MPJ/Ibis, a Flexible and Efficient Message Passing Platform for Java

Rob van Nieuwpoort
rob@cs.vu.nl

MPI

- Message Passing Interface
- Language independent specification
- Language bindings
  - C, C++, Fortran, ...
- High performance
- Available for many platforms
- Widely used

MPI operations

```
MPI_Send(buf, BUFSIZE, MPI_CHAR, dest, TAG,
         MPI_COMM_WORLD);

MPI_Recv(buf, BUFSIZE, MPI_CHAR, from, TAG,
         MPI_COMM_WORLD, &status);
```

- Point-to-point
  - Send / receive (only explicit!)
  - Synchronous / asynchronous
- Collective operations
  - broadcast, reduce, scatter, gather, ...
- Closed world

MPI bindings for Java

- Many Java/MPI bindings:
  - JavaMPI, JMPI, MPIJ, CCJ, etc.
- MPJ: Proposed by the Java Grande Forum
  - A Java language binding for MPI 1.1
  - Developed benchmark suite
- Implementations:
  - MPIJava, built on top of native MPI library
  - MPJ/Ibis, built on top of Ibis

MPJ

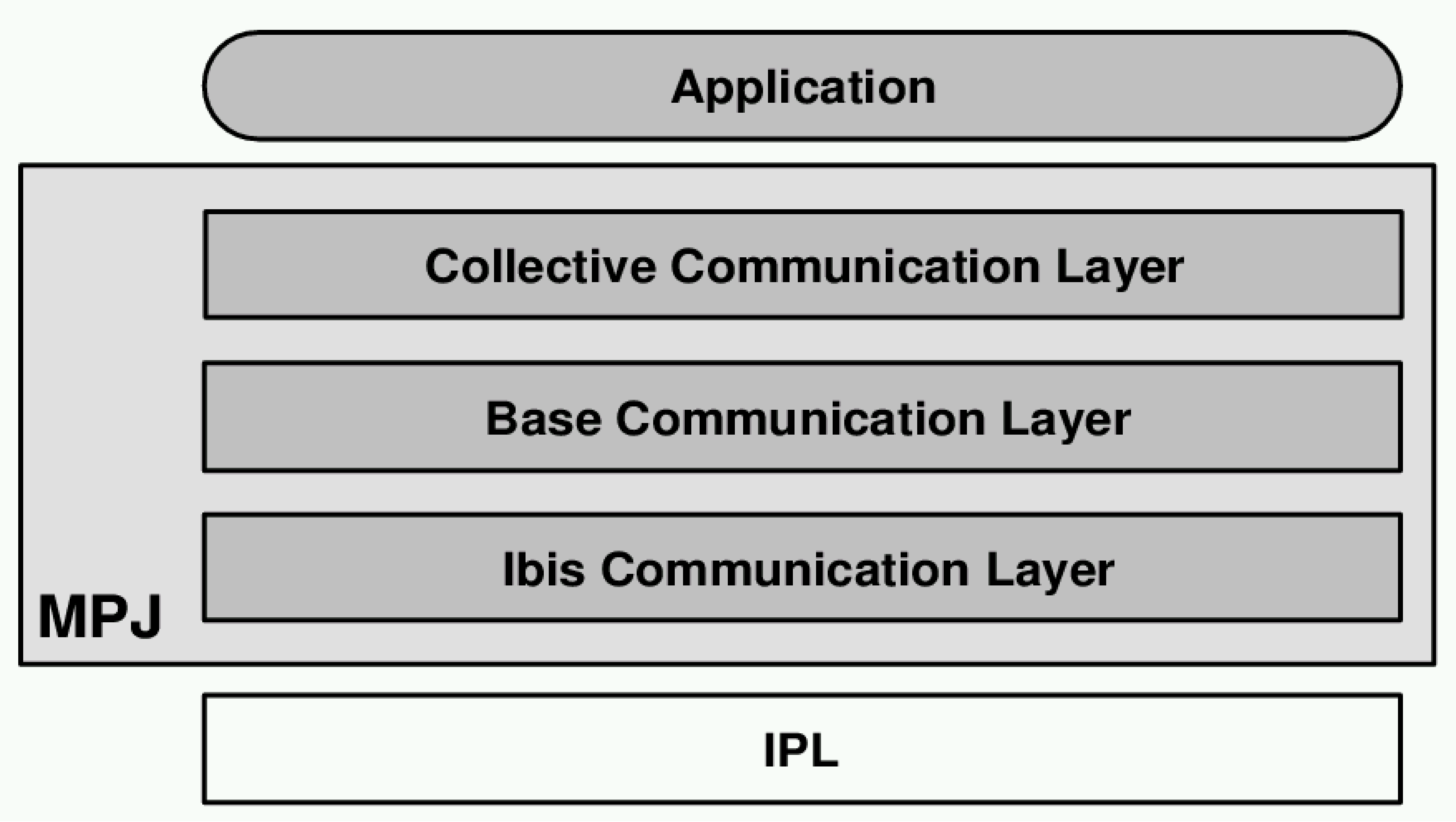
```
void Comm.send(Object buf, int offset, int count,
               Datatype type, int dest, int tag)
               throws MPJException
```

