Performance Analysis of Java Message-Passing Libraries on Fast Ethernet, Myrinet and SCI Clusters

Guillermo L. Taboada, Juan Touriño and Ramon Doallo
{taboada,juan,doallo}@udc.es

Dept. of Electronics and Systems
University of A Coruña, Spain

IEEE Cluster 2003 – Hong Kong, DECEMBER 2003

Overview

- Our analysis combines:
  - State of the Art in Java Message-Passing Libraries
  - Modeling Message-Passing Primitives
  - Performance Analysis on Fast Ethernet, Myrinet and SCI clusters
- Results:
  - Evaluation of the most outstanding Java Message-Passing Libraries and performance implications

Outline

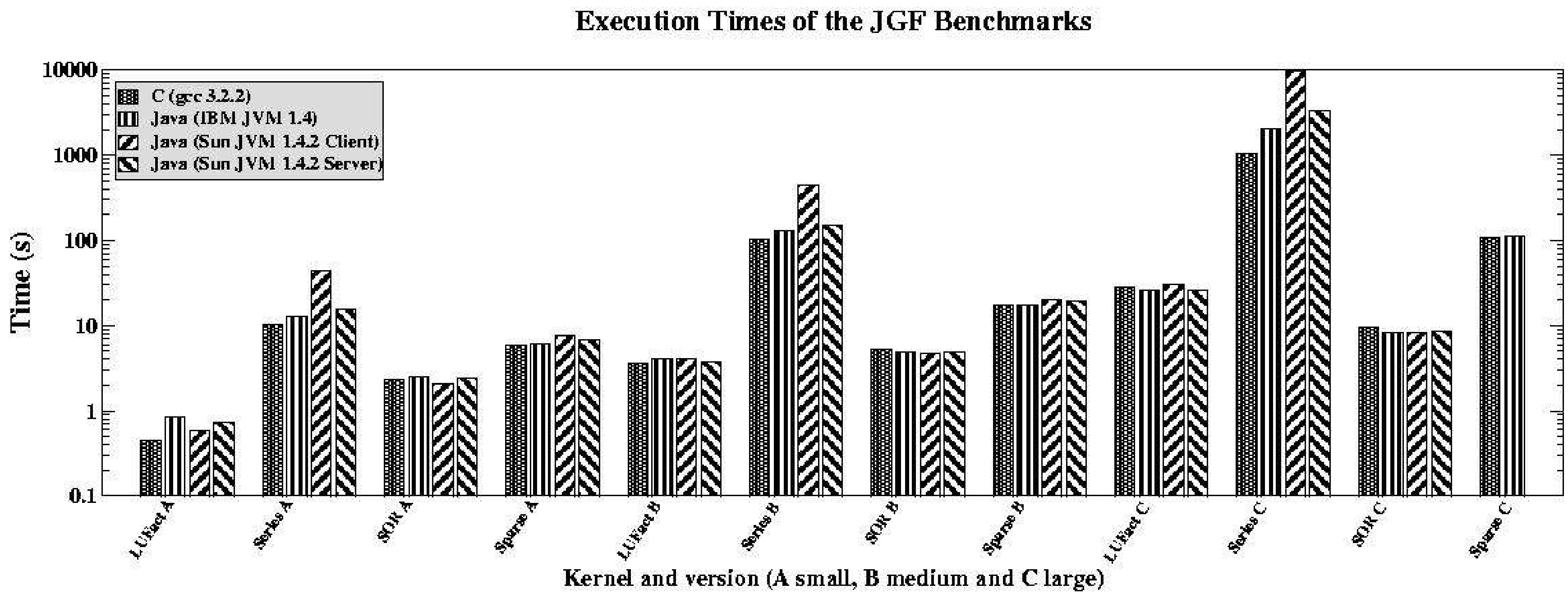
- Introduction
- Java Message-Passing Libraries
- Modeling Message-Passing Primitives
- Experimental Conditions
- Experimental Results
- Analysis of Performance Results
- Conclusions

Introduction

- Cluster Computing Architectures are an emerging option
- Java Message-Passing Libraries are an alternative due to:
  - Platform independence
  - Portability
  - Integration
  - … although probably at a performance cost
- Overall application performance will largely depend on the performance of the underlying Java MP libraries
- Our goal: Provide performance results for Fast Ethernet, Myrinet and SCI clusters which can guide developers of Java parallel applications

Introduction

- Nowadays, differences between Java and native codes are narrower



Introduction

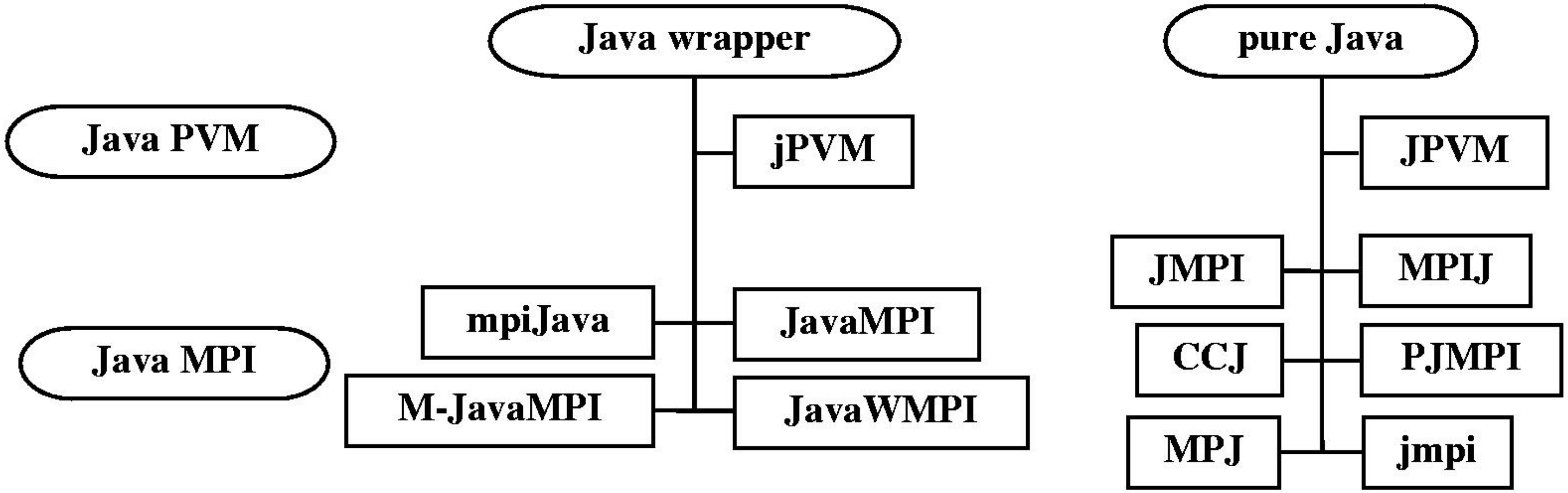
- Related work: Papers by Stankovic and Zhang [1] and by Getov, Gray and Sunderam [2]. Both works evaluate out-of-date libraries and do not derive performance analytical models
- Main contributions:
  - Survey of the state of the art in Java message-passing libraries
  - An updated performance evaluation of these libraries on Fast Ethernet, Myrinet and SCI clusters
  - Their performance analytical models

Java Message-Passing Libraries

- Two main types of implementations of Message-Passing for Java:
  - **Java wrapper** provides efficient MPI communication through calling native methods using JNI. The major drawback is lack of portability
  - **Pure Java** provides a portable message-passing implementation since the whole library is developed in Java, although the communication is less efficient

Java Message-Passing Libraries

- Taxonomy of Existing Libraries:



Java Message-Passing Libraries

- Selected Java Message-Passing libraries:
  - **mpiJava** the most active Java wrapper project
  - **CCJ** pure Java implementation of the Manta Team (Netherlands)
  - **JMPI** pure Java implementation of the Univ. of Massachusetts

Modeling Message-Passing Primitives

- Point-to-point communications:

$$T(n) = t_s + t_b n$$
$$Bw(n) = n/T(n)$$

- Collective communications:

$$T(n,p) = t_s(p) + t_b(p)\,n$$
$$Bw(n,p) = n/T(n,p)$$

T: message latency

n: message size (bytes)

t_s : startup time

t_b : transfer time (per byte)

p: processors.

