSHMEM Sample Usage (F77)

```
 program sample_f77
 implicit none
 include "shmem.fh"
C
 real*8 ref,source, target
 integer ierr,pe,mype,npes,
    psync(SHMEM_BARRIER_SYNC_SIZE)
 integer foo(256)
 save foo
C
 dimension target(*), source(*)
 pointer (t_p,target),(s_p,source)
C
 call shmem_init()
 call shpalloc_init(foo,1024)
 call shpalloc(t_p, 2, ierr, 0)
 call shpalloc(s_p, 2, ierr, 0)
```

```
 target(1)= 99.d0
 mype = shmem_my_pe()
 npes = shmem_n_pes()
C
 source(1) = mype+1
 ref = mod((mype+npes-1),npes)+1
 pe = mod((mype + 1), npes)
C
 call shmem_put(target,source,1,pe)
 call shmem_barrier(0,0,npes,pSync)
C
 call shpdeallc(s_p, ierr, 0)
 call shpdeallc(t_p, ierr, 0)
 call shmem_finalize()
C
 stop
 end
```


© 2004 IBM Corporation

Sample Usage

- Compile:

  mpxlf_r sample.f –lsma

  mpxlf90_r –q64 sample.f –lsma

- Run: (e.g., POWER4, shared memory only)

  MP_PROCS=16
  MP_MSG_API=mpi,lapi
  MP_SHARED_MEMORY=yes
  MP_EUILIB=ip
  MP_RESD=no
  LAPI_USE_SHM=only
  poe a.out

© 2004 IBM Corporation

Symmetric Storage

- Fortran:

  Common, Save, or use module
- C/C++

  Static Declaration
- AIX unable to “shmalloc”

  Work-around using static storage

  Full semantics of shmalloc() / shpalloc() implemented using user-defined static storage defined via above methods.

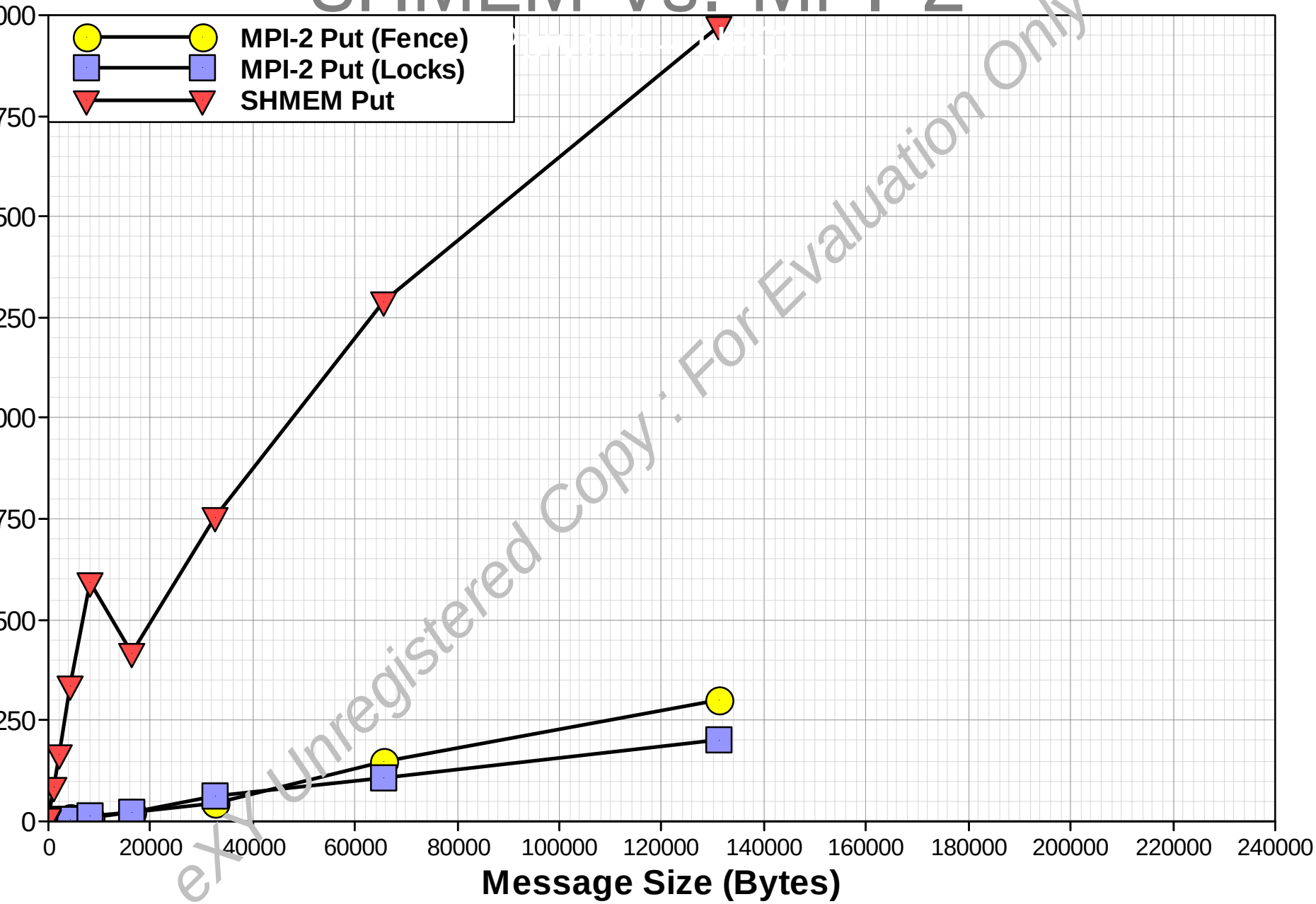
© 2004 IBM Corporation

TurboSHMEM Dynamic Dimensioning (-qddim)

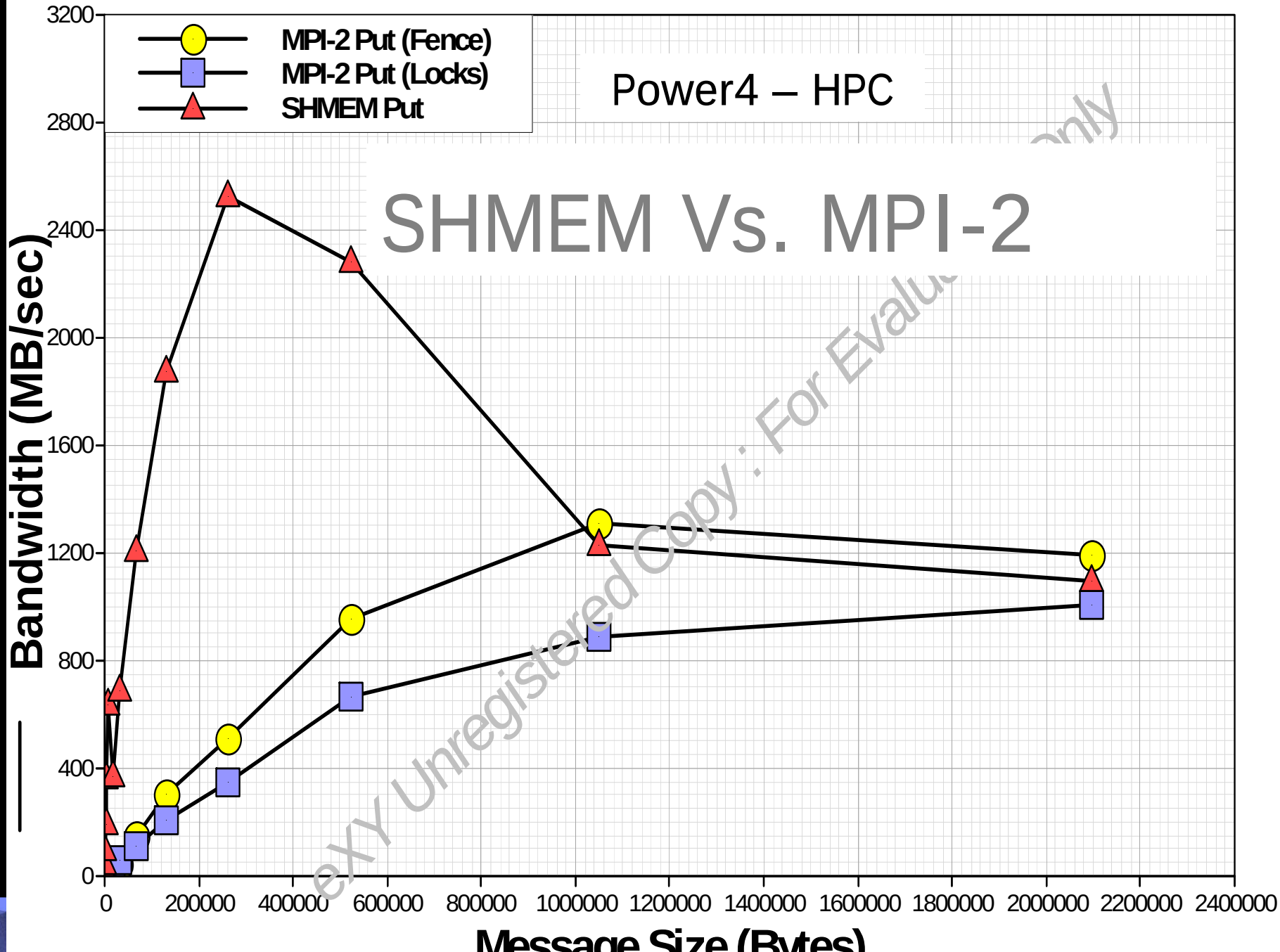
```
  integer,save,dimension(256) :: foo
  integer, parameter :: px=4, py=4, xglobal=500, yglobal=500
  integer :: P, T, xlocal, ylocal, i, j, ierr
  real, dimension(0:xlocal,0:ylocal) :: u,v
  pointer (u_ptr,u),(v_ptr,v)
!
  call shmem_init()
  call shpalloc_init(foo,256*4)
!
! Klepacki Support Routines for Data Parallel: create 125x125 subarray
  call HPF_Processors_2D(P,px,FALSE,py,FALSE)
  call HPF_Template_2D(T,P,xglobal,BLOCK,yglobal,BLOCK,xlocal,ylocal)
  call shpalloc(u_ptr, xlocal*ylocal, ierr, 0)
  call shpalloc(v_ptr, xlocal*ylocal, ierr, 0)
!
  forall (i=0:xlocal, j=0:ylocal)
     u(i,j) = …
     v(i,j) = …
  end forall
     …
```