- No status objects, but exceptions
- Separate versions for primitive types
- Parameter “buf” can be
  - Array of a primitive type
  - Array of objects
    - Multidimensional arrays
    - Arbitrarily complex data structure -> object serialization

MPJ/Ibis

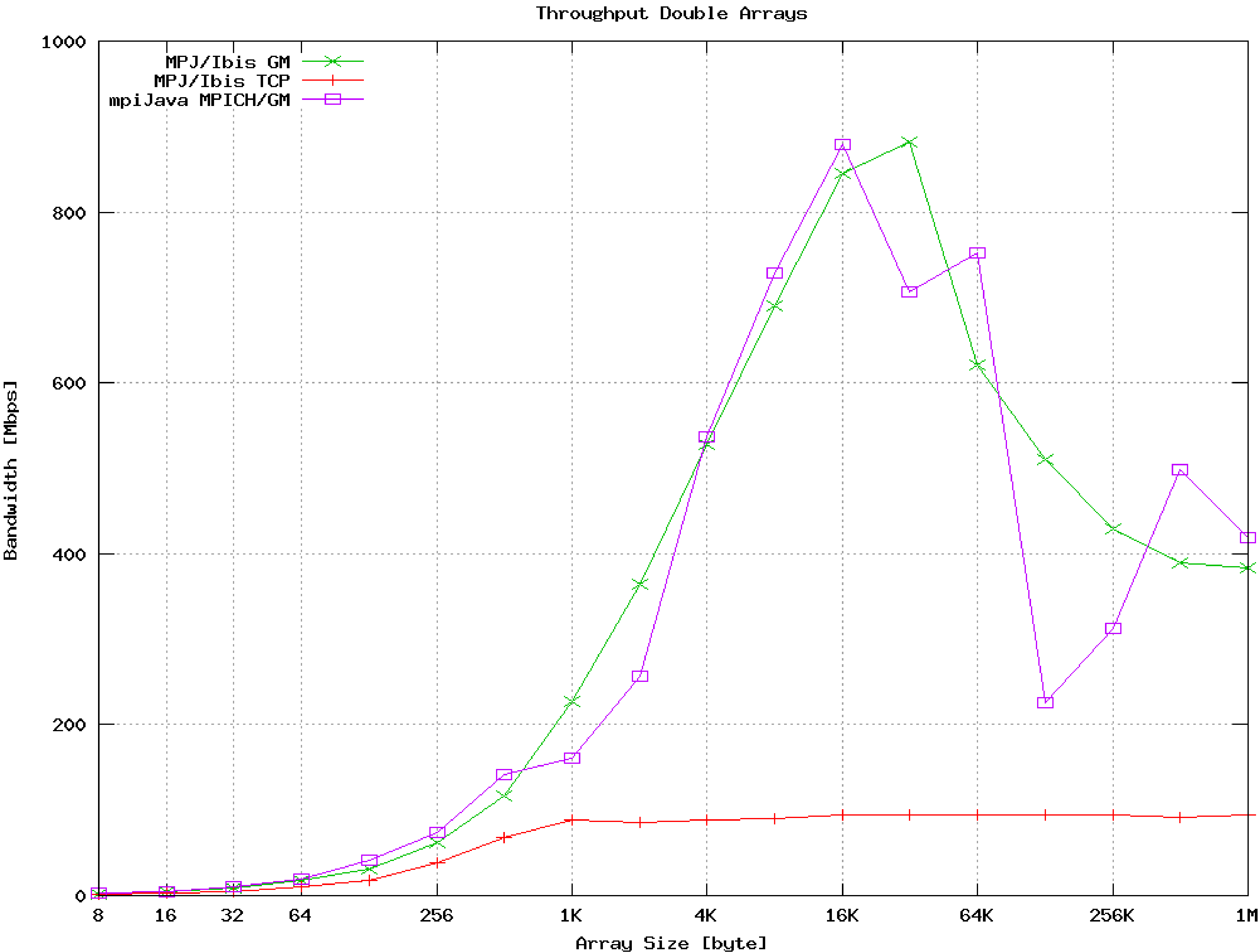
- First 100% Java MPJ implementation
- Uses Ibis IPL for communication
- Ibis provides highly efficient object serialization
- Special grid connectivity support in Ibis
  - Heterogeneous networks
  - Communicate through firewalls
- Very portable, ideal for grid computing

