

---

# File I/O and optimised MPI communication on a large Power4 system

Joachim Hein

Stephen Booth

Elena Breitmoser

Mark Bull

EPCC, The University of Edinburgh



- Overview on the HPCx system
- File I/O
  - Writing directly from the nodes
  - Using MPI2 File I/O
- Optimised Point-to-Point communication
  - Playing Multi-Ping-Pong
  - Sending messages around a ring
  - Improvements to halo swaps in domain decomposition codes
- Discussion

- Funded by the Engineering and Physical Sciences Research Council (EPSRC), UK
- Capability Computing
  - Use significant fraction of resource, e.g. 512+ CPUs
- HPCx consortium won the competitive procurement
  - Technical support by EPCC, Edinburgh and CLRC, Daresbury
  - 3-phase hardware roadmap supplied by IBM
- Service went live on 9<sup>th</sup> December 2002
- Website: [www.hpcx.ac.uk](http://www.hpcx.ac.uk)

- 40 IBM p690 Regatta H frames for compute
  - Per frame: 32 Power4 processors, 1.3 GHz, 1280 processors in total
  - Number 12 on present top 500 list (3.2 Tflops)
  - Frames divided into 4 logical partitions (lpar)
- Presently Colony Switch network
  - Dual plane, omega network, packet based
  - PCI-switch adapter, 2 per lpar (1 per plane)
- 4 dedicated I/O lpars
  - Each: 2 PCI-switch adapt., 4 Fibre channel adapters
  - Running GPFS
- 18 TB of disk space