© 2004 IBM Corporation

SHMEM Vs. MPI-2

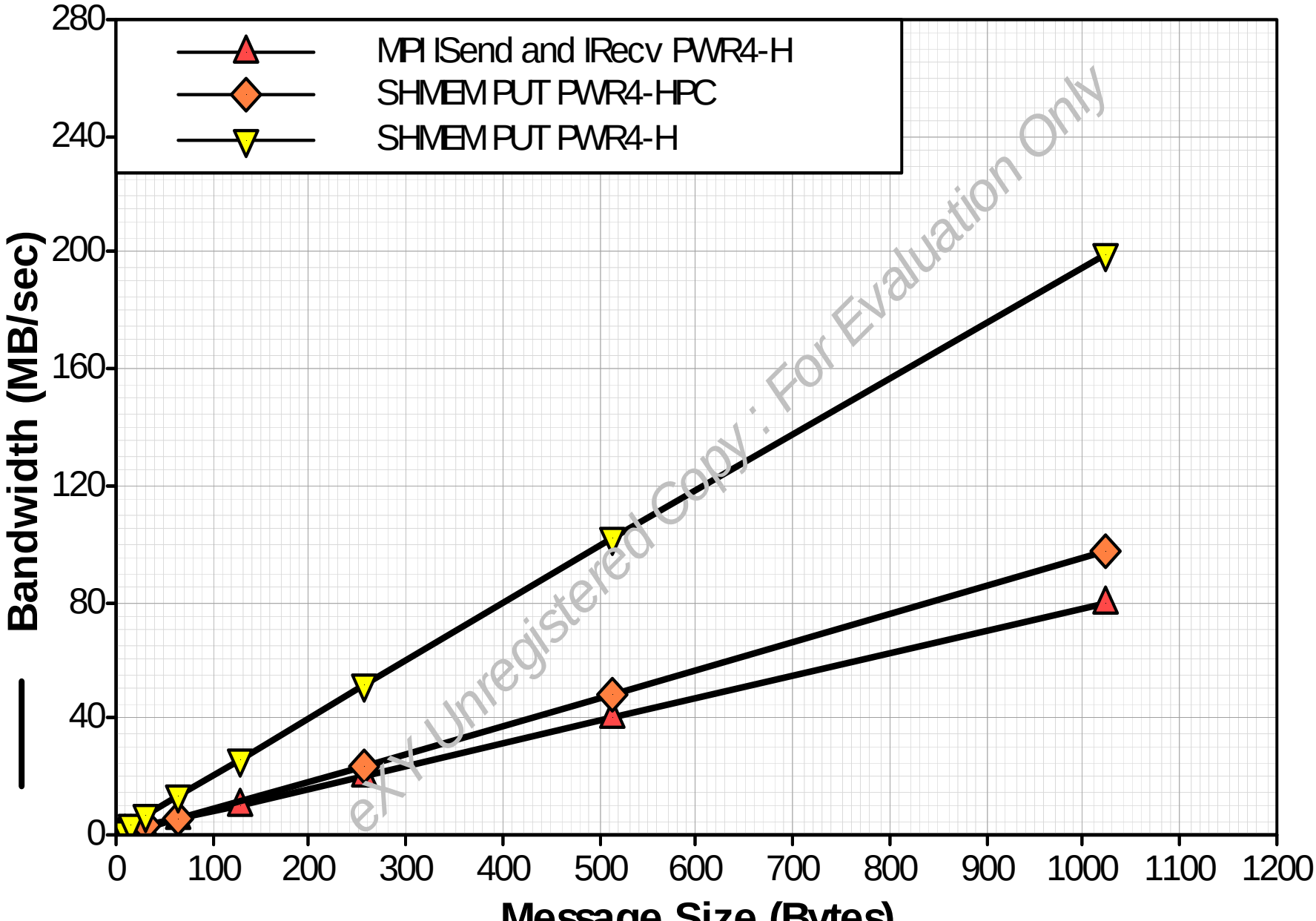


eXY Unregistered Copy : For Evaluation Only



eXY Unregistered Copy : For Evaluation Only

SHMEM Vs. MPI-1



eXY Unregistered Copy : For Evaluation Only

SHMEM Problem

```
do i = 1, nbprocs
   index = (rank * kval) + 1
   call shmem_ixget(temp, fp(index,1,1), tab,  nys*kval*kval, i-1)
   L = 1
   index = ( ( i – 1 ) * kval ) + 1
   do k = index, index + kval - 1
       do j = 1, nys
          do k1 = 1, kval
               fpp(k1, j, k) = temp(L)
                 L = L + 1
          enddo
        enddo
   enddo
enddo
```

Linear in procs

Index vector

VLASOV Hydrodynamics

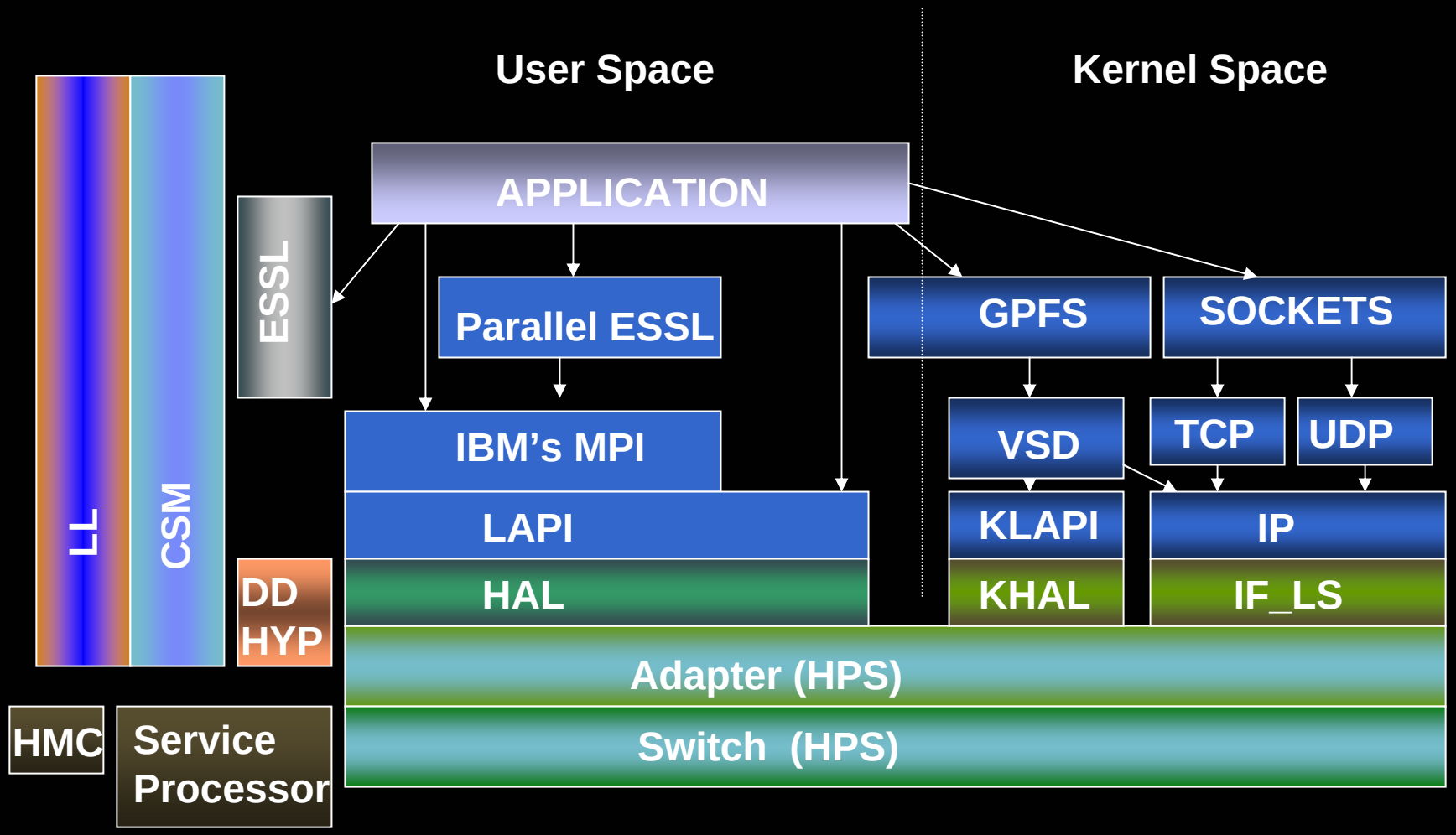
© 2004 IBM Corporation

Solution: New TurboSHMEM Function

- SHMEM_Alltoall()
  - Log_2 performance with procs
  - One-line of code replacement for previous loop!
  - “Turbo” performance: -lsma –lturbo1

© 2004 IBM Corporation

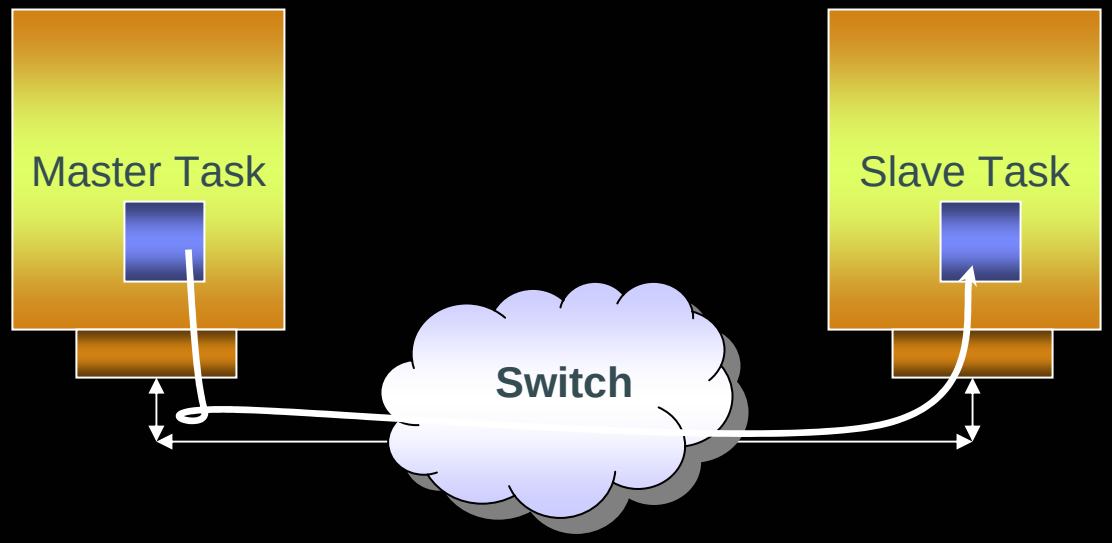
Communication Software Architecture



© 2004 IBM Corporation

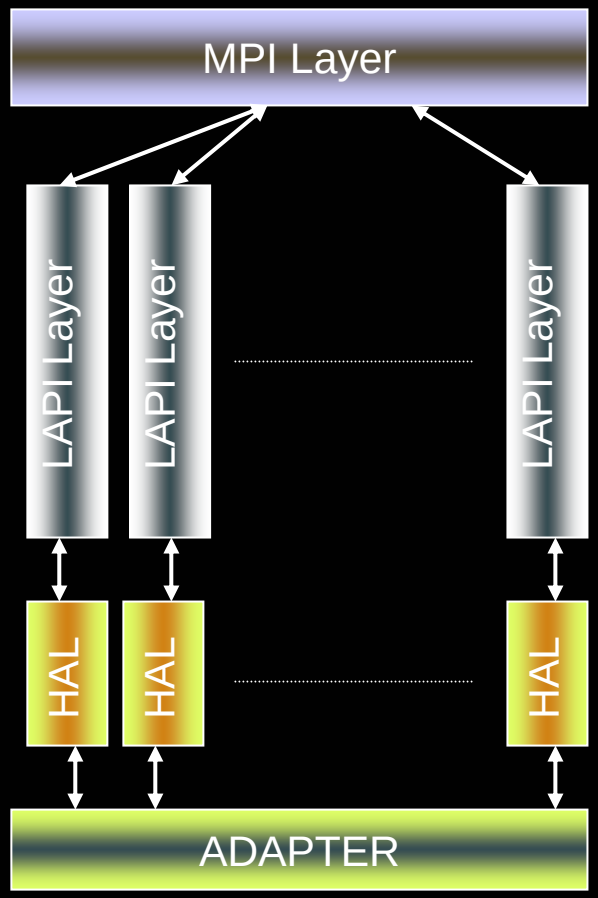
What is RDMA?