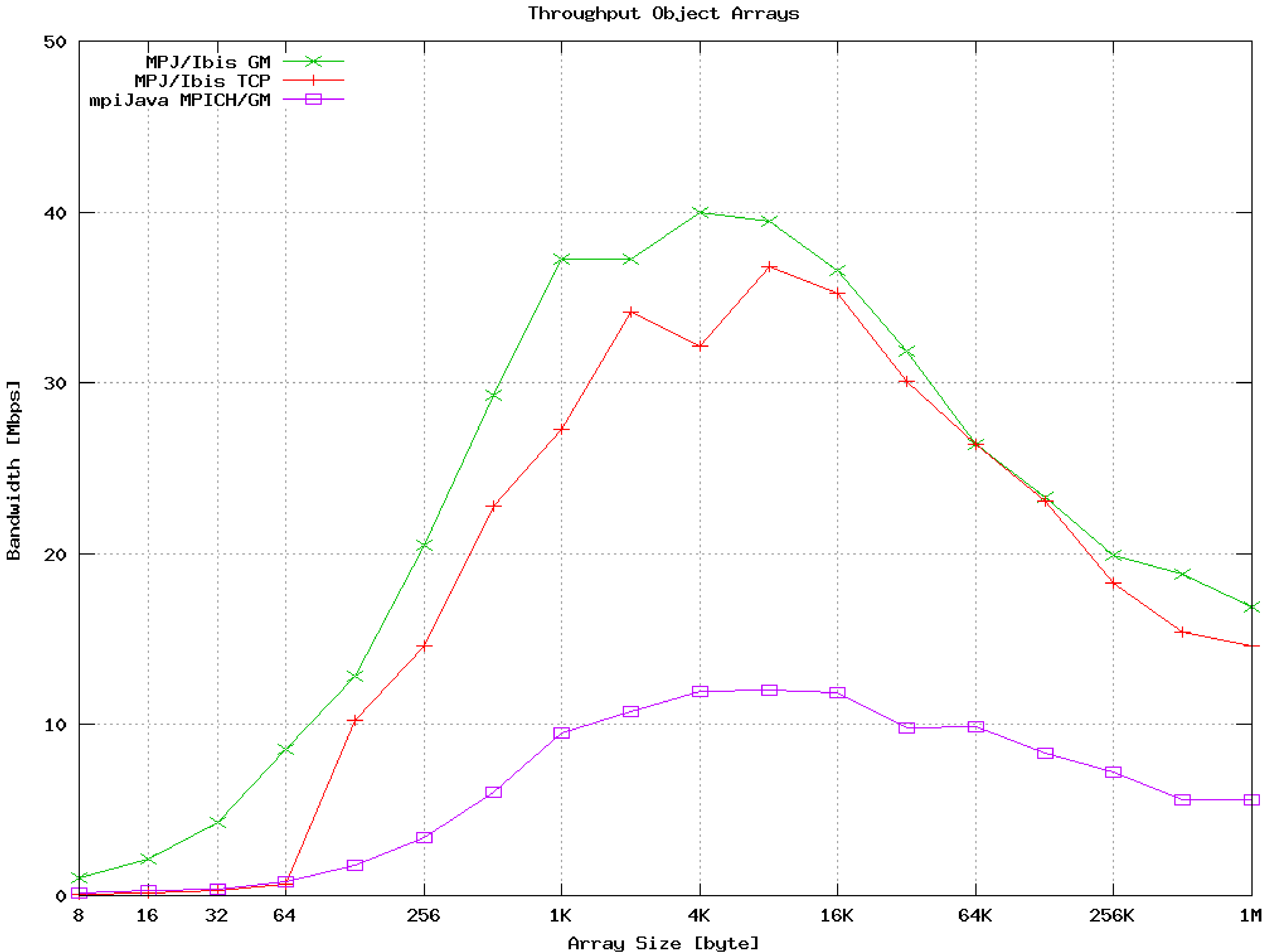
MPJ/Ibis structure

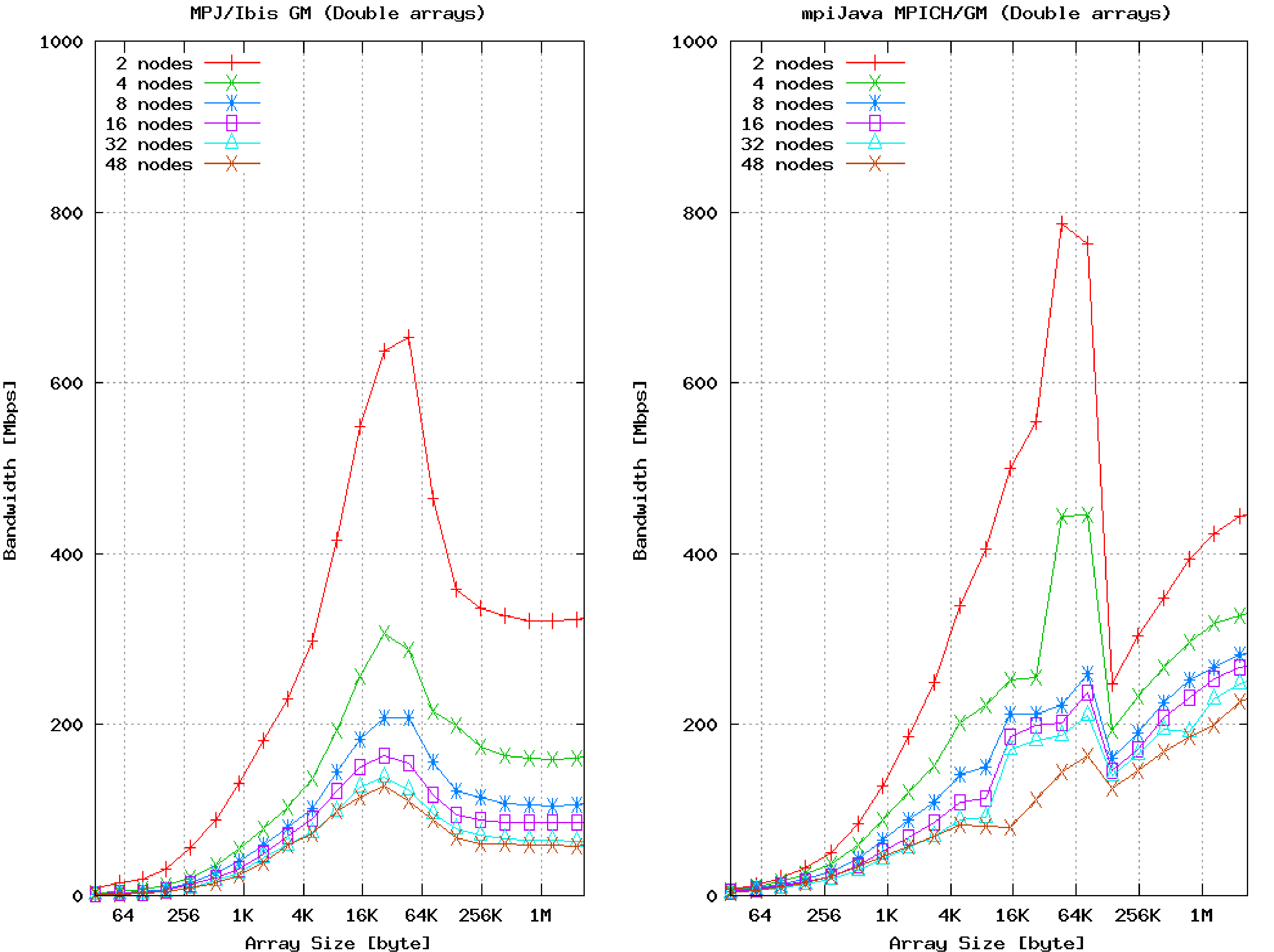