# The cabinets

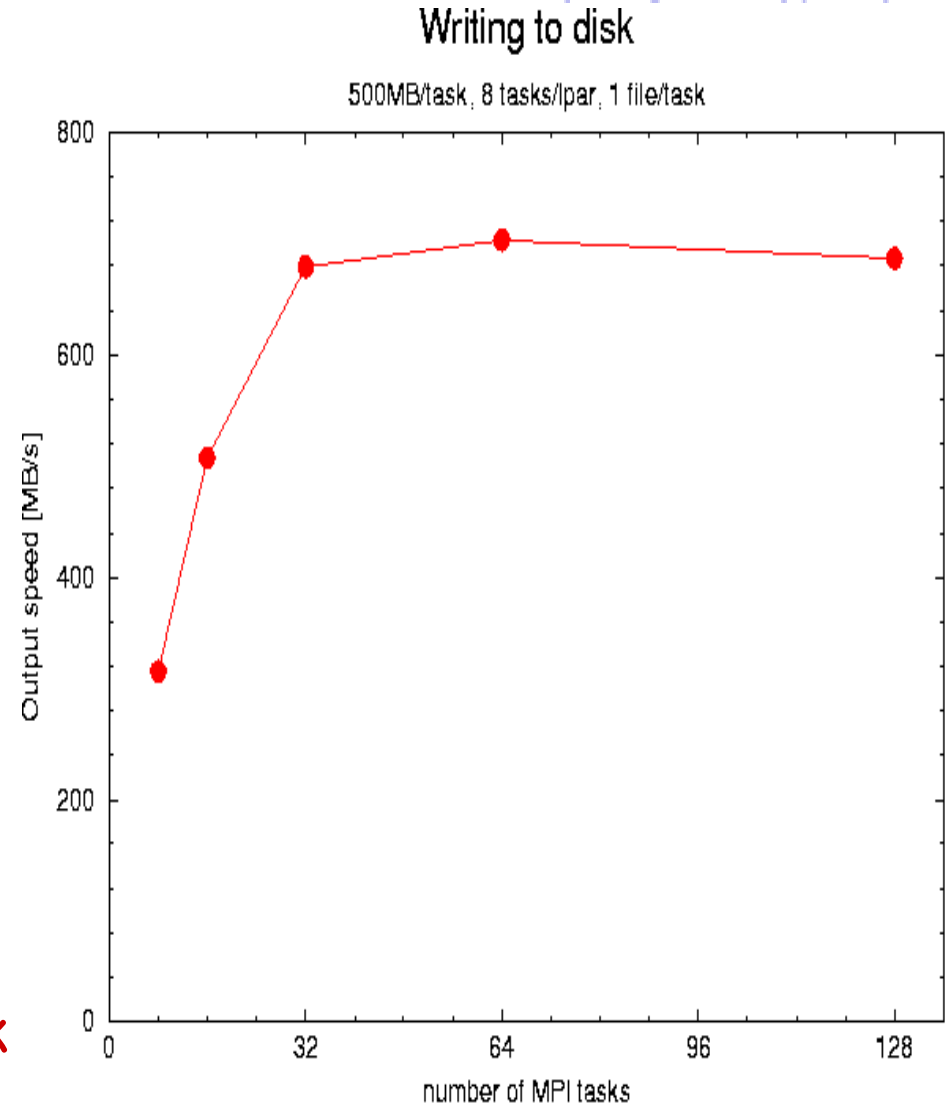
---



- Look at different strategies to read/write data from/to disk system
  - a) Each processor writes its own file
  - b) Each processor writes to a different position in a single random access file
  - c) Each processor reads the same file
  - d) Use MPI File I/O
- Points (a) and (c) are of particular interest for a task farm
- Lpars have 8 GB of main memory for 8 processors, we are writing/reading 500MB per task (half of main memory)

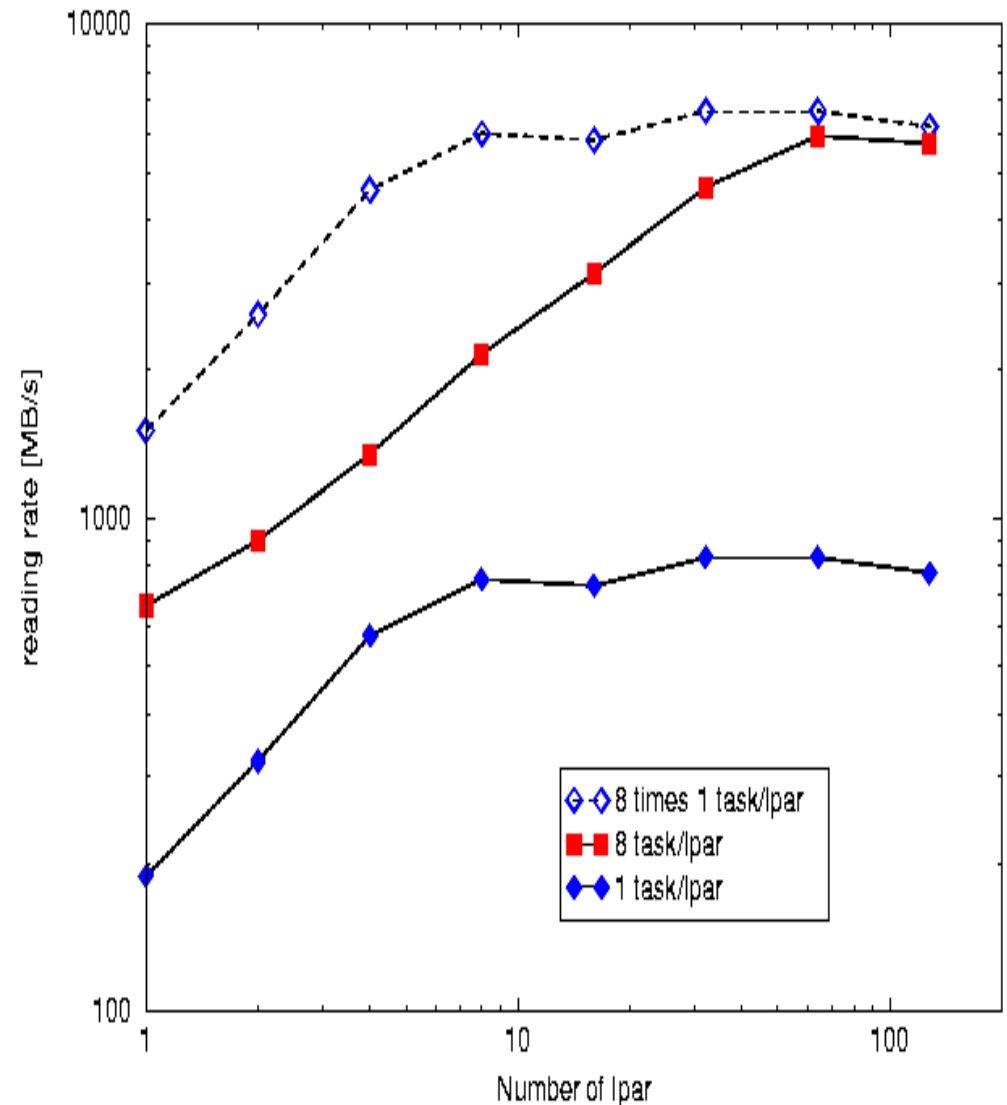
# Each task writes its own file

- **Benchmark:**
  - Write 500 MB/task to file
  - 8 task/lpar
  - Report on best result
- Single lpar rate limited by switch adapter
- Saturates from 4 or more lpars
- Insensitive to tasks/lpar
- Similar picture for
  - Read
  - Random access file
- **Bad:** single lpar accessing disk