- Remote Direct Memory Access
- Memory Semantics for Remote Access
- No slave side protocol processing
  - No interim packet arrival interrupts
  - Assumes pinned/mapped transfers

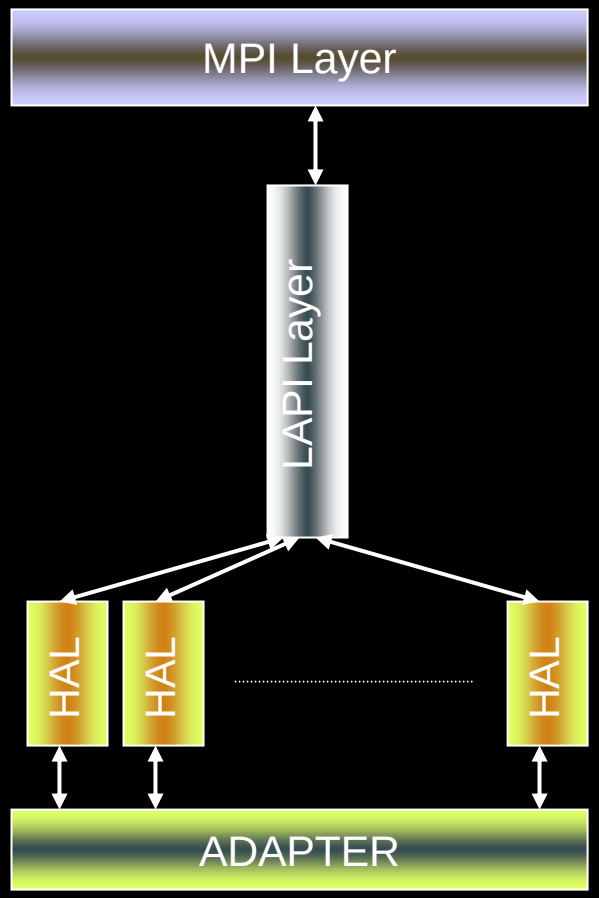


© 2004 IBM Corporation

Striping Implementation Models



Multiple threads doing copies model



Single Thread with Pipelined RDMA model

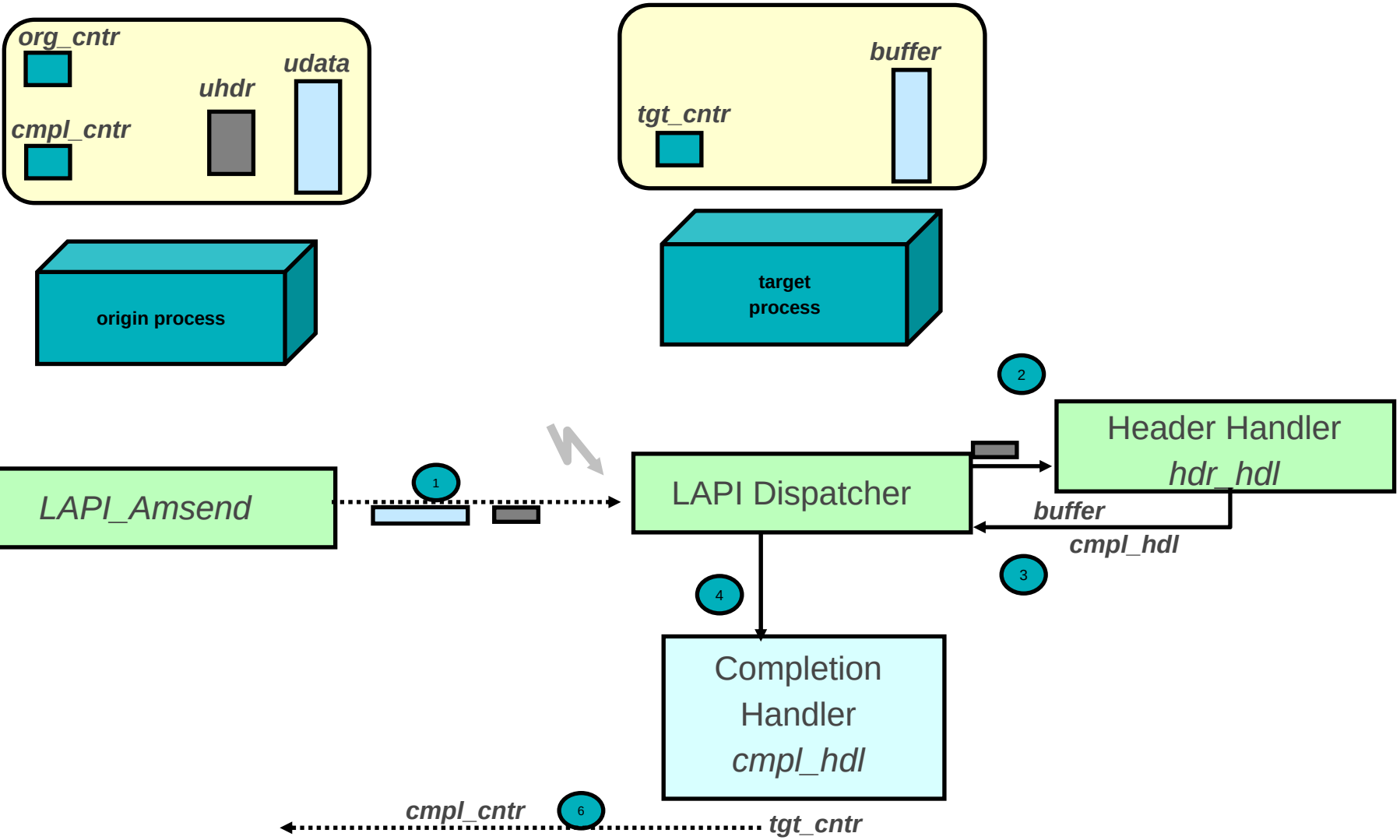
© 2004 IBM Corporation

Messaging Foundation: IBM LAPI

- One sided communication programming model
- Simulate synchronous and standard communication behavior of MPI
- Completion of non-blocking communication signaled at both ends.
  - origin counter
  - target counter
  - completion counter
- Reliable
- Enhanced Active Message Interface
- Low overhead interface designed to minimize copies
- No ordering of messages
- C , C++ and Fortran interfaces
- IBM designed interface (not a standard)

© 2004 IBM Corporation

LAPI_Amend (hndl, tgt, *hdr_hndler, *uhdr, uhd_len, *udata, udata_len, *tgt_cntr, *org_cntr, *cmpl_cntr).



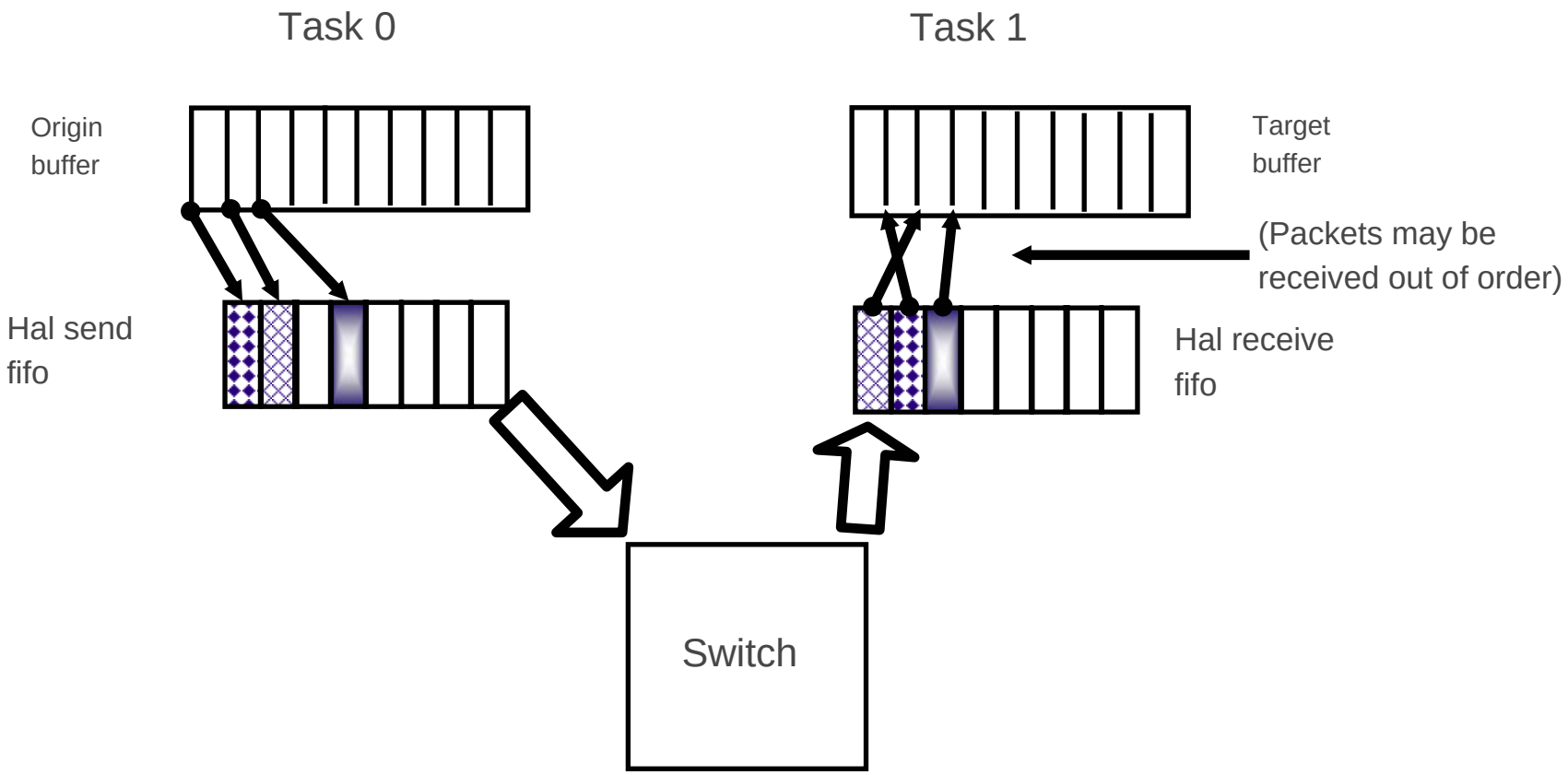
© 2004 IBM Corporation

LAPI Internode Communication

- With switch only
  - Copy to send fifo
  - DMA to adapter memory
  - DMA from adapter memory
  - Copy from receive fifo

© 2004 IBM Corporation

LAPI Put Example



© 2004 IBM Corporation

LAPI Intranode Communication

- With Shared memory segment
  - Copy data to shared segment
  - Use semaphore to signal other task
  - Copy data from shared segment
- Two copies in total
- Can we eliminate one copy?