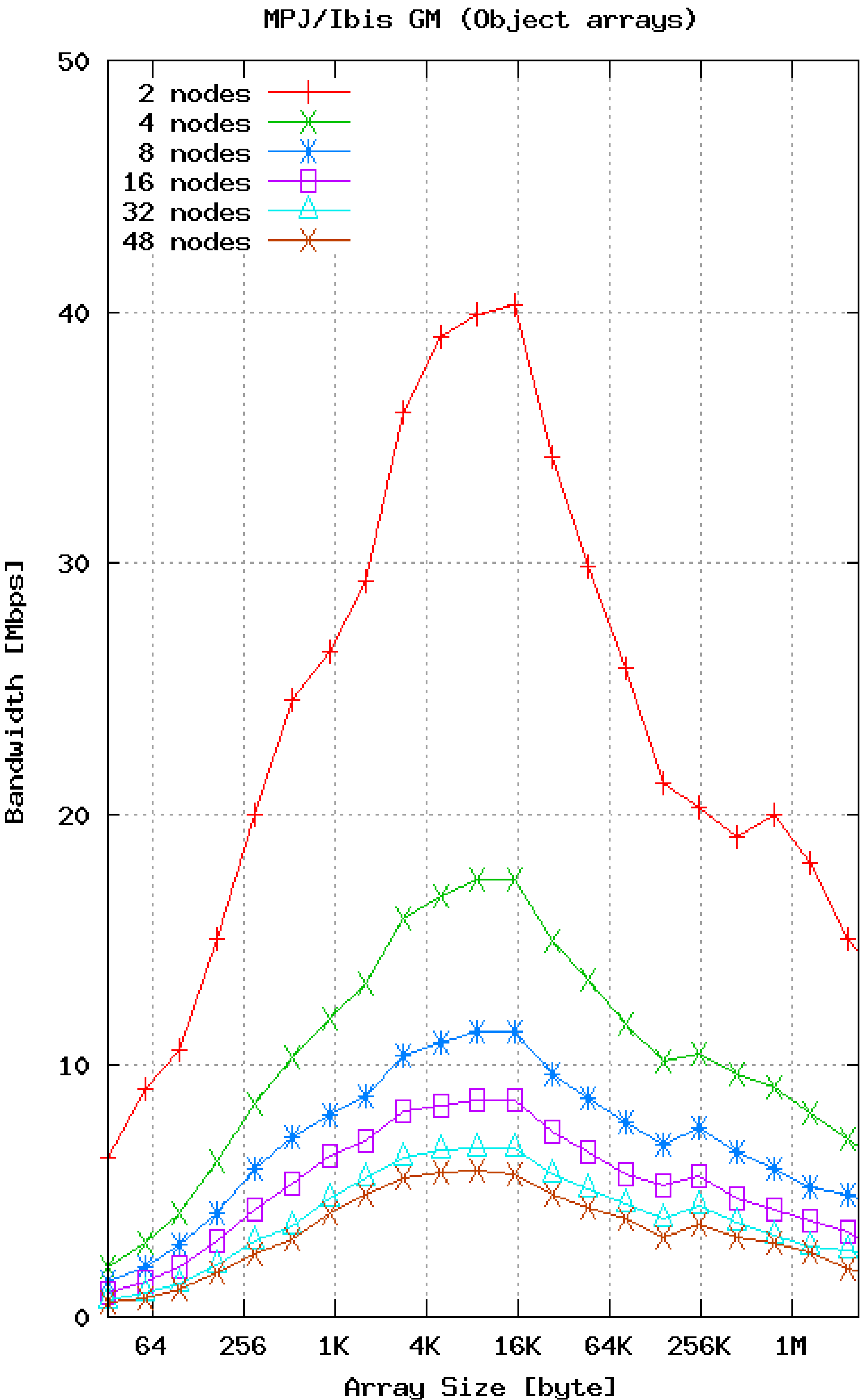
MPJ/Ibis latency P-III 1 GHz

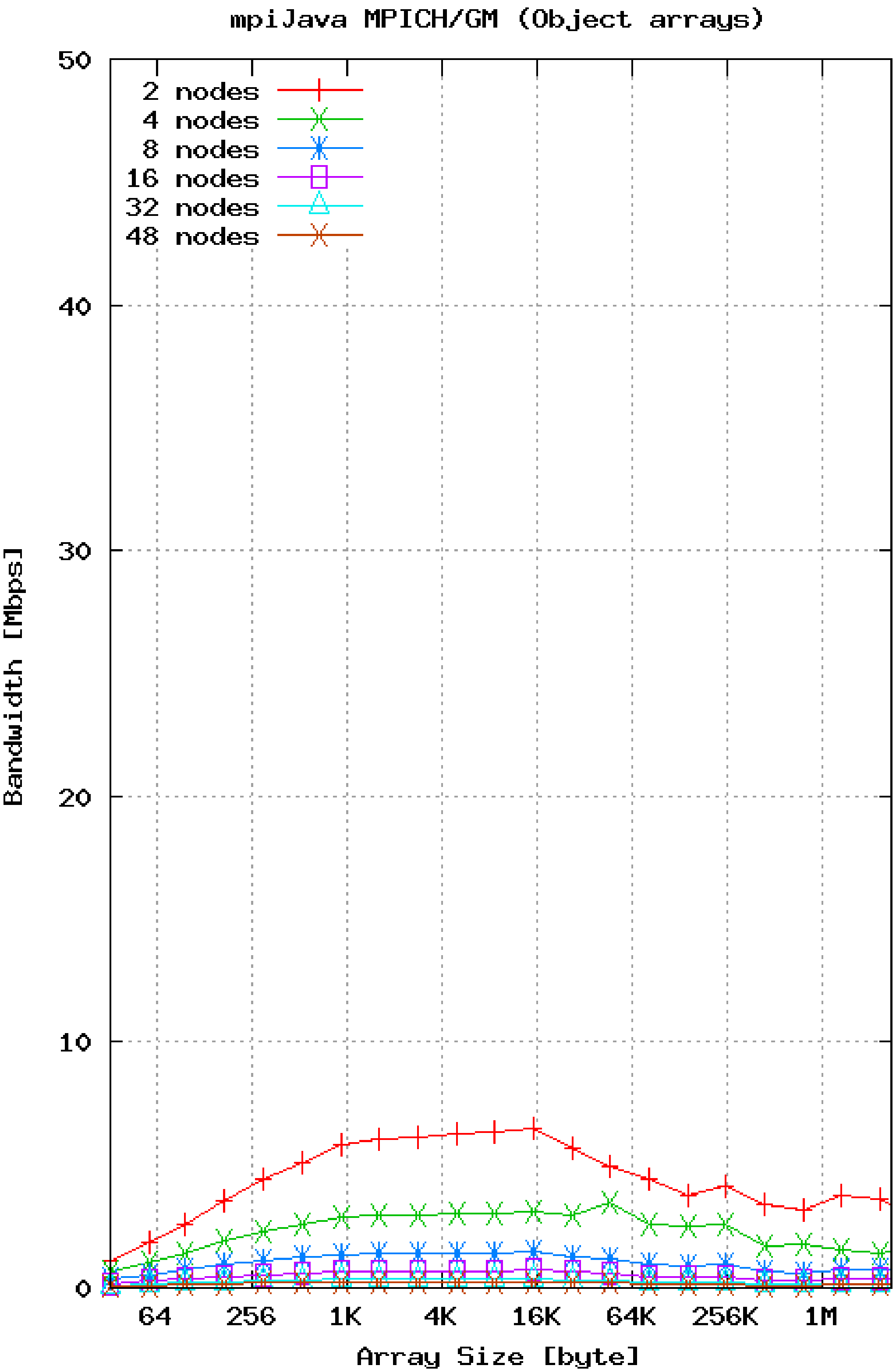
	round-trip
Implementation	latency (us)
MYRINET	
mpiJava (MPICH 1.2.6/GM)	28
Ibis (GM)	44
MPJ/Ibis (GM)	50
FAST ETHERNET	
Ibis (TCP)	113
MPJ/Ibis (TCP)	120

