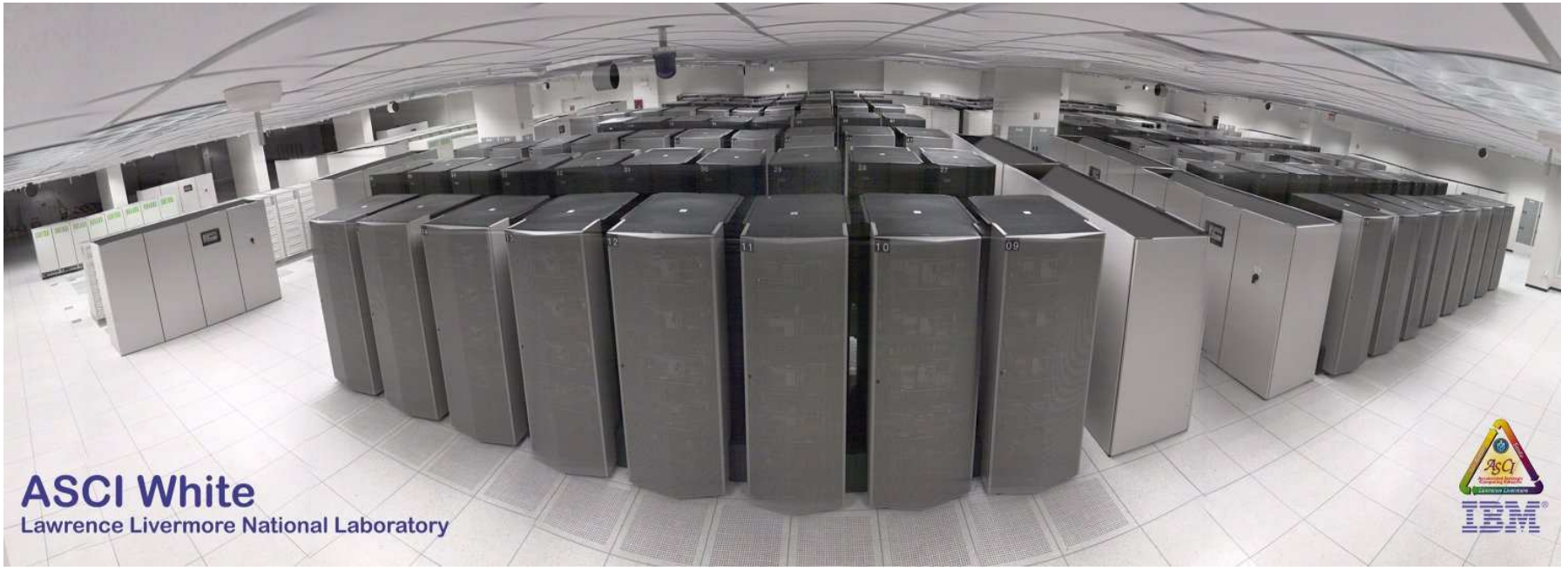
How to use Hybrid MPI-OpenMP on IBM SP Systems

Edmond Chow

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

ASCI White



Current and Emerging IBM Systems

Machine	No. Procs	CPU		Peak (TF/s)	Procs/Node
ASCI Blue-SST	5808	PPC 604e	332 MHz	3.9	4
Snow	128	Power3	222 MHz	0.1	8
ASCI White	8192	Power3	375 MHz	12.3	16
HPCx, pSeries 690	1280	Power4	1.3 GHz	6.7	32
Purple	12608	Power5	2.0 GHz	100	64
Blue Planet	16384	Power5	2.4 GHz	160	8
BG/L	131072	PPC 440	700 MHz	360	2

Trends:

- Blue Planet: with Virtual Vector Architecture
- BG/L: System-on-a-Chip
- Memory bandwidth and Network latency?

Hybrid programming

- Hybrid programs use multithreading within a node and message passing between nodes
- Seems natural for attaining high performance on SMP clusters (e.g., no MPI within a node)
- However, many hybrid codes do not achieve the performance of equivalent message passing codes
- Motivation: to try to understand hybrid program performance
- We investigate the combination of MPI and OpenMP. Many other alternatives are available.