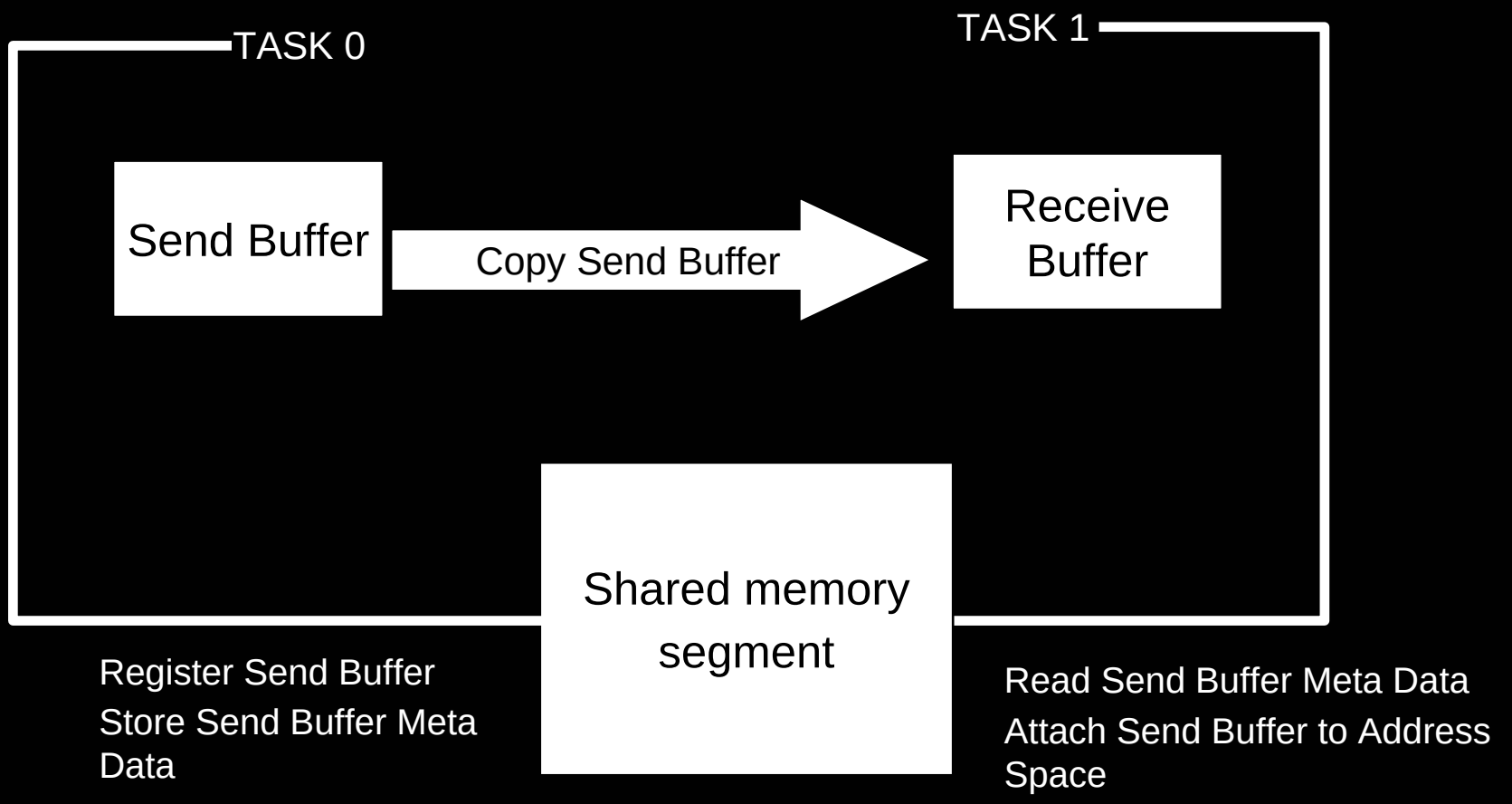
© 2004 IBM Corporation

LAPI Kernel Extension

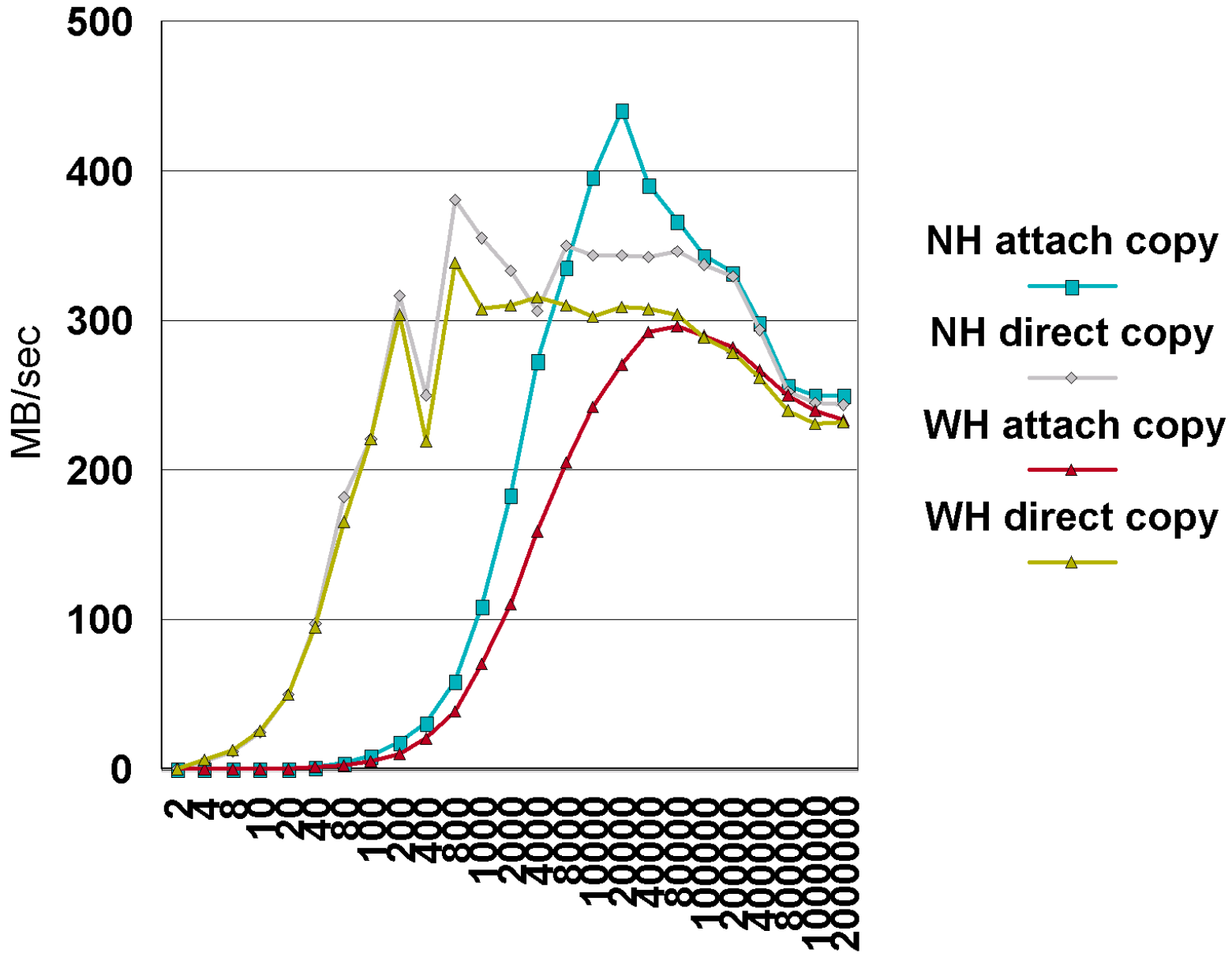
- Task 1 registers segment containing data--gets handle back
- Task 1 copies handle to shared segment
- Task 1 wakes up task 2
- Task 2 retrieves handle
- Task 2 attaches segment to it's address space
- Task 2 copies data
- One copy only.

© 2004 IBM Corporation

LAPI Shared Memory Put Example

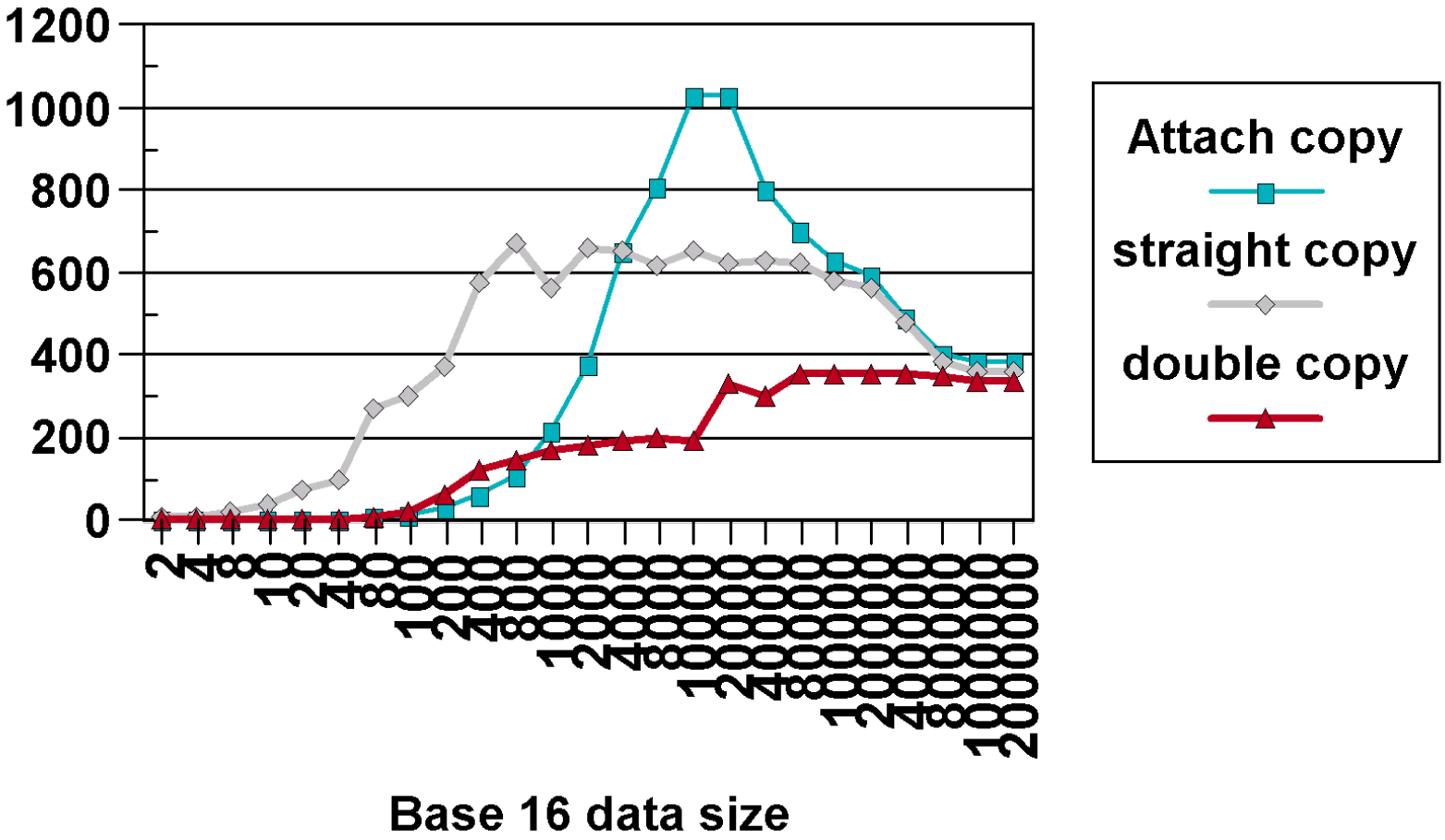


© 2004 IBM Corporation



© 2004 IBM Corporation

Performance of shared memory copy



attach/detach time 16 usec
spin lock exchange time 2 usec

© 2004 IBM Corporation

Performance Gains with LAPI shared Memory

NWCHEM bechmarks (8 way)