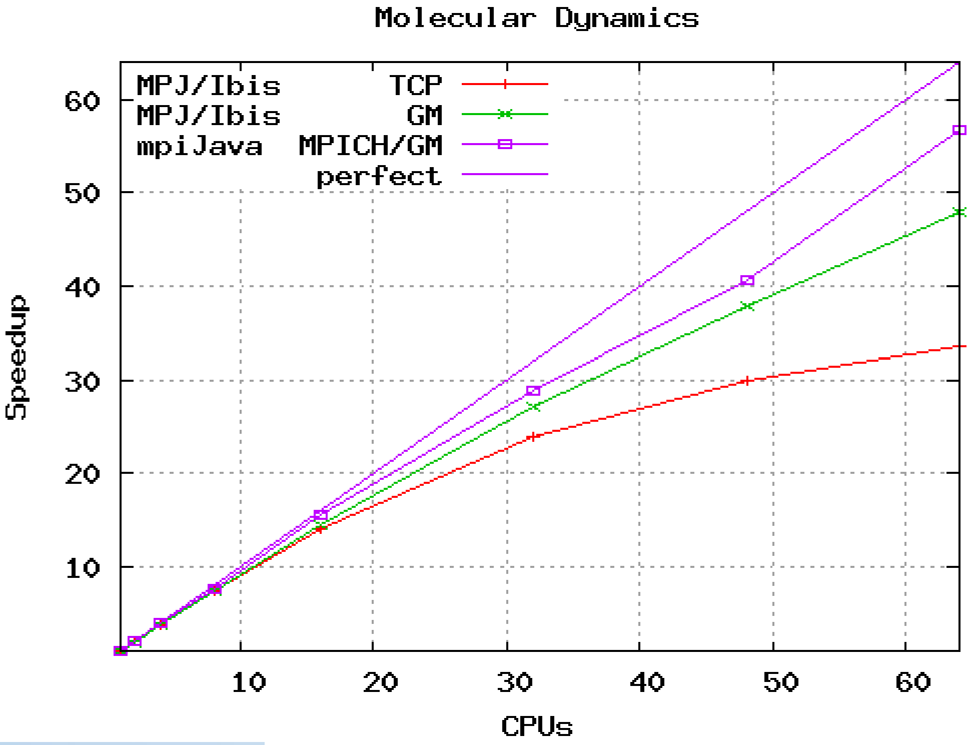

Throughput Double Arrays
MPJ/Ibis GM
MPJ/Ibis TCP
mpiJava MPICH/GM
Bandwidth [Mbps]
1000
800
600
400
200
0
8
16
32
64
256
1K
4K
16K
64K
256K
1M
Array Size [byte]


Throughput Object Arrays
MPJ/Ibis GM
MPJ/Ibis TCP
mpiJava MPICH/GM
Bandwidth [Mbps]
0
10
20
30
40
50
8
16
32
64
256
1K
4K
16K
64K
256K
1M
Array Size [byte]

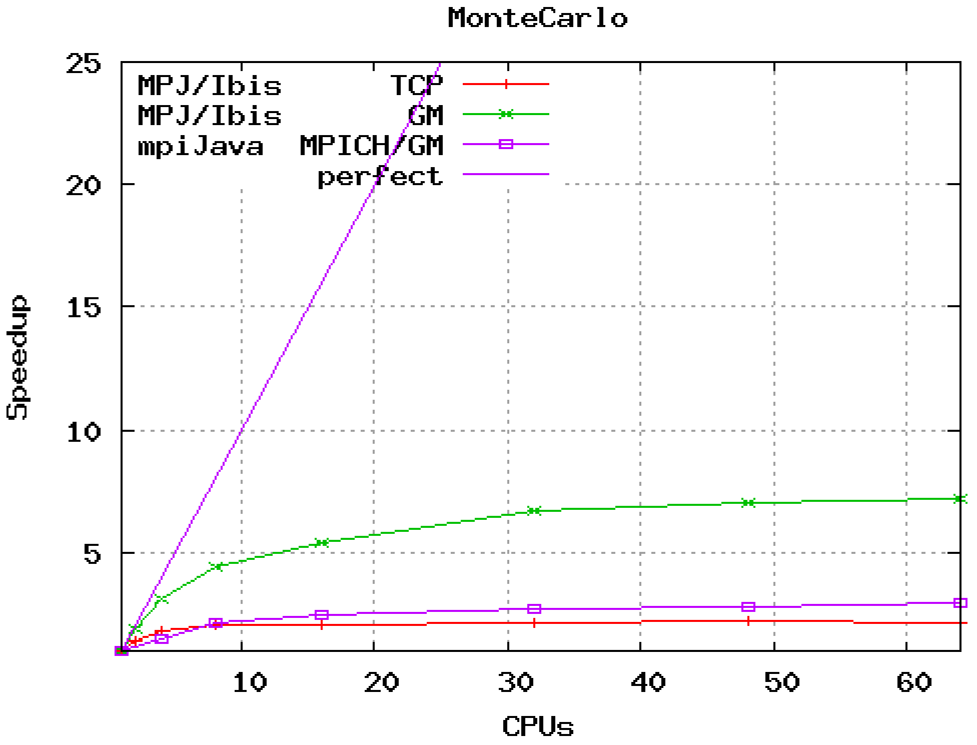

MPJ/Ibis GM (Double arrays)
mpiJava MPICH/GM (Double arrays)
2 nodes
4 nodes
8 nodes
16 nodes
32 nodes
48 nodes
Bandwidth [Mbps]
0
200
400
600
800
1000
Array Size [byte]
64
256
1K
4K
16K
64K
256K
1M