Modeling Message-Passing Primitives

- We have developed our own microbenchmark suite adapted to our specific needs (eg, timing outliers are discarded)
  - Point to point primitives
    - Ping-Pong test is repeated several times in a loop (from 0B to 1MB messages)
    - Least-squares fit of *T* against *n* using minimal times
  - Collective primitives
    - Message size: from 0B to 1MB messages
    - Number of processors: up to the maximum number of processors of the clusters
    - Least-square fit of *T* against *n* and *p*

Experimental Conditions

- Cluster bw:
  - 16 nodes PIII at 1 GHz with 512MB of memory
  - Interconnected via Myrinet and Fast Ethernet
  - Linux Red Hat 7.1, kernel 2.4.7-10 and gcc 2.96
- Cluster muxia
  - 8 nodes PIV Xeon at 1.8 GHz (hyperthreading disabled)
  - 1GB of memory
  - Interconnected via SCI (Scalable Coherent Interface)
  - Linux Red Hat 7.3, kernel 2.4.19 and gcc 2.96

Experimental Conditions

- JVM:
  - IBM JVM 1.4 JITC
  - Sun JVM Sun 1.4.1 HotSpot
- Libraries
  - MPICH 1.2.4, MPICH-GM and SCI-MPICH
  - ScaMPI
  - mpiJava 1.2.5 over native libraries
  - CCJ 0.1
  - JMPI

Analytical models

- Send metrics on Fast Ethernet:

Library	*Analytical Model*		*Experimental Results*	
	t_s {µs}	t_b {ns/B}	T(16B){µs}	Bw(1MB){MB/s}
MPICH	69	90.3	72	11.061
mpiJava	101	100.7	105	9.923
CCJ	800	138.2	800	7.217
JMPI	4750	154.4	4750	6.281

Analytical models

- Send metrics on Myrinet:

Library	Analytical Model		Experimental Results	
	t_s {μs}	t_b {ns/B}	T(16B){μs}	Bw(1MB){MB/s}
MPICH-GM	9	5.4	10	183.30
mpiJava	15	5.26	16	189.21
CCJ	650	33.68	700	29.13
JMPI	3850	52.42	3850	17.88

Analytical models

- Send metrics on SCI:

Library	*Analytical Model*		*Experimental Results*	
	t_s {μs}	t_b {ns/B}	T(16B){μs}	Bw(1MB){MB/s}
ScaMPI	4	3.89	5	256.88
mpiJava	10	3.92	10	254.26
CCJ	500	86.16	500	11.54
JMPI	950	90.71	950	10.91

Analytical models

- Send metrics on Fast Ethernet, Myrinet and SCI:

Library	Analytical Model		Experimental Results	
	t_s {µs}	t_b {ns/B}	T(16B){µs}	Bw(1MB){MB/s}
MPICH	69	90.3	72	11.061
mpiJava	101	100.7	105	9.923
CCJ	800	138.2	800	7.217
JMPI	4750	154.4	4750	6.281

Library	Analytical Model		Experimental Results	
	t_s {µs}	t_b {ns/B}	T(16B){µs}	Bw(1MB){MB/s}
MPICH-GM	9	5.4	10	183.30
mpiJava	15	5.26	16	189.21
CCJ	650	33.68	700	29.13
JMPI	3850	52.42	3850	17.88

Library	Analytical Model		Experimental Results	
	t_s {µs}	t_b {ns/B}	T(16B){µs}	Bw(1MB){MB/s}
ScaMPI	4	3.89	5	256.88
mpiJava	10	3.92	10	254.26
CCJ	500	86.16	500	11.54
JMPI	950	90.71	950	10.91

Analytical models

- Broadcast metrics on Fast Ethernet ($lp=\log_2 p$) :

Library	Analytical Model		Experimental Results	
	$t_s(p)$ {μs}	$t_b(p)$ {ns/B}	T(16B,8)	Bw(1MB,8)
MPICH	$7+117\lceil lp \rceil$	$-0.3+90.4\lceil lp \rceil$	364	3.686
mpiJava	$19+124\lceil lp \rceil$	$10.1+90.5\lceil lp \rceil$	406	3.546
CCJ	$-430+1430\lceil lp \rceil$	$6.4+130.4\lceil lp \rceil$	3800	2.506
JMPI	-9302+7151p	-123.2+175.7p	41600	0.752

Analytical models

- Broadcast metrics on Myrinet ($lp=\log_2 p$) :

Library	Analytical Model		Experimental Results	
	$t_s(p)$ {μs}	$t_b(p)$ {ns/B}	T(16B,8)	Bw(1MB,8)
MPICH-GM	$3+8\lceil lp \rceil$	$0.012+5.741\lceil lp \rceil$	28	57.77
mpiJava	$20+17\lceil lp \rceil$	$0.036+5.263\lceil lp \rceil$	101	62.78
CCJ	$-800+1600\lceil lp \rceil$	$-10.64+40.69\lceil lp \rceil$	4000	8.72
JMPI	$-8617+5356p$	$-61.57+66.7p$	32400	1.80

Analytical models

- Broadcast metrics on SCI ($lp = \log_2 p$) :

Library	*Analytical Model*		*Experimental Results*	
	$t_s(p)$ {μs}	$t_b(p)$ {ns/B}	T(16B,8)	Bw(1MB,8)
ScaMPI	$6\lceil lp \rceil$	$-0.105+4.128\lceil lp \rceil$	18	81.71
mpiJava	$33+7\lceil lp \rceil$	$-0.197+4.458\lceil lp \rceil$	48	76.71
CCJ	$-800+1400\lceil lp \rceil$	$-4.242+93.01\lceil lp \rceil$	3400	3.63
JMPI	-2800+2900p	-89.71+95.52p	20600	1.45

Analytical models

- Broadcast metrics on Fast Ethernet, Myrinet and SCI :

Library	Analytical Model		Experimental Results	
	$t_s(p)$ {µs}	$t_b(p)$ {ns/B}	T(16B,8)	Bw(1MB,8)
MPICH	$7+117\lceil lp \rceil$	$-0.3+90.4\lceil lp \rceil$	364	3.686
mpiJava	$19+124\lceil lp \rceil$	$10.1+90.5\lceil lp \rceil$	406	3.546
CCJ	$-430+1430\lceil lp \rceil$	$6.4+130.4\lceil lp \rceil$	3800	2.506
JMPI	$-9302+7151p$	$-123.2+175.7p$	41600	0.752