Input deck	Shared	Switch
BR2_DK	195.0	309.0
MD_HAD	8593.8	8792.6
PSPW	61.7	63.2
U02_DFT	232.4	709.1
DFT_SiOSI3	185.7	285.4

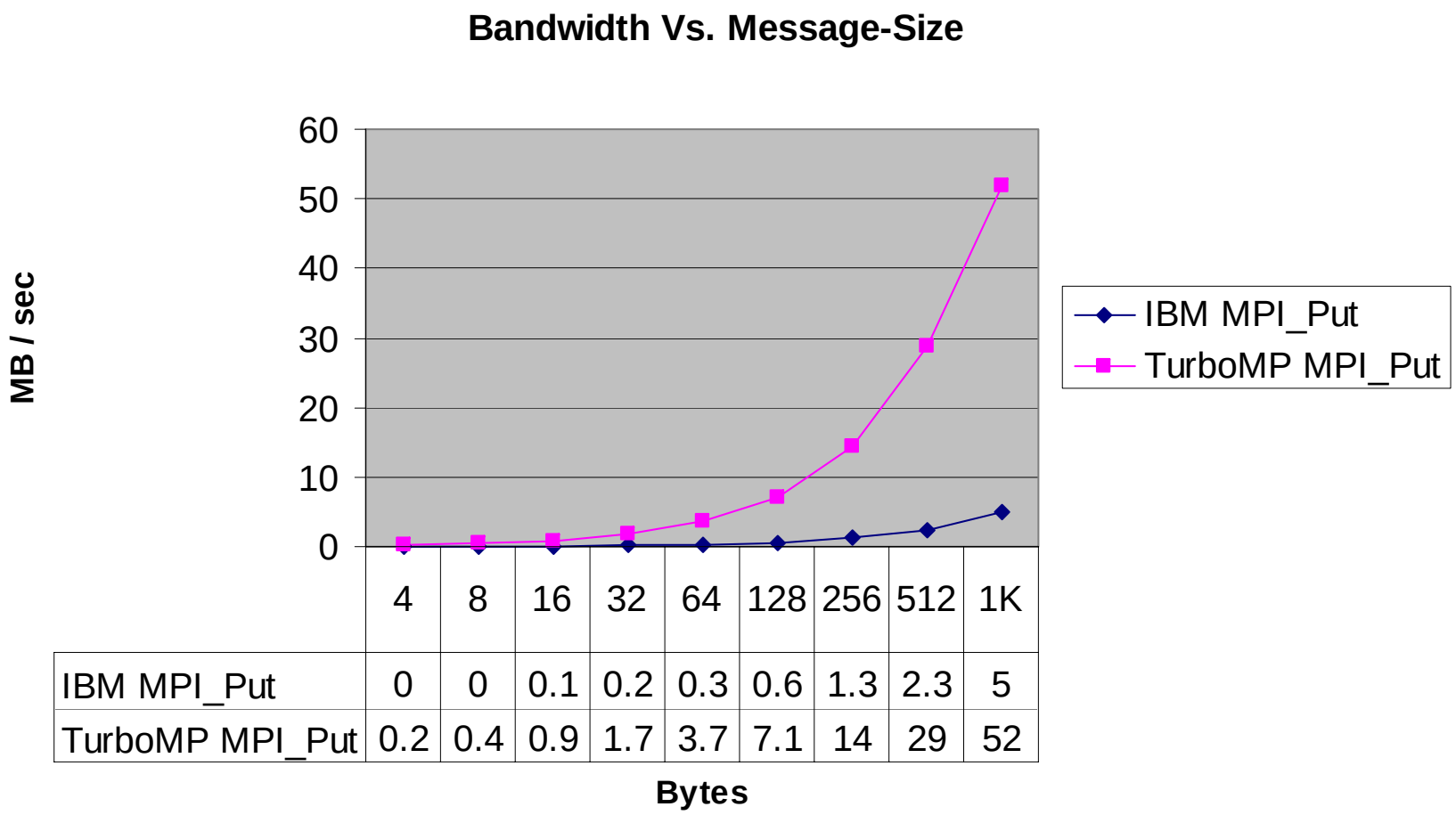
© 2004 IBM Corporation

MPI_PUT + MPI+Win_fence

```
for (i=0; i<NUM_ITER; i++)
if (rank == 0) {
   MPI_Put(outbuf, msg_sz, MPI_INT, 1, 0, msg_sz, MPI_INT, win);
 }
MPI_Win_fence(0, win);
 }
```


© 2004 IBM Corporation

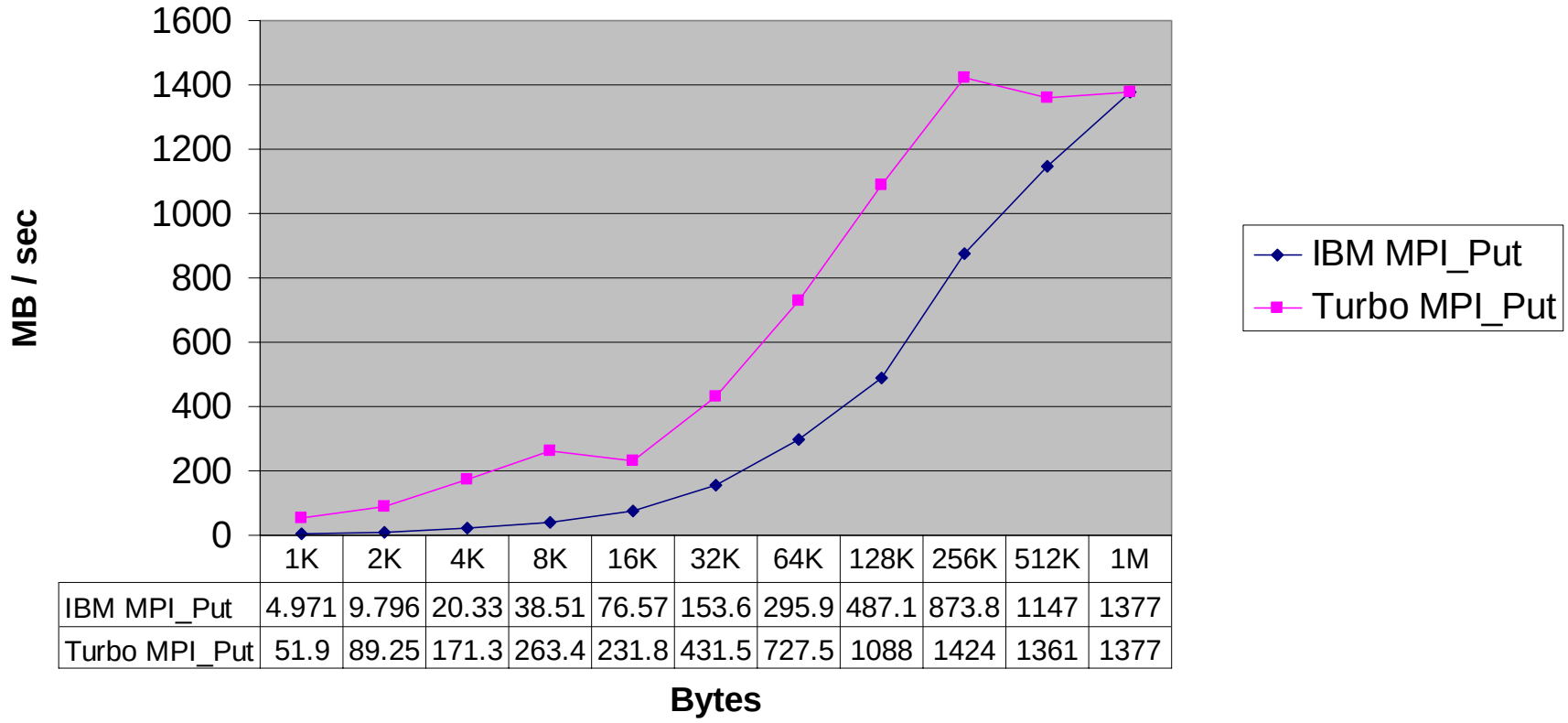
MPI_Win_Fence Synchronization (SMP)



	4	8	16	32	64	128	256	512	1K
IBM MPI_Put	0	0	0.1	0.2	0.3	0.6	1.3	2.3	5
TurboMP MPI_Put	0.2	0.4	0.9	1.7	3.7	7.1	14	29	52

© 2004 IBM Corporation

MPI_Win_Fence Synchronization (SMP)



	1K	2K	4K	8K	16K	32K	64K	128K	256K	512K	1M
IBM MPI_Put	4.971	9.796	20.33	38.51	76.57	153.6	295.9	487.1	873.8	1147	1377
Turbo MPI_Put	51.9	89.25	171.3	263.4	231.8	431.5	727.5	1088	1424	1361	1377