Assumptions

- Domain decomposition
- OpenMP used to thread the outer loops
- The Hybrid and Pure-MPI programs do exactly the same work

When to definitely use a Hybrid model

- Shared memory parallel algorithm (not domain decomposition) is more efficient than MPI parallel algorithm
- Algorithm is more efficient using larger subdomains
- Different lines of execution in the code are naturally executed by threads (e.g., master and slave)
- Load balancing is needed within a node

Timings for SPMV on four 16-way SMP nodes

40x40x3 per node

Tasks/Threads per node	Time (ms)
1/16	0.33
4/4	0.33
16/1	0.29

120x120x3 per node

Tasks/Threads per node	Time (ms)
1/16	2.28
4/4	1.93
16/1	1.53

Timings for SPMV on four 16-way SMP nodes

40x40x3 per node

Tasks/Threads per node	Comm. (ms)	Comp. (ms)	Total (ms)
1/16	0.10	0.17	0.33
2/8	0.13	0.15	0.30
4/4	0.19	0.13	0.33
8/2	0.19	0.10	0.29
16/1	0.22	0.08	0.29

120x120x3 per node

Tasks/Threads per node	Comm. (ms)	Comp. (ms)	Total (ms)
1/16	0.29	1.96	2.28
2/8	0.28	1.84	2.14
4/4	0.31	1.62	1.93
8/2	0.26	1.40	1.65
16/1	0.35	1.26	1.53

Factors affecting hybrid program performance

1. Thread synchronization, loop scheduling, and other overheads
2. Critical sections that cannot be multithreaded
3. Data partitionings based on SMP nodes rather than threads, leading to poor cache utilization compared to pure message passing programs
4. Differences in communication performance when fewer message passing processes share a network interface
5. Differences in scalability since fewer processes are involved in message passing communication

Factors 1, 3, and 4 affect the results in the previous tables.

Communication parameters for 16-way SMP nodes

	Number of pairs	Overhead (μs)	Bandwidth (MB/s)	
			Max per proc	Aggregate
off-node	1	26.3	202.2	202.2
	2	29.2	186.6	373.2
	4	33.3	145.3	581.2
	8	58.4	82.5	660.0
	16	106.2	41.4	662.4

Communication parameters for 16-way SMP nodes

	Number of pairs	Overhead (μs)	Bandwidth (MB/s)	
			Max per proc	Aggregate
off-node	1	26.3	202.2	202.2
	2	29.2	186.6	373.2
	4	33.3	145.3	581.2
	8	58.4	82.5	660.0
	16	106.2	41.4	662.4
on-node	1	26.4	174.3	174.3
	2	32.8	149.4	298.8
	4	55.4	83.3	333.2
	8	106.8	42.3	338.4
on-node (sh.mem)	1	19.5	306.6	306.6
	2	20.8	269.9	539.8
	4	22.1	228.5	914.0
	8	22.1	139.0	1112.0

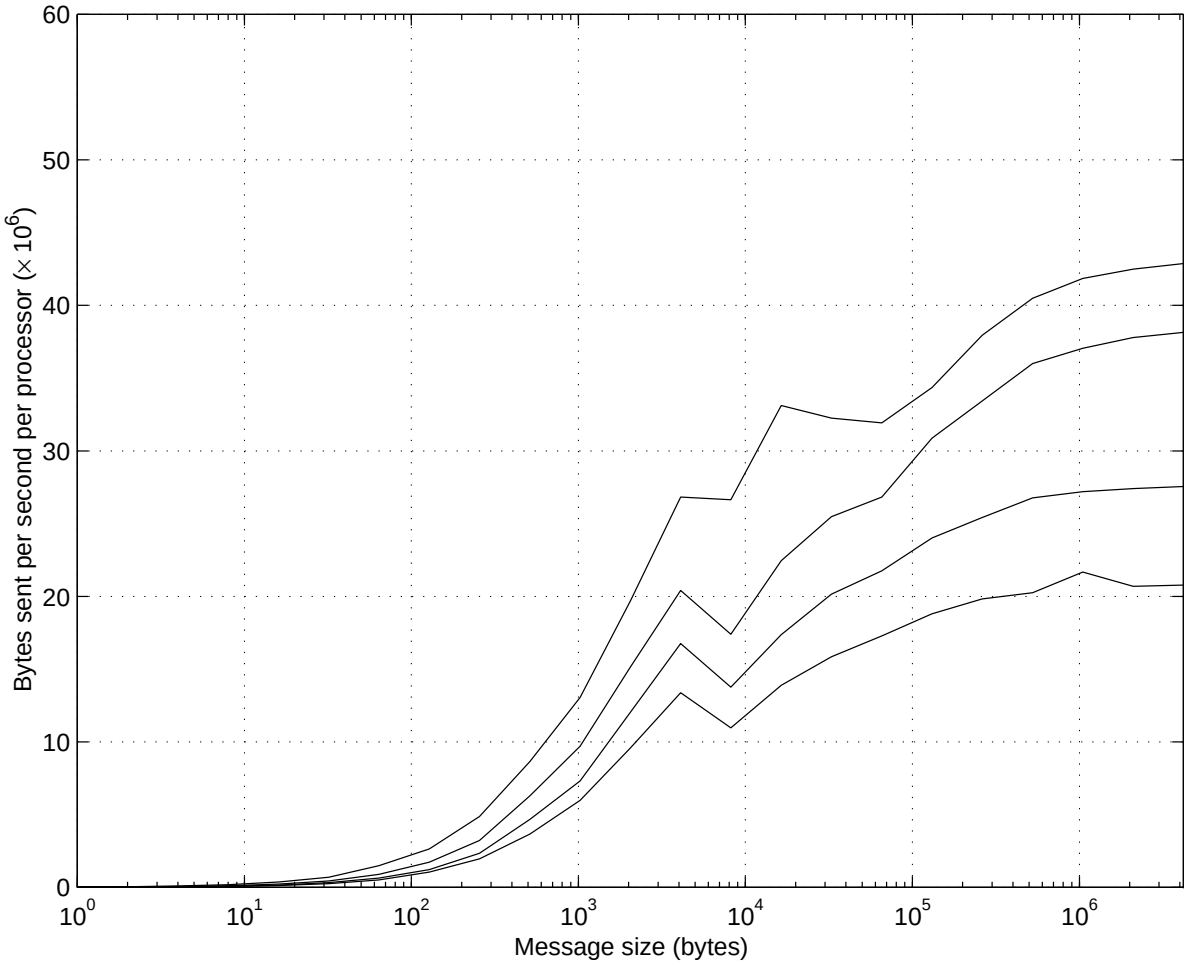
Communication parameters for 8-way SMP nodes