Library	Analytical Model		Experimental Results	
	$t_s(p)$ {µs}	$t_b(p)$ {ns/B}	T(16B,8)	Bw(1MB,8)
MPICH-GM	$3+8\lceil lp \rceil$	$0.012+5.741\lceil lp \rceil$	28	57.77
mpiJava	$20+17\lceil lp \rceil$	$0.036+5.263\lceil lp \rceil$	101	62.78
CCJ	$-800+1600\lceil lp \rceil$	$-10.64+40.69\lceil lp \rceil$	4000	8.72
JMPI	$-8617+5356p$	$-61.57+66.7p$	32400	1.80

Library	Analytical Model		Experimental Results	
	$t_s(p)$ {µs}	$t_b(p)$ {ns/B}	T(16B,8)	Bw(1MB,8)
ScaMPI	$6\lceil lp \rceil$	$-0.105+4.128\lceil lp \rceil$	18	81.71
mpiJava	$33+7\lceil lp \rceil$	$-0.197+4.458\lceil lp \rceil$	48	76.71
CCJ	$-800+1400\lceil lp \rceil$	$-4.242+93.01\lceil lp \rceil$	3400	3.63
JMPI	$-2800+2900p$	$-89.71+95.52p$	20600	1.45

Analytical models

- Collective metrics on Myrinet (several design issues):

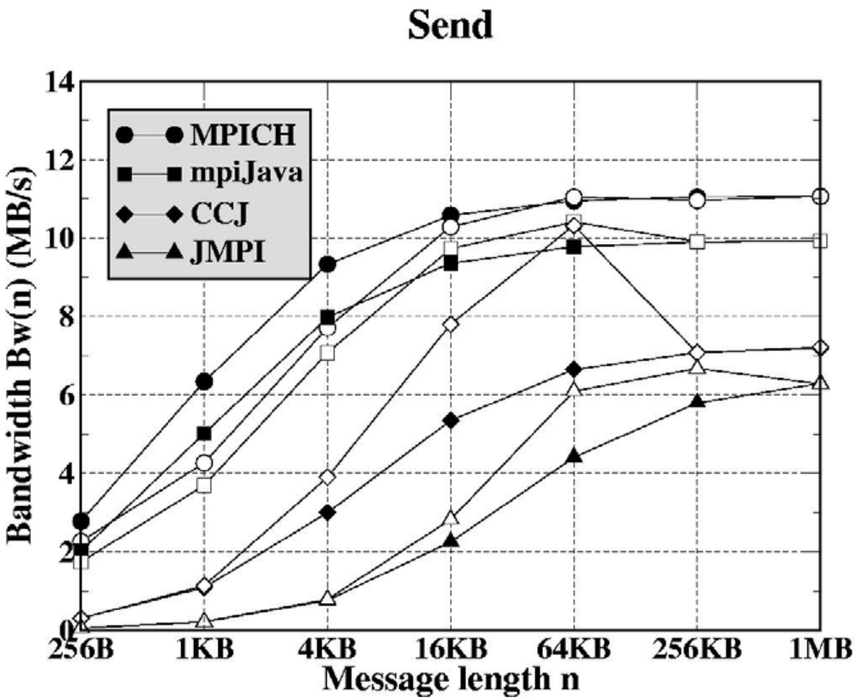
Library	Broadcast Analytical Model		Experimental Results	
	$t_s(p)$ {μs}	$t_b(p)$ {ns/B}	T(16B,8)	Bw(1MB,8)
MPICH-GM	$3+8\lceil lp \rceil$	$0.012+5.741\lceil lp \rceil$	28	57.77
mpiJava	$20+17\lceil lp \rceil$	$0.036+5.263\lceil lp \rceil$	101	62.78
CCJ	$-800+1600\lceil lp \rceil$	$-10.64+40.69\lceil lp \rceil$	4000	8.72
JMPI	$-8617+5356p$	$-61.57+66.7p$	32400	1.80

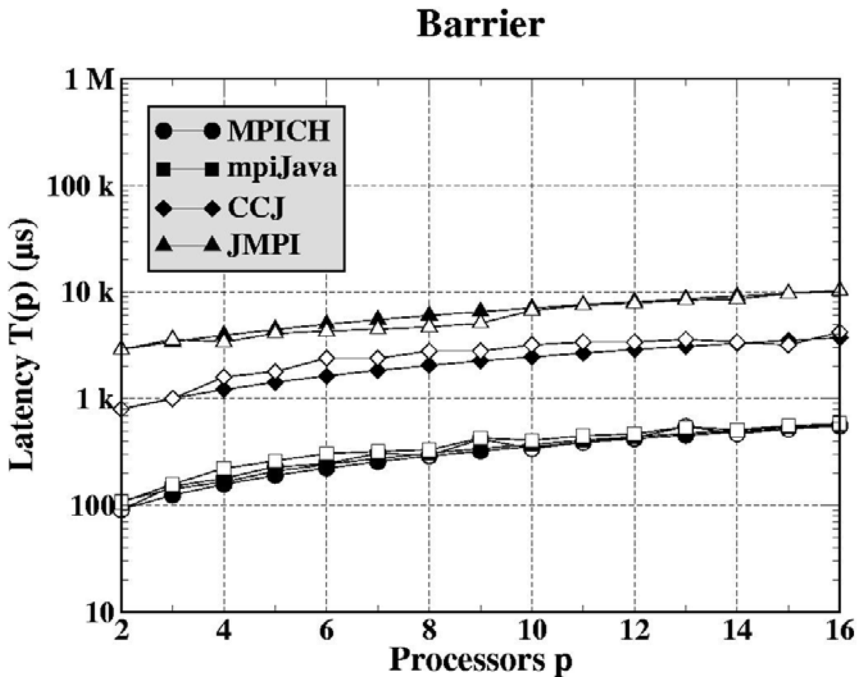
Library	Scatter Analytical Model		Experimental Results	
	$t_s(p)$ {μs}	$t_b(p)$ {ns/B}	T(16B,8)	Bw(1MB,8)
MPICH-GM	$-7+9p$	$4.321+0.414\lceil lp \rceil$	65	190.26
mpiJava	$42+10p$	$7.223-0.358\lceil lp \rceil$	131	167.95
CCJ	$217+604p$	$19.11+10.38\lceil lp \rceil$	5000	18.26
JMPI	$-8287+6438p$	$46.04+11.66\lceil lp \rceil$	40400	9.23

Library	Allgather Analytical Model		Experimental Results	
	$t_s(p)$ {μs}	$t_b(p)$ {ns/B}	T(16B,8)	Bw(1MB,8)
MPICH-GM	$-10+15p$	$5.308+1.098\lceil lp \rceil$	104	116.48
mpiJava	$30+17p$	$8.560+0.483\lceil lp \rceil$	173	100.63
CCJ	$817+944p$	$13.60+20.79p$	9400	5.25
JMPI	$-8296+7506p$	$-31.16+41.36p$	42400	2.85