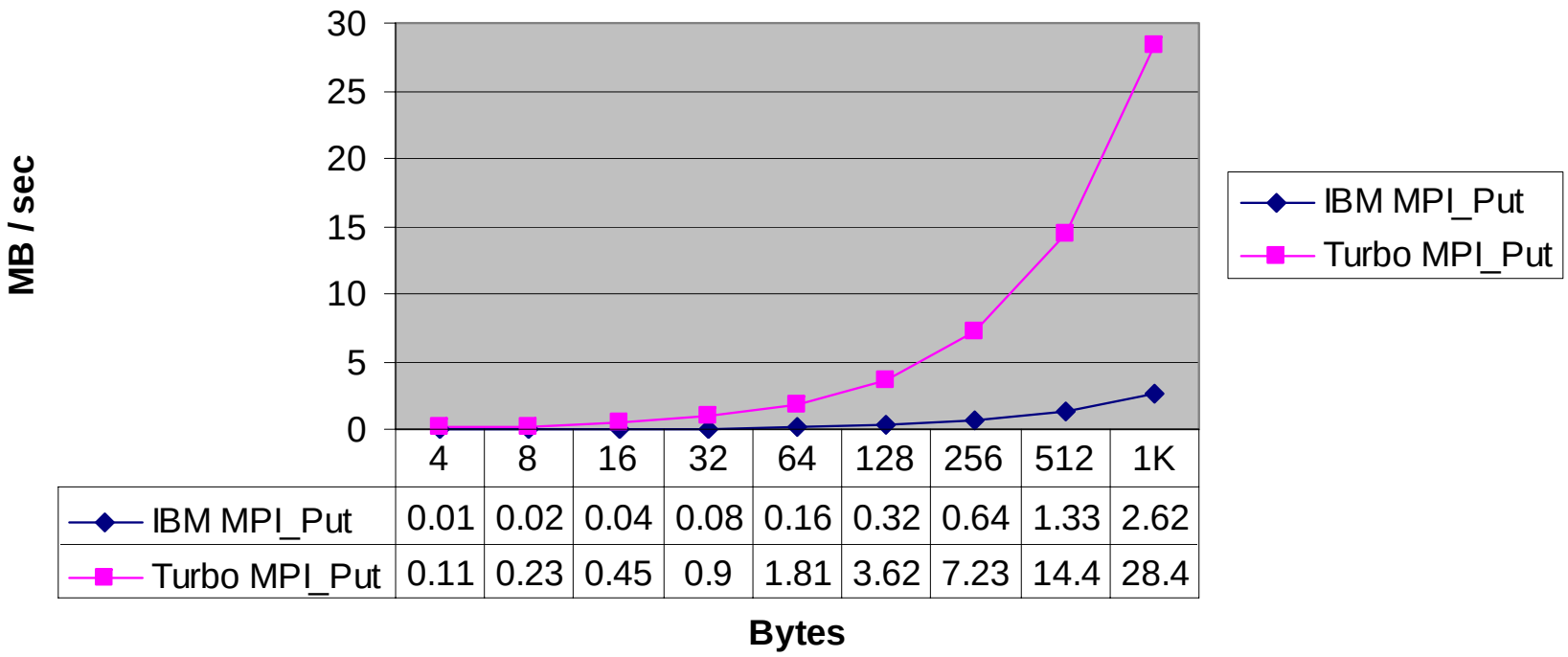
© 2004 IBM Corporation

MPI_Put + MPI_Win_lock/unlock

```
for (i=0; i<NUM_ITER; i++) {
        if (rank == 0) {
           MPI_Win_lock(lock_type, 1, 0, win);
           MPI_Put(outbuf, msg_sz, MPI_INT, 1, 0, msg_sz, MPI_INT, win);
           MPI_Win_unlock(1, win);
        }
    }
```


© 2004 IBM Corporation

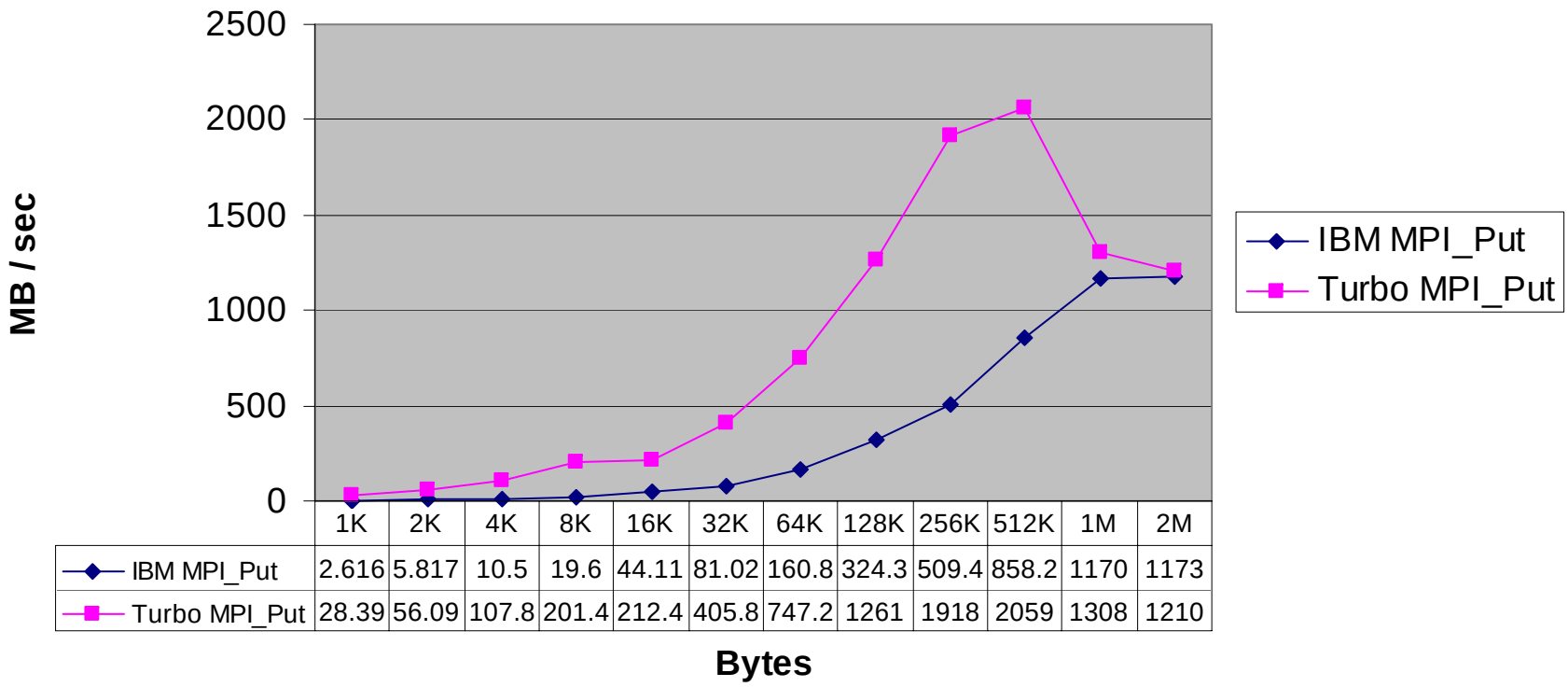
MPI_Win_Lock / Unlock – Exclusive Synchronization (SMP)



	4	8	16	32	64	128	256	512	1K
IBM MPI_Put	0.01	0.02	0.04	0.08	0.16	0.32	0.64	1.33	2.62
Turbo MPI_Put	0.11	0.23	0.45	0.9	1.81	3.62	7.23	14.4	28.4

© 2004 IBM Corporation

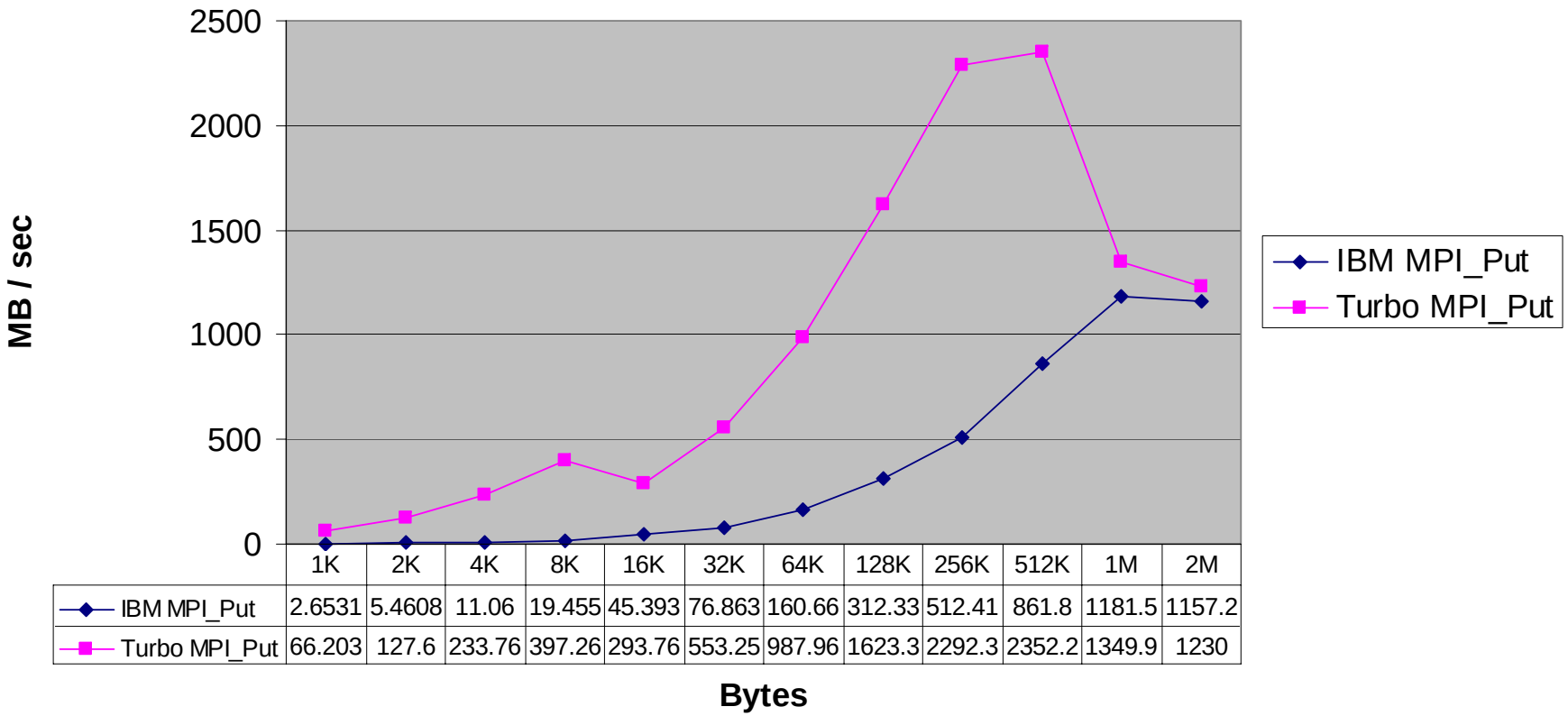
MPI_Win_Lock / Unlock – Exclusive Synchronization (SMP)



	1K	2K	4K	8K	16K	32K	64K	128K	256K	512K	1M	2M
IBM MPI_Put	2.616	5.817	10.5	19.6	44.11	81.02	160.8	324.3	509.4	858.2	1170	1173
Turbo MPI_Put	28.39	56.09	107.8	201.4	212.4	405.8	747.2	1261	1918	2059	1308	1210