	Number of pairs	Overhead (μs)	Bandwidth (MB/s)	
			Max per proc	Aggregate
off-node	1	52.8	130.5	130.5
	2	51.6	123.8	247.6
	4	46.5	84.4	337.6
	8	63.0	43.1	344.8
on-node	1	46.2	128.9	257.8
	2	44.3	88.3	353.2
	4	61.3	43.8	350.4
on-node (sh.mem)	1	27.5	175.4	350.8
	2	28.7	174.3	697.2
	4	30.3	168.8	1350.4

Communication parameters for 4-way SMP nodes

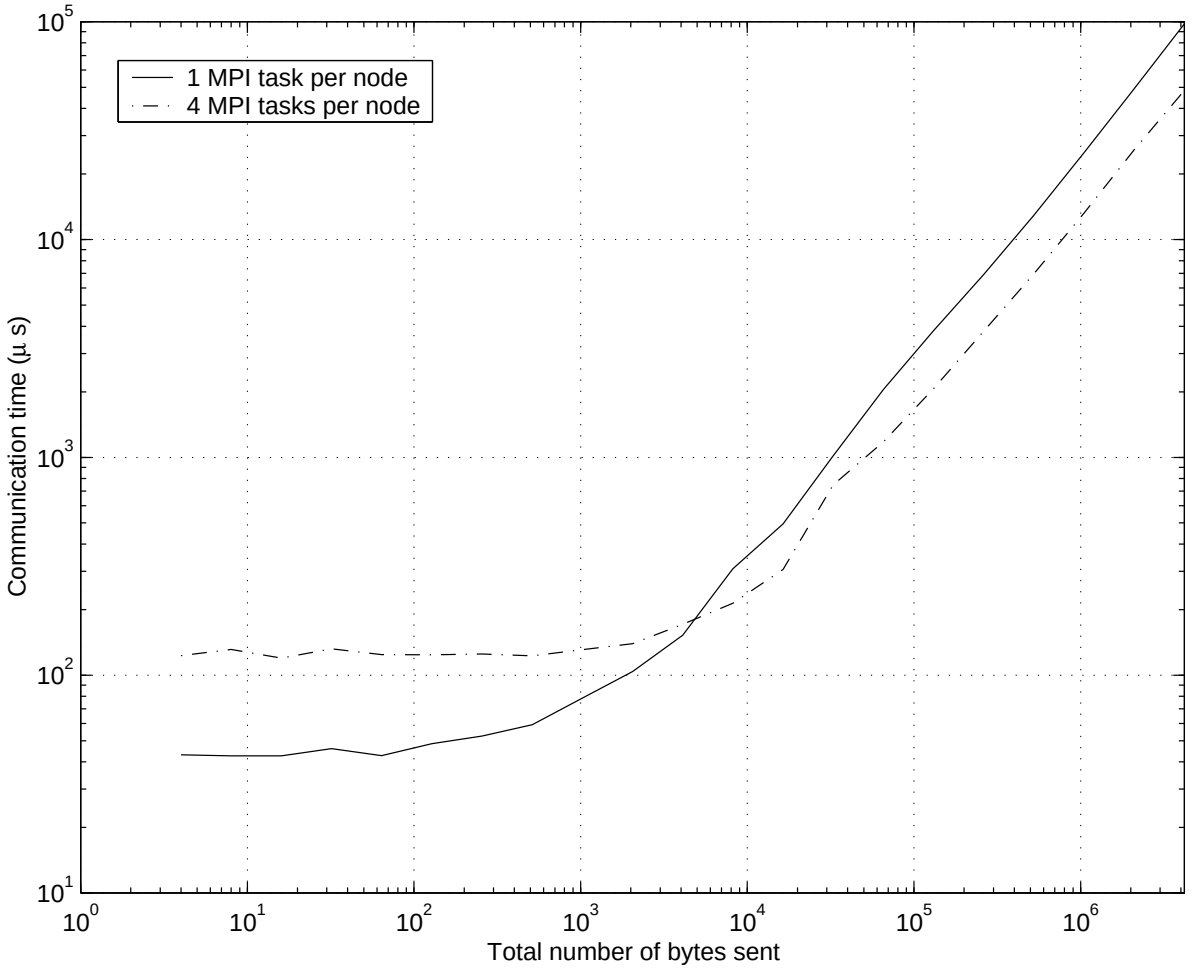
	Number of pairs	Overhead (μs)	Bandwidth (MB/s)	
			Max per proc	Aggregate
off-node	1	43.5	42.9	42.9
	2	71.5	38.2	76.4
	3	98.1	27.6	82.8
	4	125.2	20.8	83.2
on-node	1	76.1	36.8	73.6
	2	136.0	20.7	82.8
on-node (sh.mem)	1	28.2	54.8	109.6
	2	30.3	59.8	239.2