Experimental Results

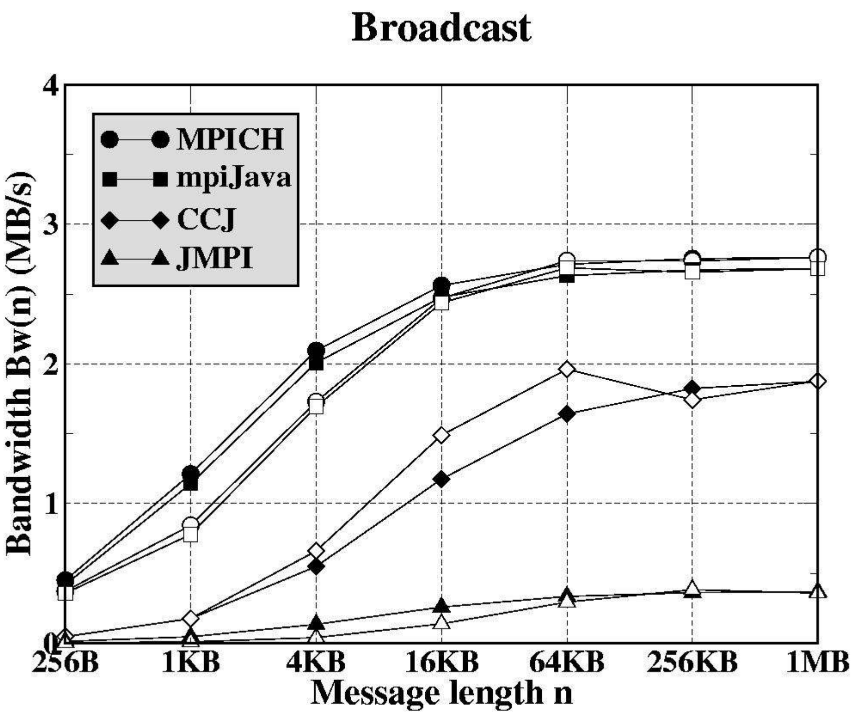
- Fast Ethernet: measured and estimated metrics on 2 and 16 processors

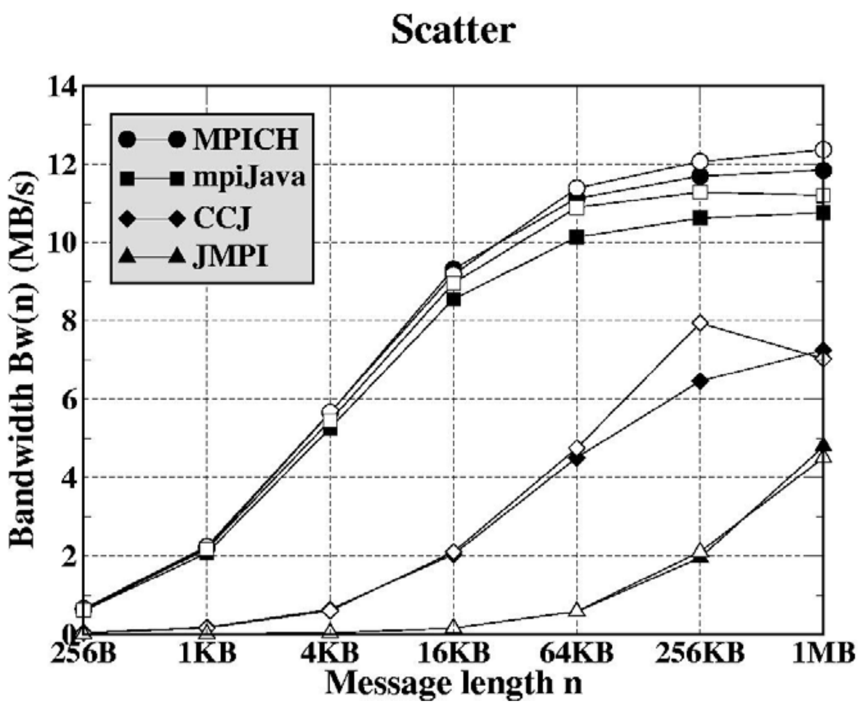




Experimental Results

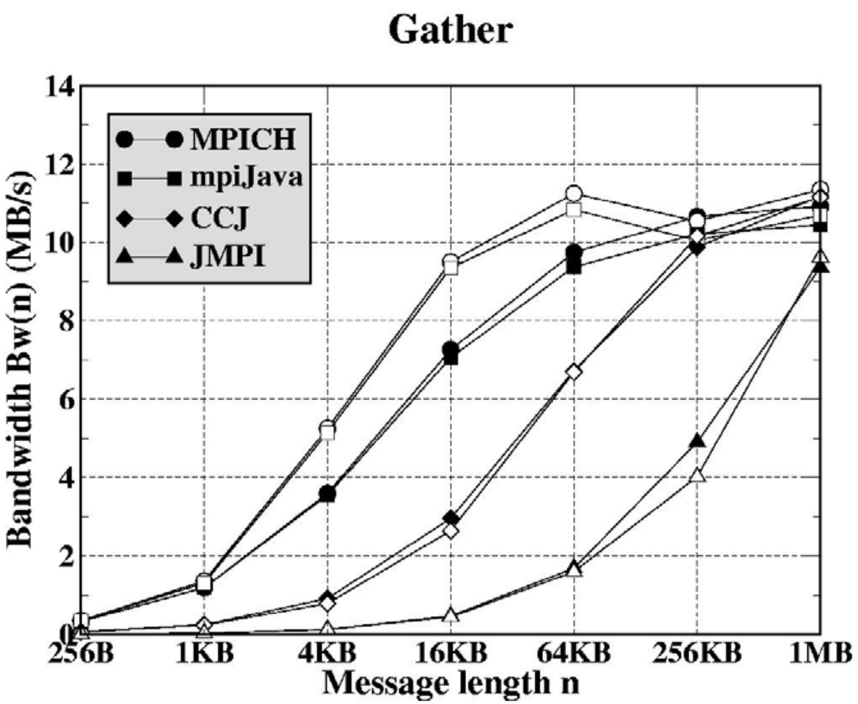
- Fast Ethernet: measured and estimated metrics on 16 processors