© 2004 IBM Corporation

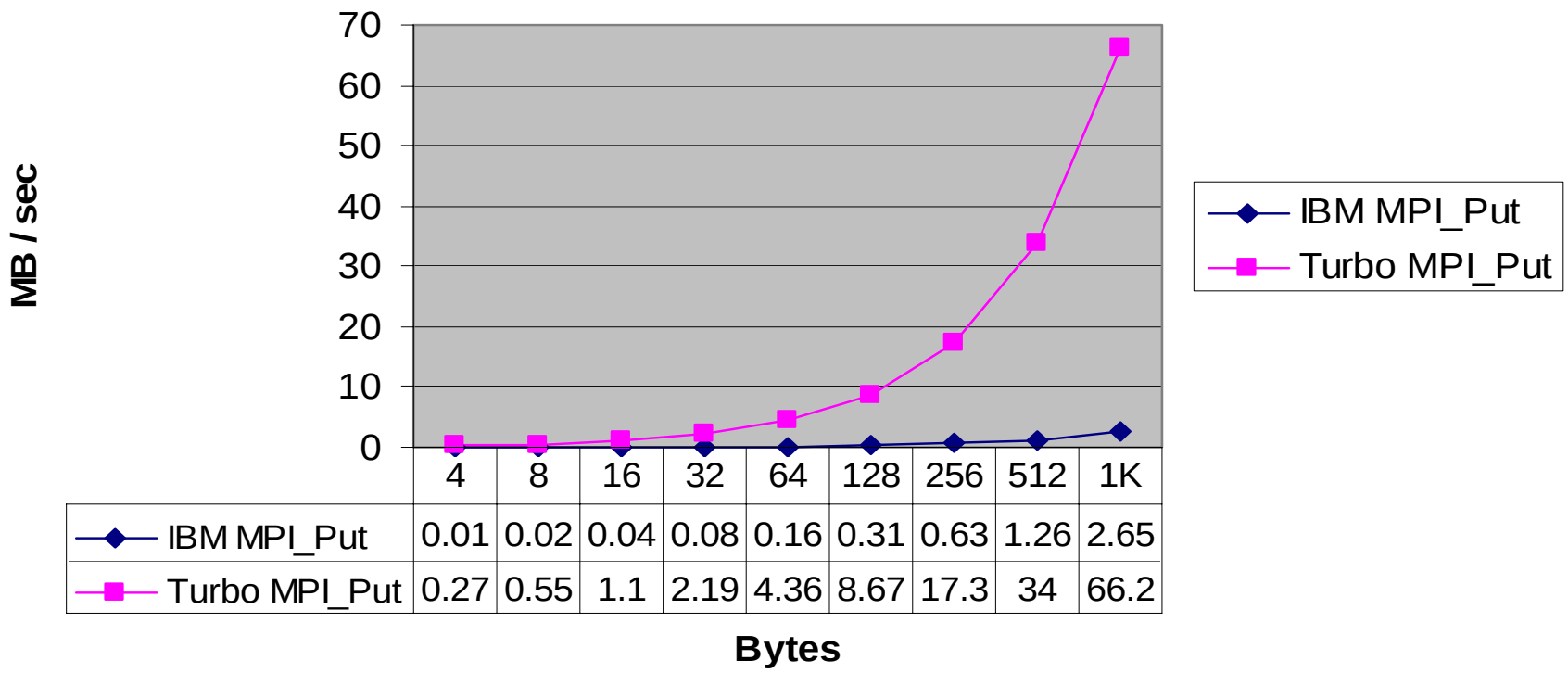
MPI_Win_Lock / Unlock – Shared Synchronization (SMP)



	1K	2K	4K	8K	16K	32K	64K	128K	256K	512K	1M	2M
IBM MPI_Put	2.6531	5.4608	11.06	19.455	45.393	76.863	160.66	312.33	512.41	861.8	1181.5	1157.2
Turbo MPI_Put	66.203	127.6	233.76	397.26	293.76	553.25	987.96	1623.3	2292.3	2352.2	1349.9	1230

© 2004 IBM Corporation

MPI_Win_Lock / Unlock – Shared Synchronization (SMP)



	4	8	16	32	64	128	256	512	1K
IBM MPI_Put	0.01	0.02	0.04	0.08	0.16	0.31	0.63	1.26	2.65
Turbo MPI_Put	0.27	0.55	1.1	2.19	4.36	8.67	17.3	34	66.2

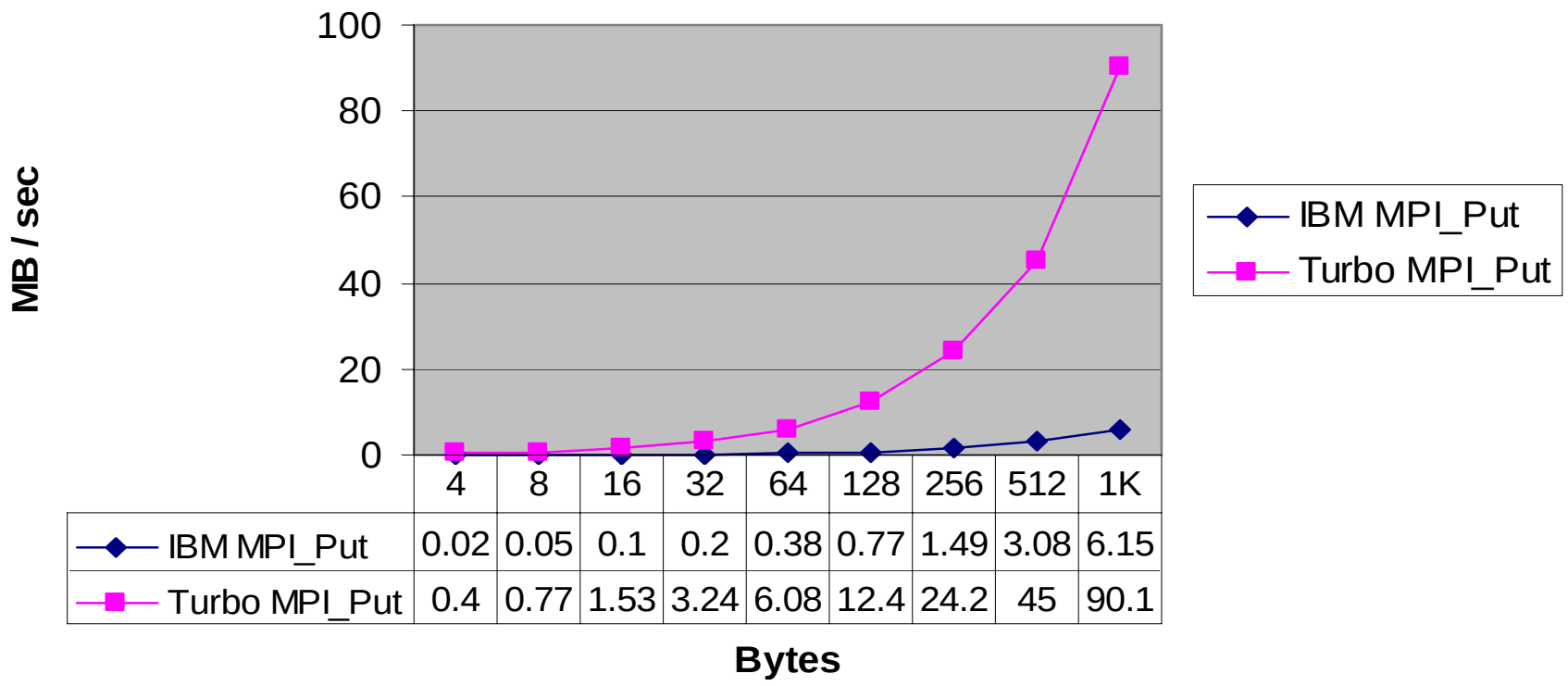
© 2004 IBM Corporation

MPI_Put + MPI_Win/start/post/complete/wait

```
for (i=0; i<NUM_ITER; i++) {
        MPI_Win_start(to_from_group, 0, win);
        MPI_Win_post(to_from_group, 0, win);
        MPI_Put(outbuf,msg_sz,MPI_INT,dest,0,msg_sz,MPI_INT,win);
        MPI_Win_complete(win);
        MPI_Win_wait(win);
    }
```


© 2004 IBM Corporation

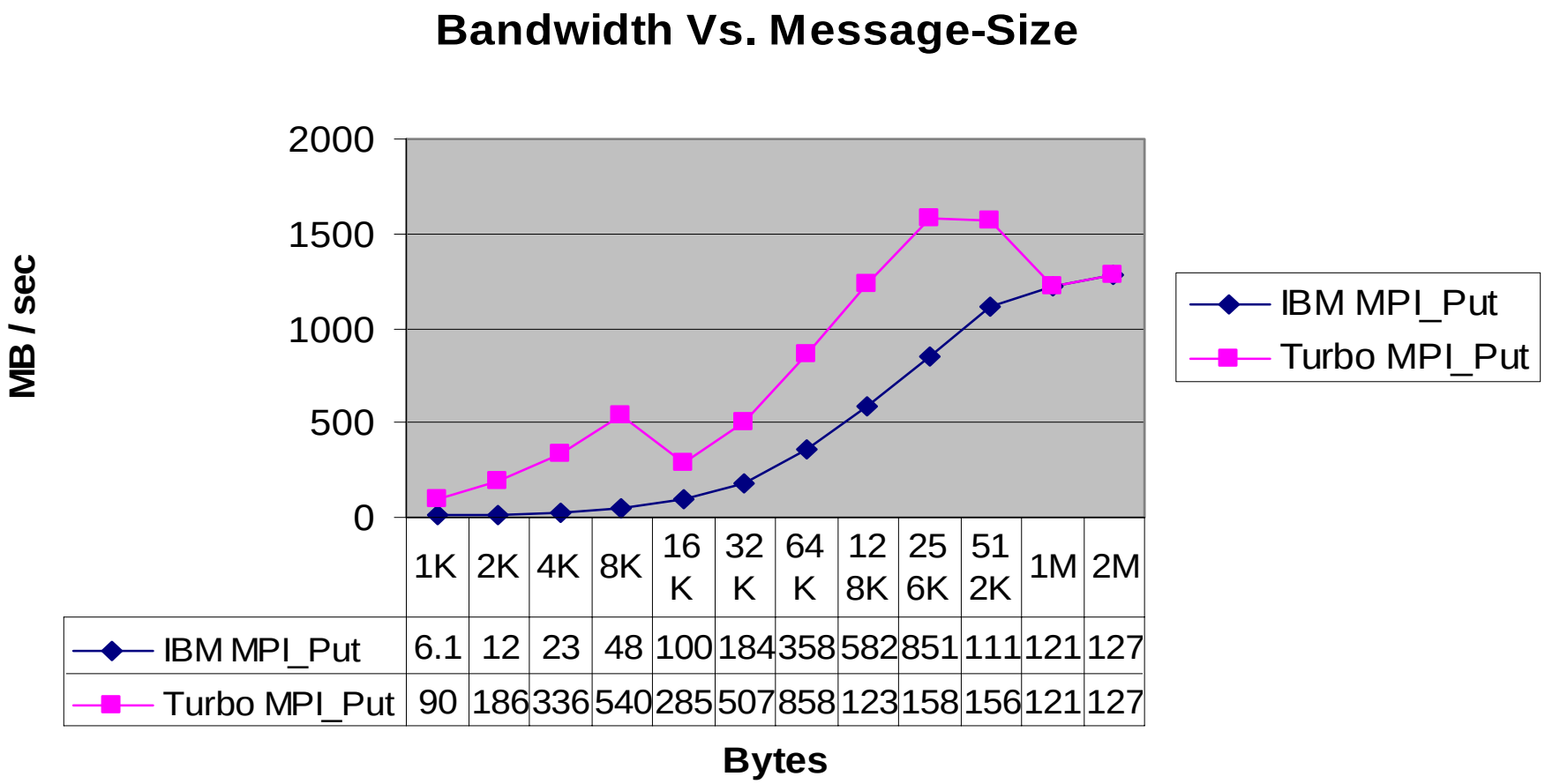
MPI_Start/Post/Wait/Complete Synchronization (SMP)