MPJ/Ibis GM (Object arrays)
2 nodes
4 nodes
8 nodes
16 nodes
32 nodes
48 nodes
Bandwidth [Mbps]
0
10
20
30
40
50
64
256
1K
4K
16K
64K
256K
1M
Array Size [byte]

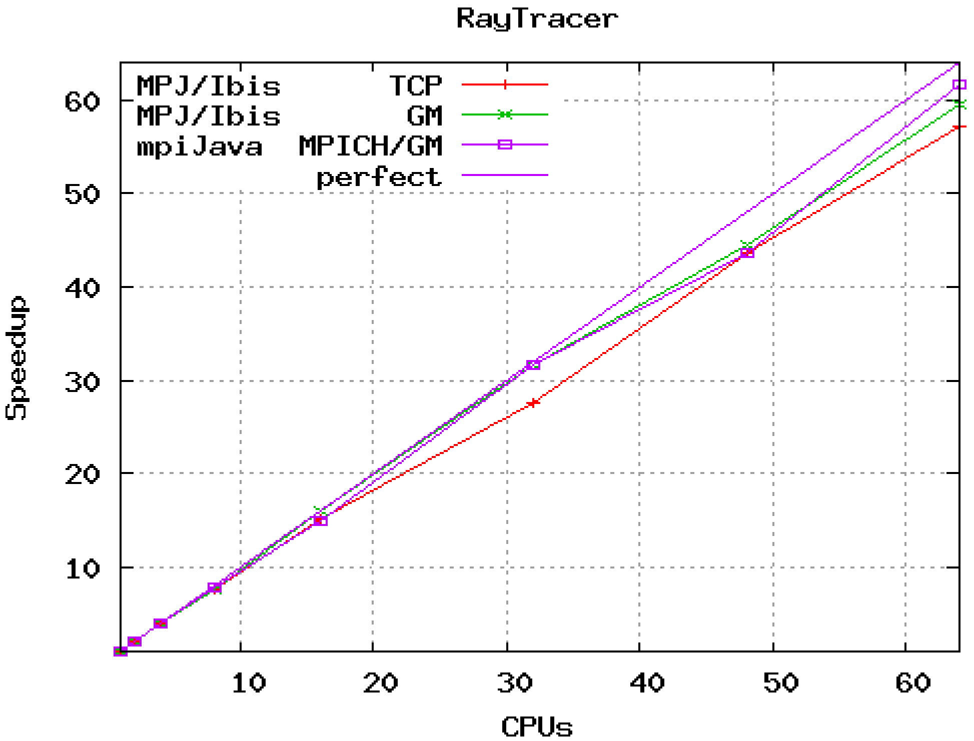

mpiJava MPICH/GM (Object arrays)
2 nodes
4 nodes
8 nodes
16 nodes
32 nodes
48 nodes
Bandwidth [Mbps]
0
10
20
30
40
50
64
256
1K
4K
16K
64K
256K
1M
Array Size [byte]