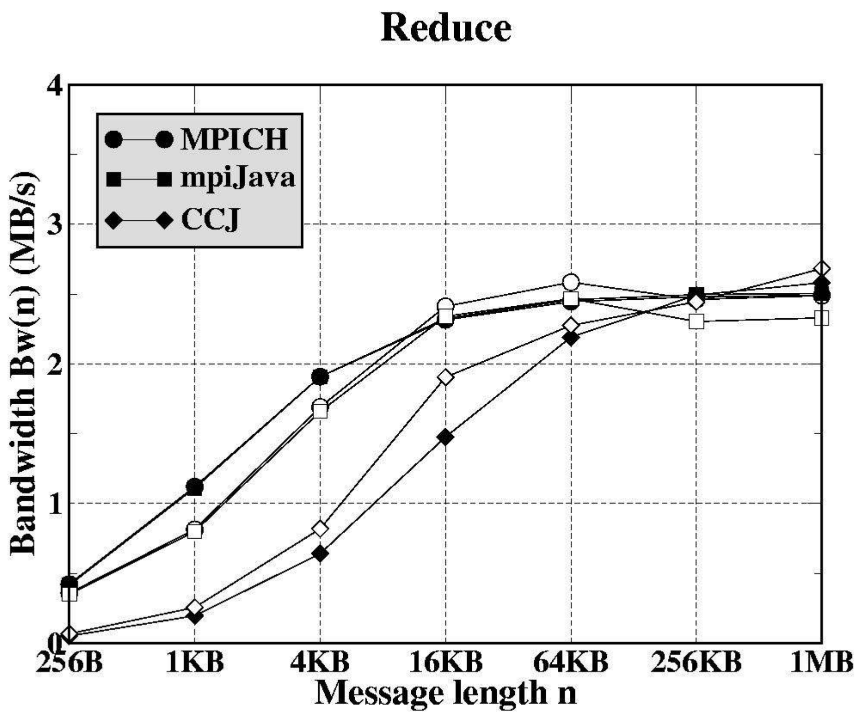




Experimental Results

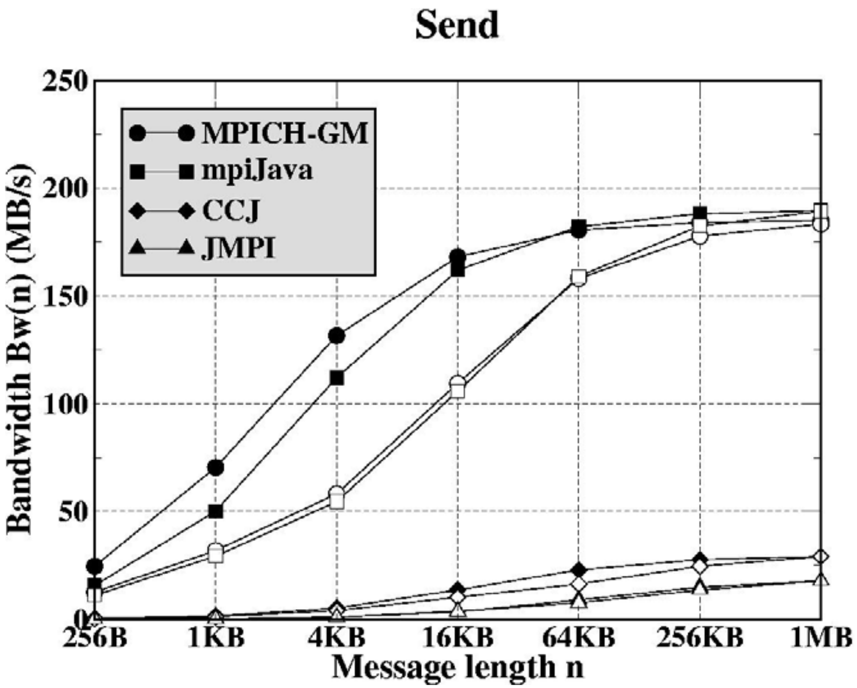
- Fast Ethernet: measured and estimated metrics on 16 processors

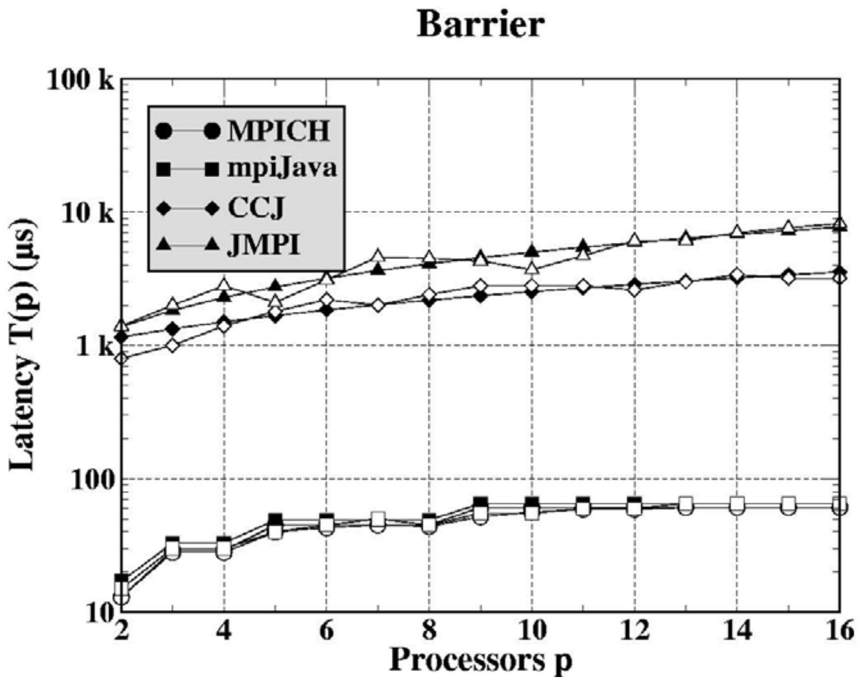




Experimental Results

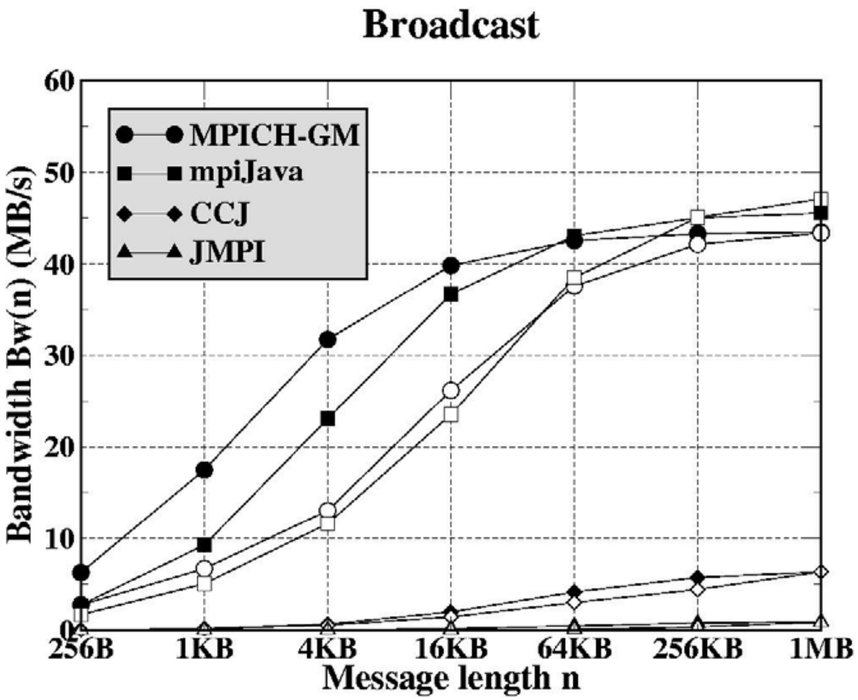
- Myrinet: measured and estimated metrics on 2 and 16 processors