	4	8	16	32	64	128	256	512	1K
IBM MPI_Put	0.02	0.05	0.1	0.2	0.38	0.77	1.49	3.08	6.15
Turbo MPI_Put	0.4	0.77	1.53	3.24	6.08	12.4	24.2	45	90.1

© 2004 IBM Corporation

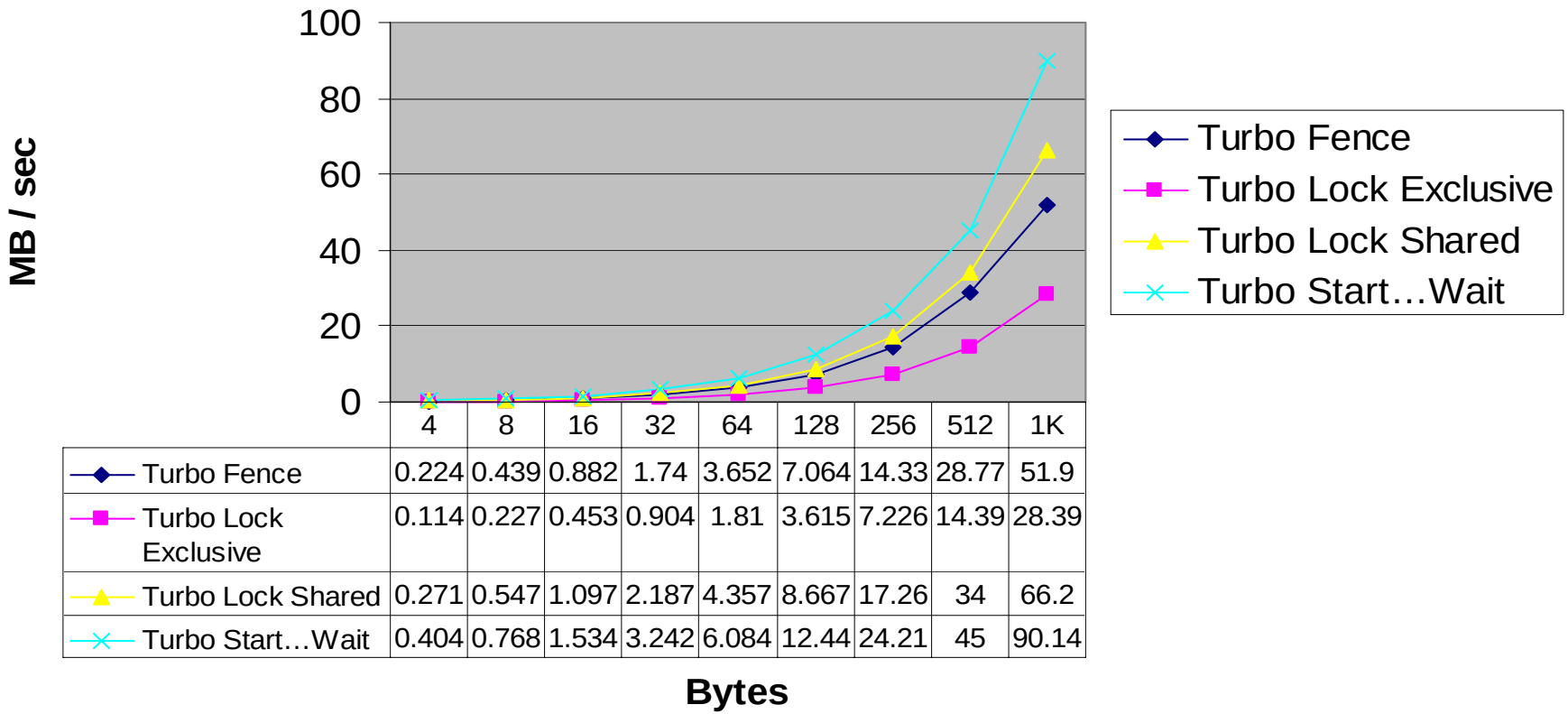
MPI_Start/Post/Wait/Complete Synchronization (SMP)



	1K	2K	4K	8K	16 K	32 K	64 K	12 8K	25 6K	51 2K	1M	2M
IBM MPI_Put	6.1	12	23	48	100	184	358	582	851	111	121	127
Turbo MPI_Put	90	186	336	540	285	507	858	123	158	156	121	127

© 2004 IBM Corporation

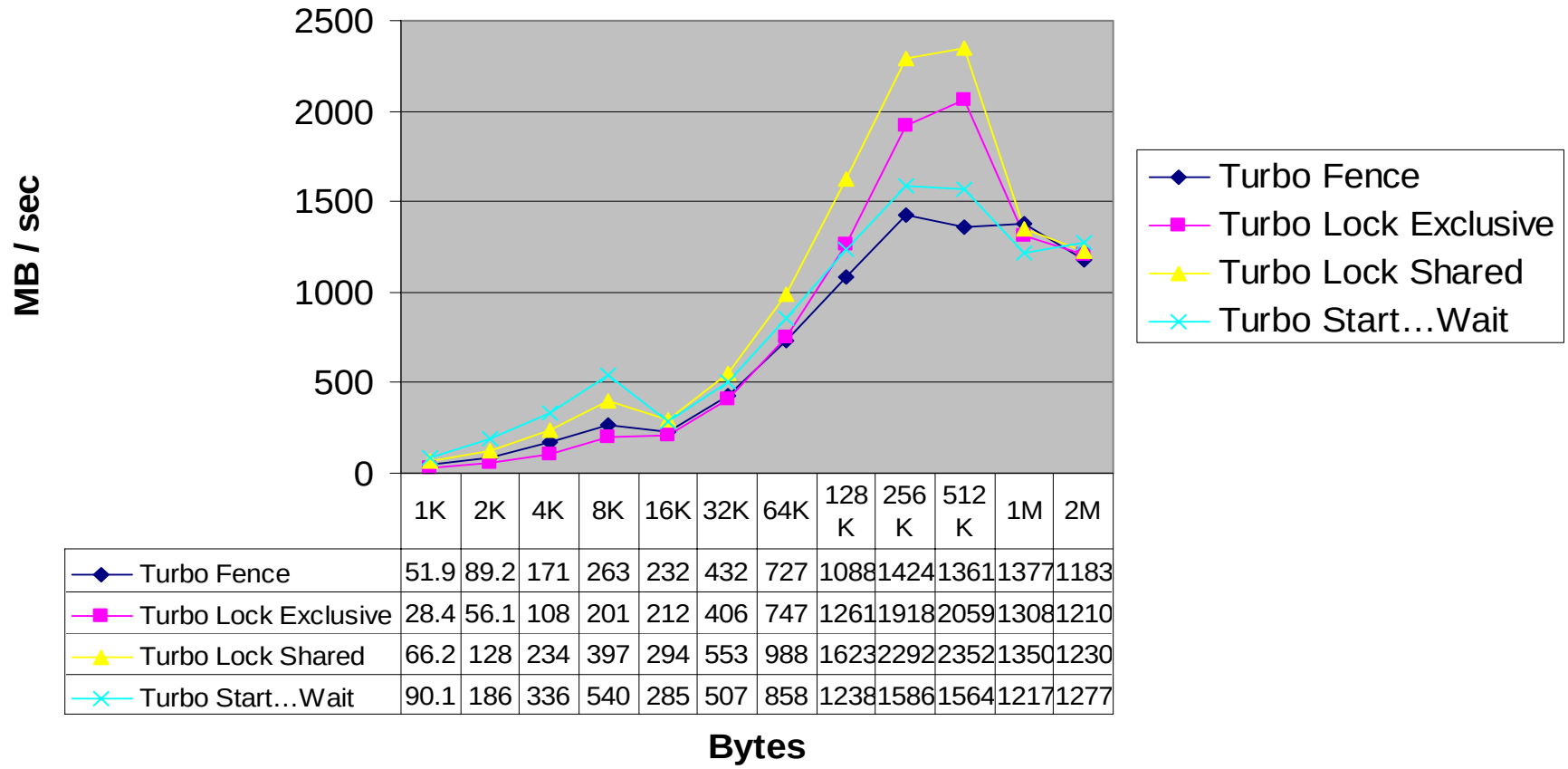
Turbo MPI_Put Comparison Synchronization (SMP)



	4	8	16	32	64	128	256	512	1K
Turbo Fence	0.224	0.439	0.882	1.74	3.652	7.064	14.33	28.77	51.9
Turbo Lock Exclusive	0.114	0.227	0.453	0.904	1.81	3.615	7.226	14.39	28.39
Turbo Lock Shared	0.271	0.547	1.097	2.187	4.357	8.667	17.26	34	66.2
Turbo Start…Wait	0.404	0.768	1.534	3.242	6.084	12.44	24.21	45	90.14

© 2004 IBM Corporation

Turbo MPI_Put Comparison Synchronization (SMP)



	1K	2K	4K	8K	16K	32K	64K	128 K	256 K	512 K	1M	2M
Turbo Fence	51.9	89.2	171	263	232	432	727	1088	1424	1361	1377	1183
Turbo Lock Exclusive	28.4	56.1	108	201	212	406	747	1261	1918	2059	1308	1210
Turbo Lock Shared	66.2	128	234	397	294	553	988	1623	2292	2352	1350	1230
Turbo Start…Wait	90.1	186	336	540	285	507	858	1238	1586	1564	1217	1277

© 2004 IBM Corporation

Put Latency Comparisons

	IBM MPI_Put (usec)	Turbo MPI_Put (usec)
Fence	62	18
Lock - Exclusive	179 → 83	35 → 30
Lock Shared	174 → 54	15 → 11
Start/Post/Wait/Complete	99	21
TurboSHMEM Put	N/A	**5**
ISend/IRecv	**11 → 5**	N/A

© 2004 IBM Corporation

How to Obtain the IBM HPC Toolkit?

- Current Possibilities:
  1. Acquire as part of new procurement.
  2. Acquire as part of ACTC Performance Tuning Workshop.
  3. Purchase license directly from IBM Research
- Future Possibility:

  Purchase from third-party vendor (e.g., Etnus)

© 2004 IBM Corporation

IBM HPC Toolkit Software Maintenance Cycle

- Major releases at SciComp meetings (twice a year)
- Minor releases (bug fixes) as-needed
- “New Technology” (e.g., DPOMP) placed on Alphaworks server:

  http://alphaworks.ibm.com

  6 month lifetime

  90-day evaluation licenses

© 2004 IBM Corporation