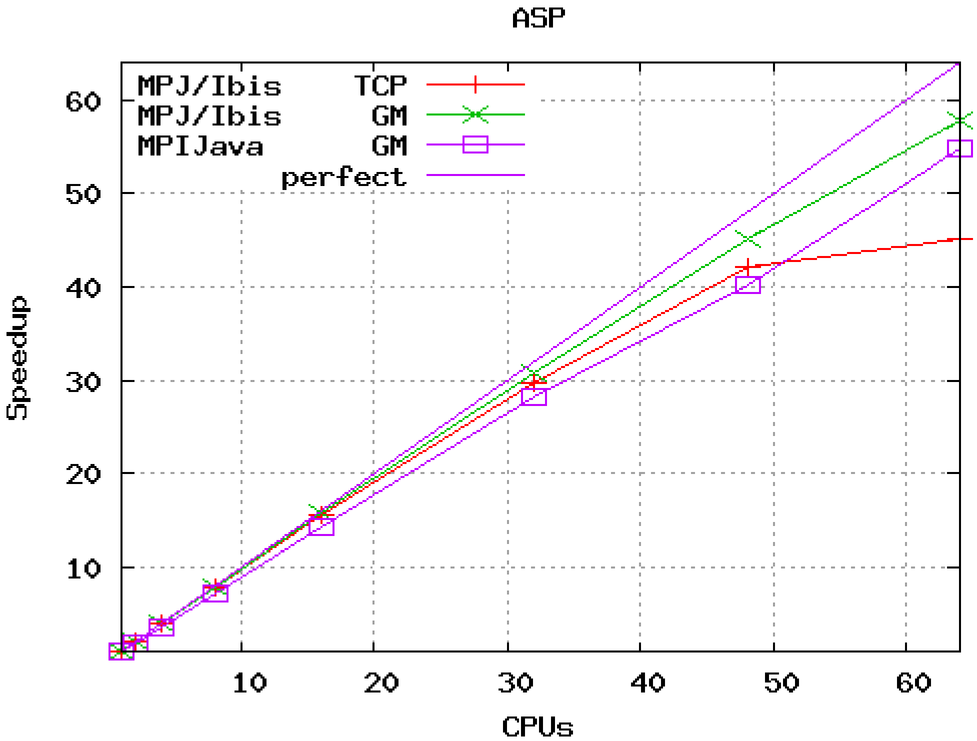
Molecular Dynamics



MonteCarlo



RayTracer

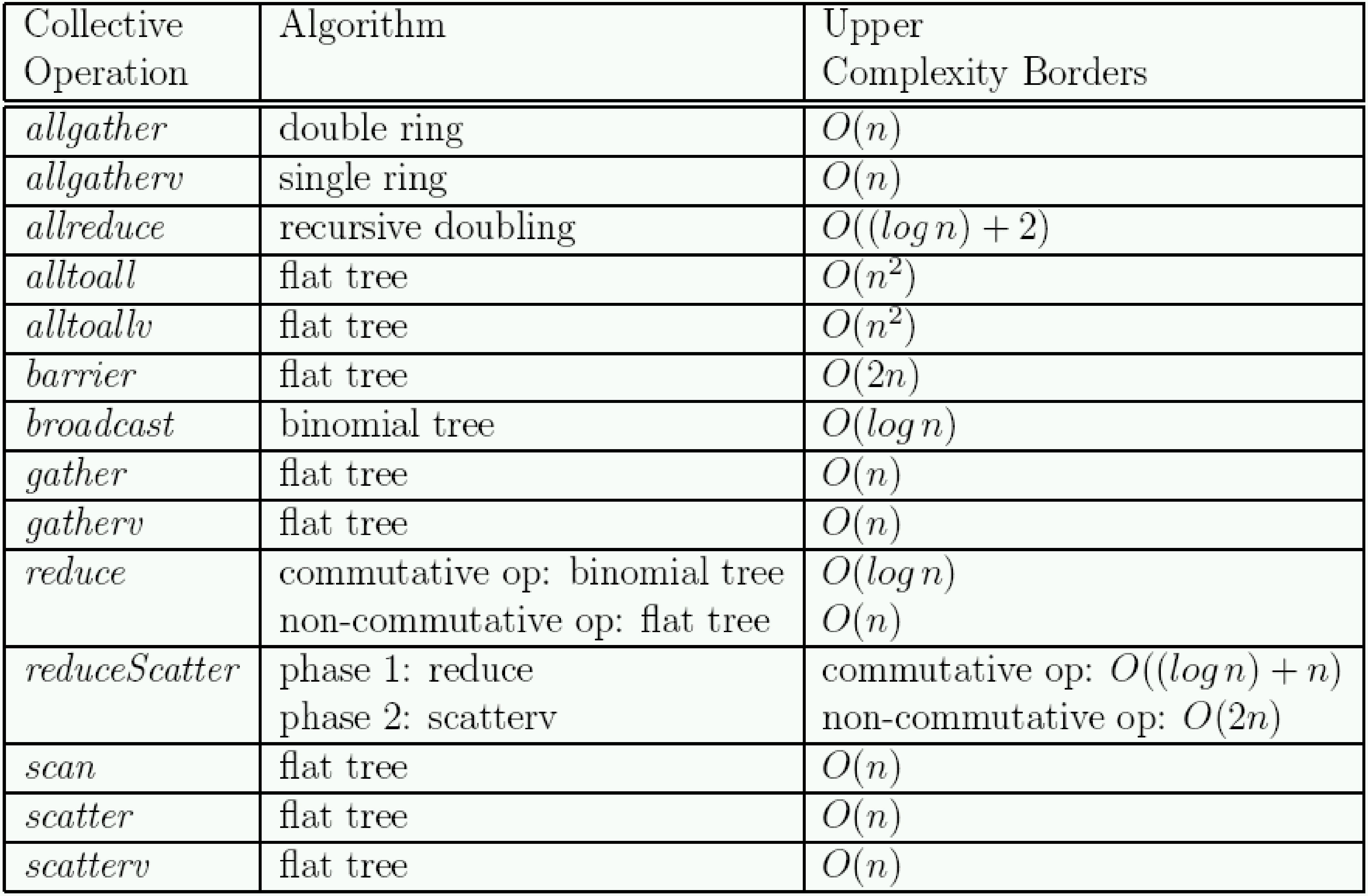


ASP

Conclusions

- Targeted at grid environments
- MPJ/Ibis is extremely flexible
  - “run everywhere”
  - Heterogeneous networks
  - Communicate through Firewalls
- Competitive performance
  - Latency and Collectives are a bit slower than native implementation
  - Object serialization is much faster
  - Application-level performance is similar

MPJ/Ibis collectives



Collective Operation	Algorithm	Upper Complexity Borders
allgather	double ring	$O(n)$
allgatherv	single ring	$O(n)$
allreduce	recursive doubling	$O((log\, n) + 2)$
alltoall	flat tree	$O(n^2)$
alltoallv	flat tree	$O(n^2)$
barrier	flat tree	$O(2n)$
broadcast	binomial tree	$O(log\, n)$
gather	flat tree	$O(n)$
gatherv	flat tree	$O(n)$
reduce	commutative op: binomial tree non-commutative op: flat tree	$O(log\, n)$ $O(n)$
reduceScatter	phase 1: reduce phase 2: scatterv	commutative op: $O((log\, n) + n)$ non-commutative op: $O(2n)$
scan	flat tree	$O(n)$
scatter	flat tree	$O(n)$
scatterv	flat tree	$O(n)$