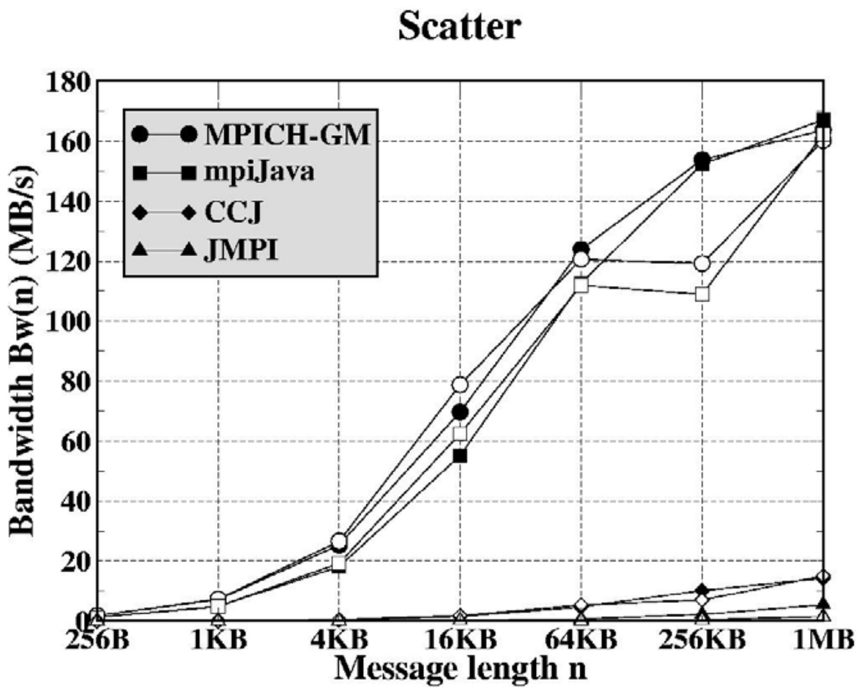




Experimental Results

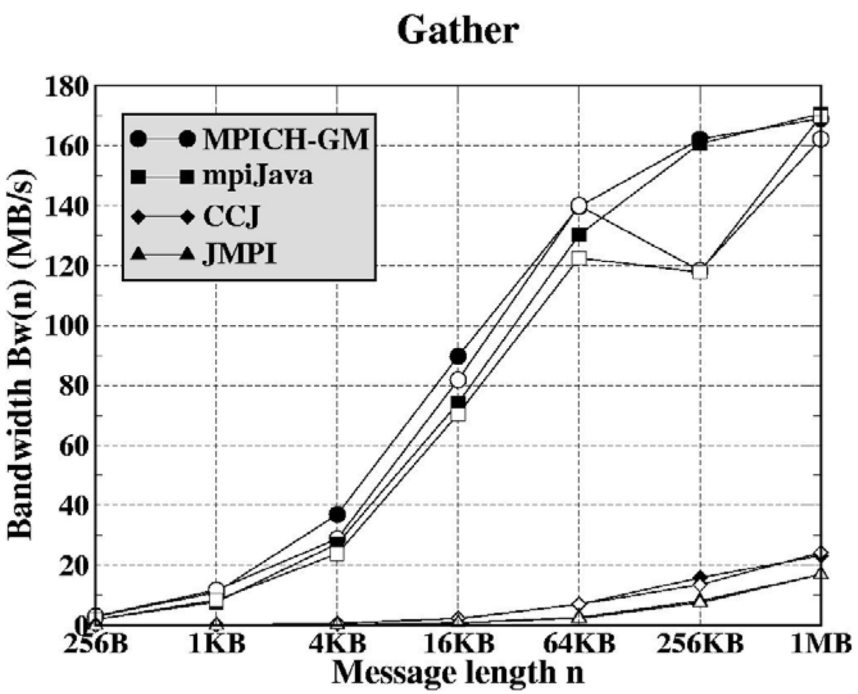
- Myrinet: measured and estimated metrics on 16 processors

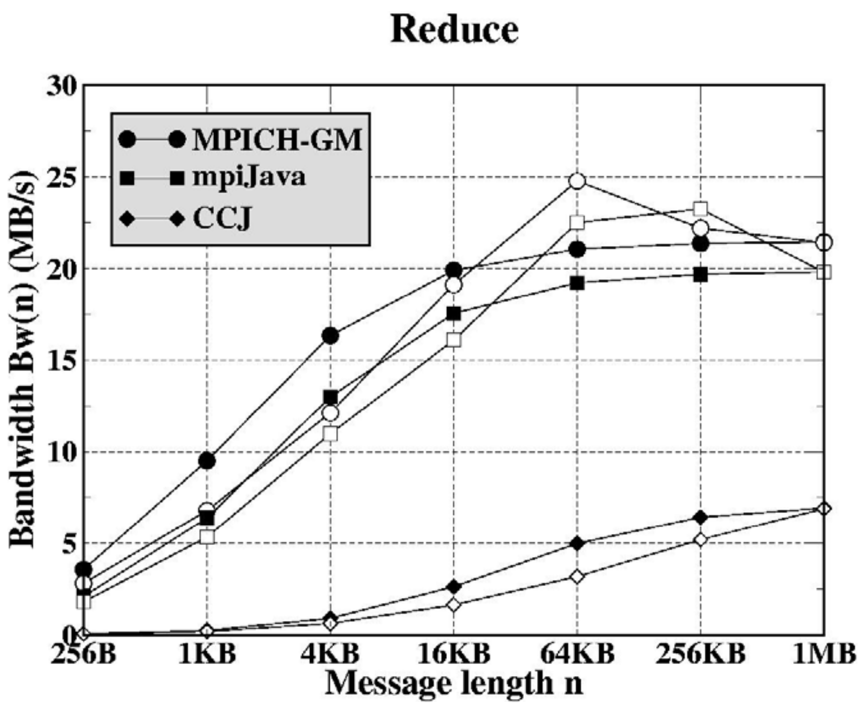




Experimental Results

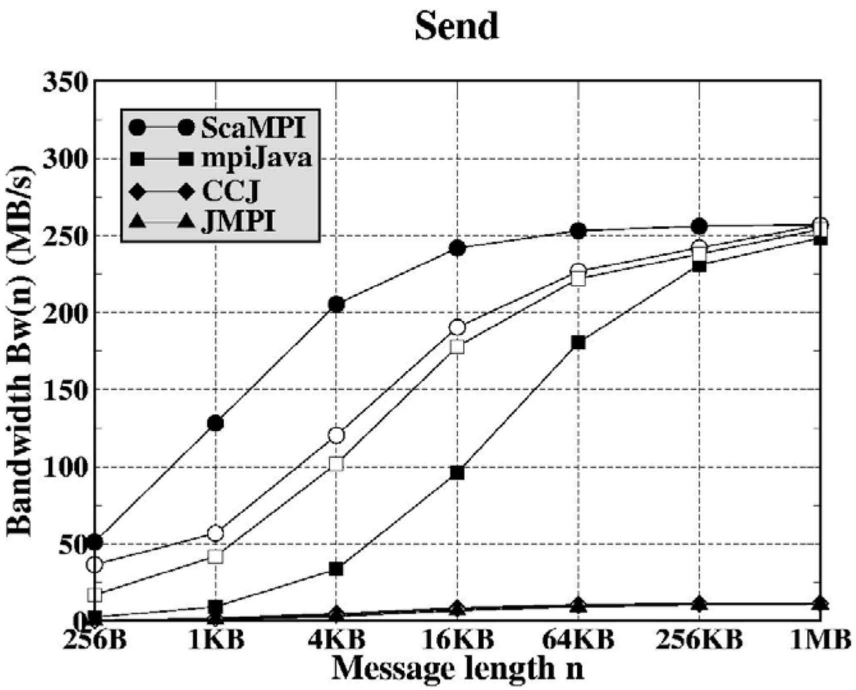
- Myrinet: measured and estimated metrics on 16 processors