Megabytes sent per second per processor



Off-node communication using 1 (top) to 4 (bottom) processors per node

Communication time



Communication time with 1 processor per node (solid) and 4 processors per node (dashed)

Model program

- SPMV kernel using matrix from a 3-D structured grid
- 3-D array is split up into subdomains for each node or processor
- communication between processors with neighboring subdomains (ghost layer exchange); at most 26 neighbors
- parameterizing the global and local problem sizes leads to realistic changes in the communication-computation ratio
- $t = t^{comm} + t^{comp}$, OpenMP applied to computation section
- results generally shown for IBM SP with silver nodes

Communication timings for SPMV on two nodes (8 processors)

Prob. Size	Communication time (s)		
	MPI	MPI-sh.mem	Hybrid
500	0.041	0.033	0.010
4000	0.075	0.057	0.038
13500	0.081	0.068	0.067
32000	0.140	0.115	0.086
62500	0.202	0.126	0.125
108000	0.210	0.172	0.150
171500	0.315	0.253	0.195
256000	0.406	0.326	0.213

- Hybrid mode is advantageous here
- Off-node comm. masks shared memory on-node comm.

Large number of nodes – small local problem size

Prob. Size	Number of Nodes	Total time (s)		Comm. time (s)	
		MPI	Hybrid	MPI	Hybrid
500	1	0.029	0.015	0.021	0.000
	8	0.071	0.042	0.063	0.027
	27	0.119	0.101	0.111	0.086
	64	0.141	0.126	0.133	0.111
	125	0.297	0.148	0.289	0.133
	216	0.416	0.182	0.408	0.167

- Again, Hybrid performance better for short messages.
- Any gains with large numbers of nodes is masked by other effects.
- Reserve a processor for daemon processes; limited by memory bandwidth anyway.

Large number of nodes – large local problem size

Prob. Size	Number of Nodes	Total time (s)		Comm. time (s)	
		MPI	Hybrid	MPI	Hybrid
256000	1	16.359	16.228	0.280	0.000
	8	17.614	17.399	1.535	1.171
	27	19.103	19.912	3.024	3.684
	64	19.472	21.960	3.393	5.732
	125	20.331	22.859	4.252	6.631
	216	21.178	24.057	5.099	7.829

Conclusions

- When computation dominates communication, hybrid performance mostly depends on the size of overheads and critical sections
- Cache behavior of hybrid programs may be worse
- Hybrid programs are better for programs with short messages
- Possible *algorithmic* improvements is the best reason to use hybrid programming, i.e., don't just think domain decomposition!

Additional information

See the paper, "Assessing Performance of Hybrid MPI/OpenMP Programs on SMP Clusters," available at

`http://www.llnl.gov/casc/people/chow/pubs/hpaper.ps`

(submitted to J. Parallel Dist. Comput.) or contact the author at

`echow@llnl.gov`

Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.