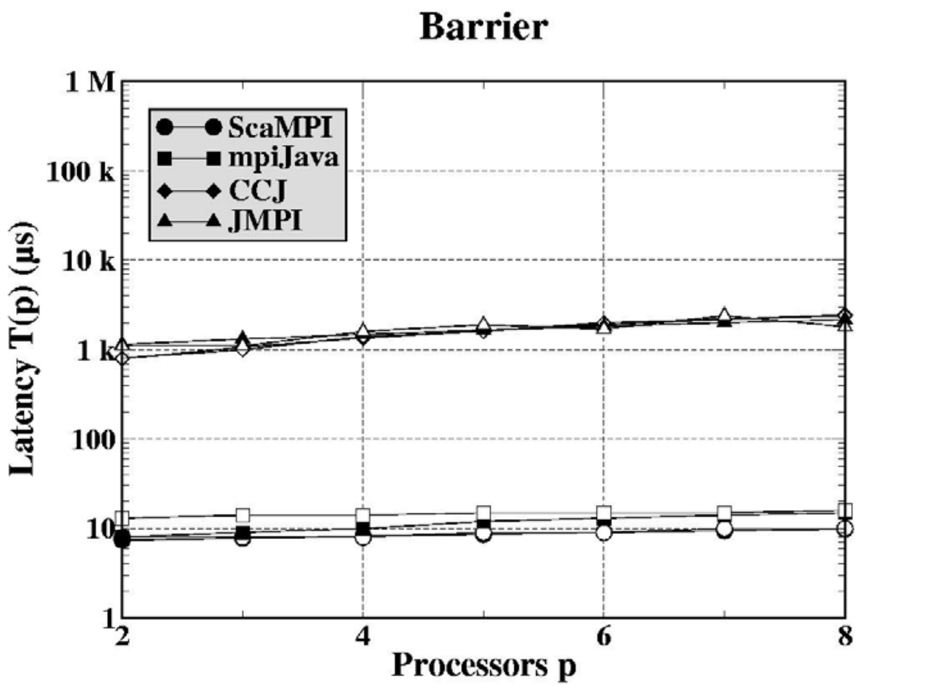




Experimental Results

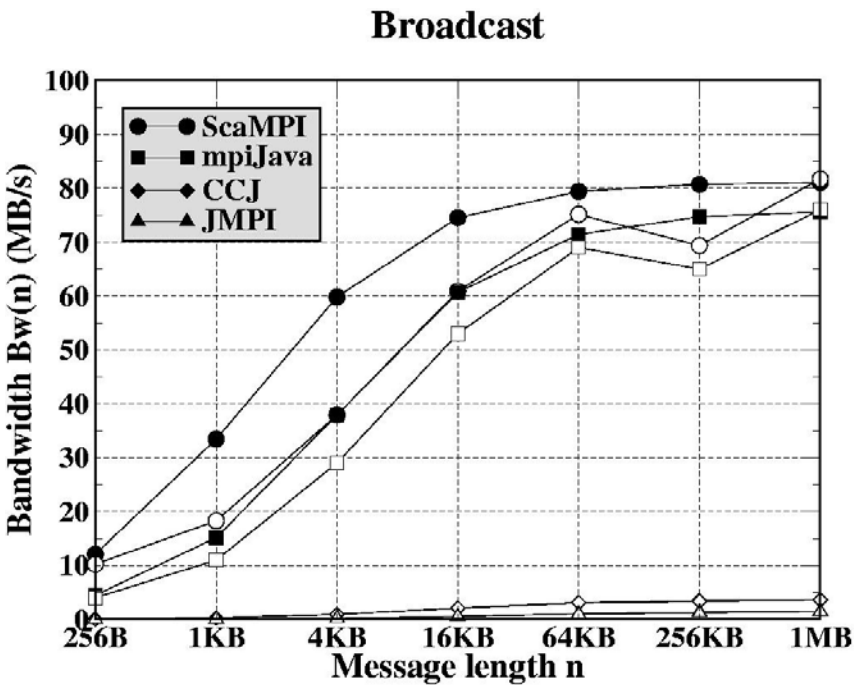
- SCI: measured and estimated metrics on 2 and 8 processors

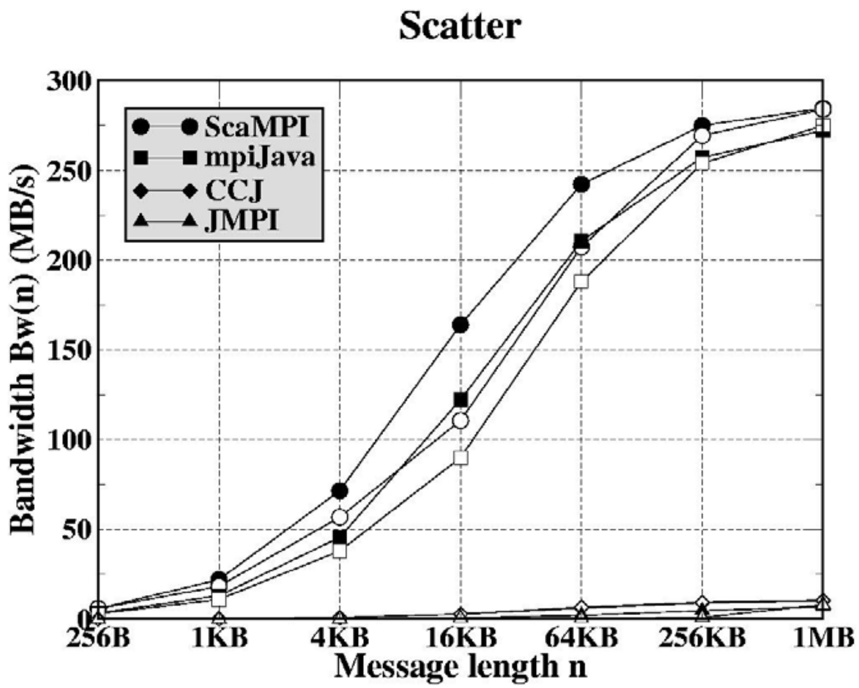




Experimental Results

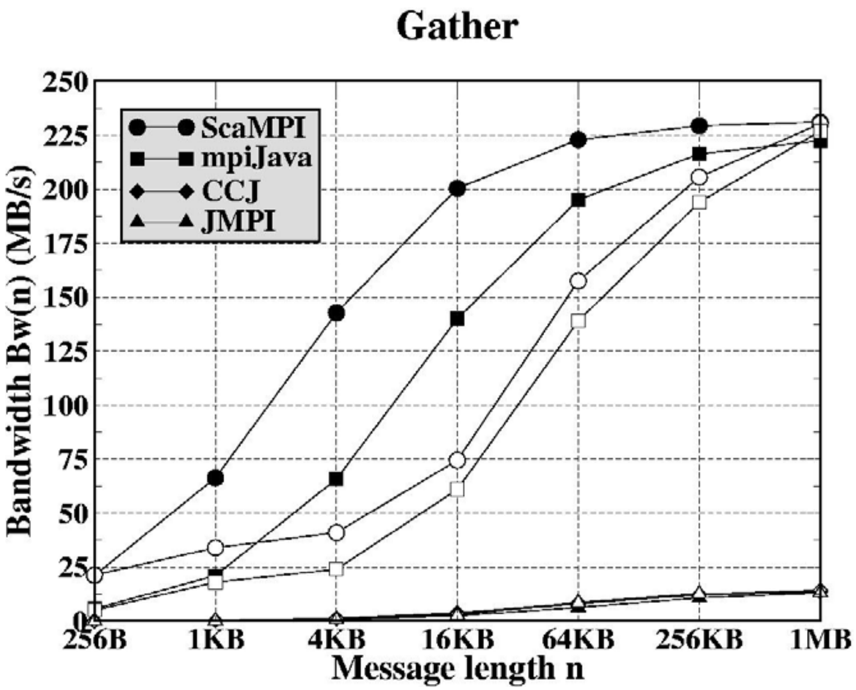
- SCI: measured and estimated metrics on 8 processors

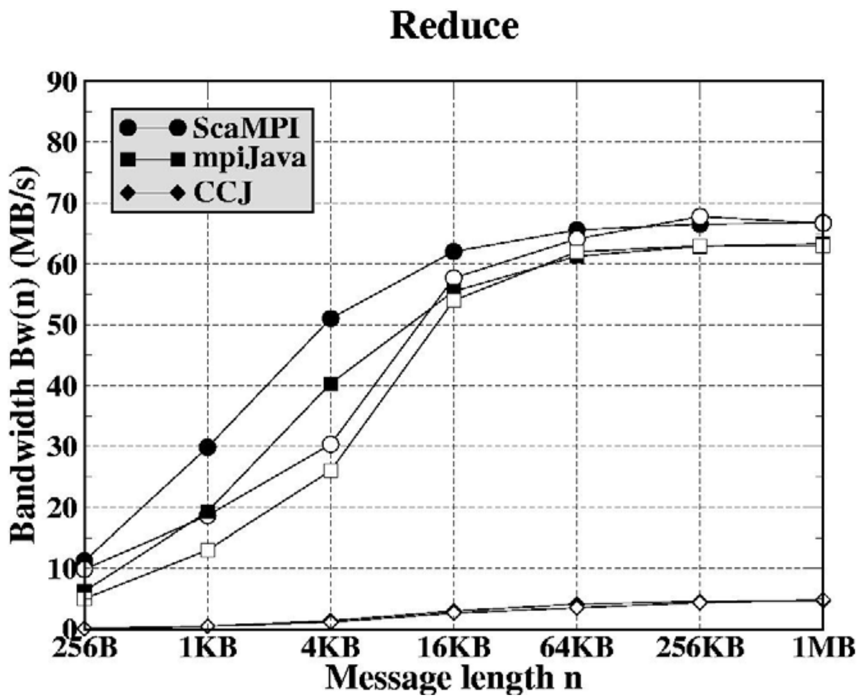




Experimental Results

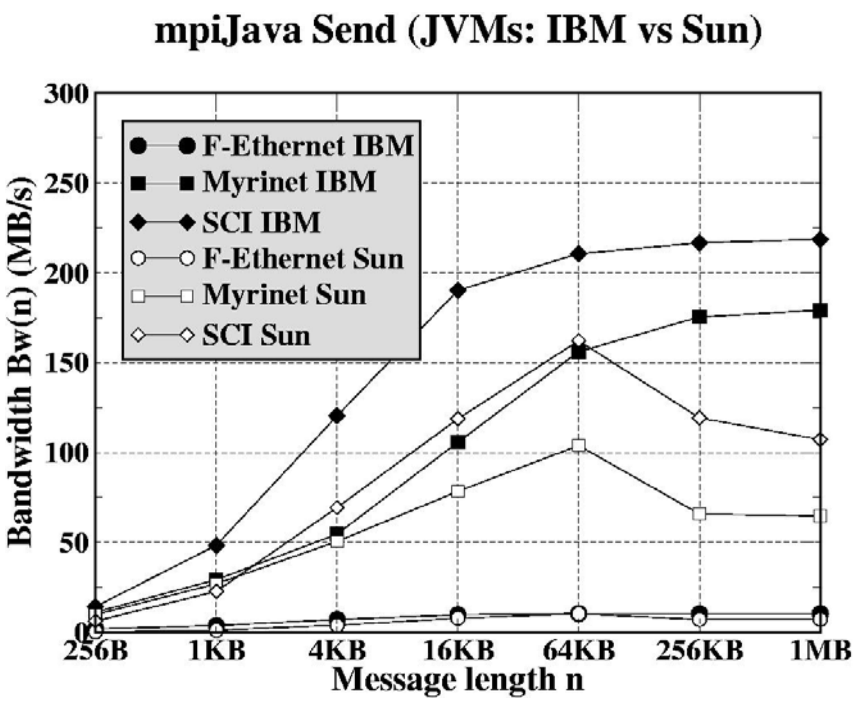
- SCI: measured and estimated metrics on 8 processors

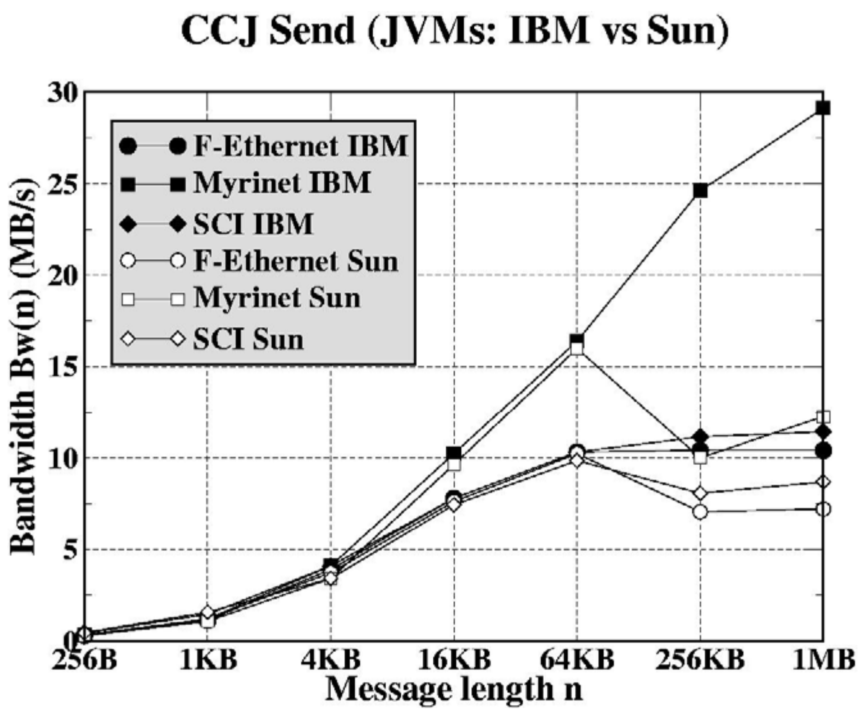




Performance Issues

- IBM JVM presents better results than SUN JVM





Performance Issues

- Design issues. Study of equivalences:
  - Broadcast = Scatter + Allgather
  - Allgather = Gather + Broadcast
  - Allreduce = Reduce + Broadcast
  - Reducescatter = Reduce + Scatter

Conclusions

- Characterization of message-passing communication overhead is an important issue in emerging environments
- As Java message-passing is an emerging option in cluster computing, this kind of studies serve as objective and quantitative guidelines for cluster parallel programming
- Java wrapper presents a good performance (mpiJava calls to native MPI have low overhead), although it is not a truly portable library
- Pure Java implementations show poorer performance (particularly JMPI), mainly for short messages, due to RMI overhead

Conclusions

- Native and wrapper libraries obtain good results in low latency networks
- Pure Java libraries need IP emulation to work on low latency clusters. The startup is about 15% better on IP over SCI whereas the transfer time is three times faster on IP over Myrinet than over SCI
- Performance results achieved from IP emulation are similar of those achieved on a Fast Ethernet cluster

Future Work

- Research efforts should concentrate on minimizing RMI overhead to consolidate and enhance the use of pure Java message-passing codes
- This is the topic of our future work, the optimization of RMI protocol, specially on low latency networks (Myrinet and SCI)

References

- [1] Stankovic, N., Zhang, K. An Evaluation of Java Implementations of Message-Passing. Software-Practice and Experience 30(7) (2000) 741-763
- [2] W. Getov, P. Gray, V. Sunderam. MPI and Java-MPI: Contrasts and Comparisons of Low-Level Communications Performance. In Proc. of Supercomputing Conference, SC'99, Portland, OR (1999)

Contact

- Guillermo L. Taboada
  taboada@udc.es

- Computer Architecture Group
  Department of Electronics and Systems
  University of A Coruña

- http://www.des.udc.es/~gltaboada

Performance Analysis of Java Message-Passing Libraries on Fast Ethernet, Myrinet and SCI Clusters

Guillermo L. Taboada, Juan Touriño and Ramon Doallo
{taboada,juan,doallo}@udc.es

Dept. of Electronics and Systems
University of A Coruña, Spain

IEEE Cluster 2003 – Hong Kong, DECEMBER 2003