# All tasks read the same file

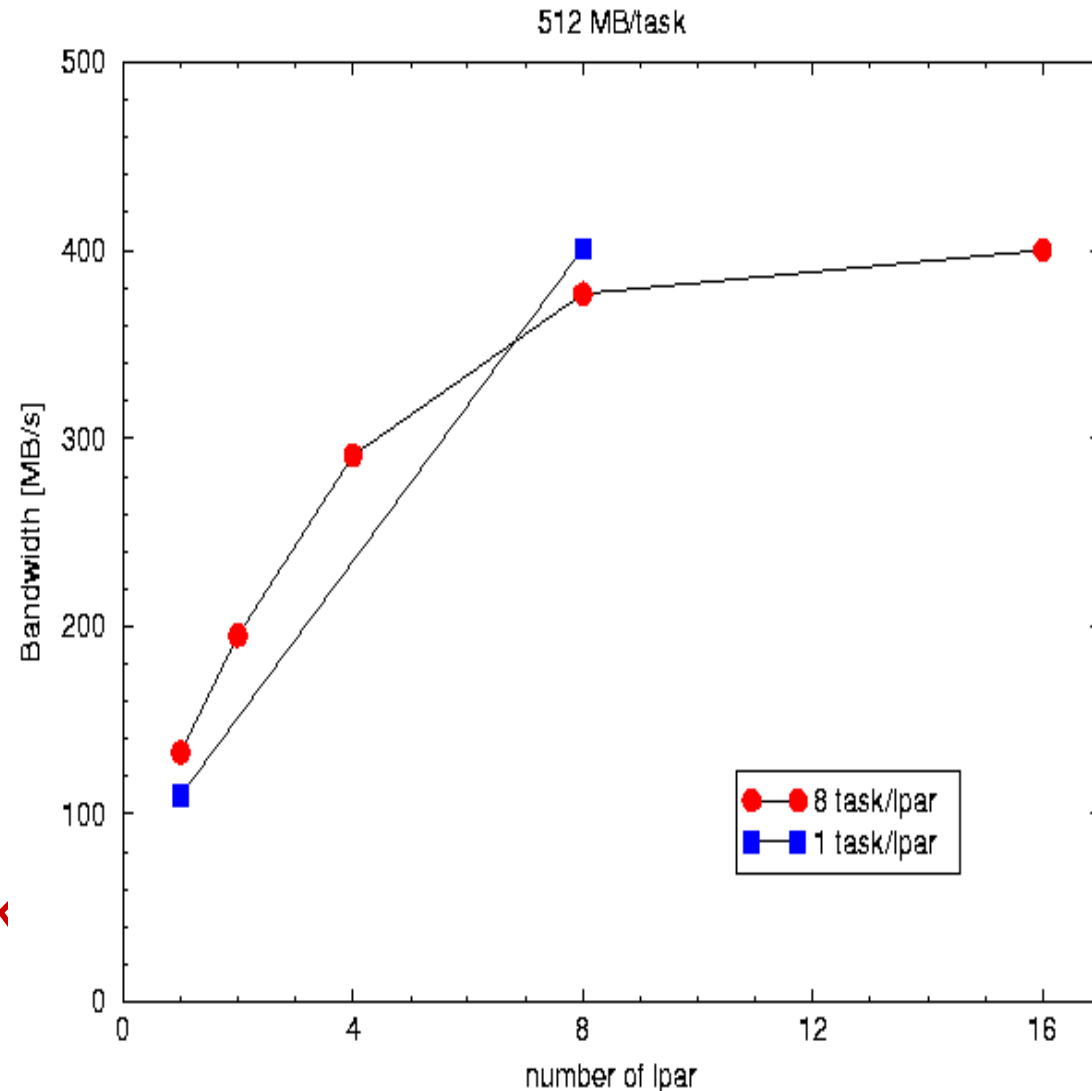
- Particular interest for a task farm
- For 1 task/lpar rate saturates at 800 MB/s for 8 or more lpars
- For 8 task/lpar up to 8 times this rate. Lpars able to cache data internally



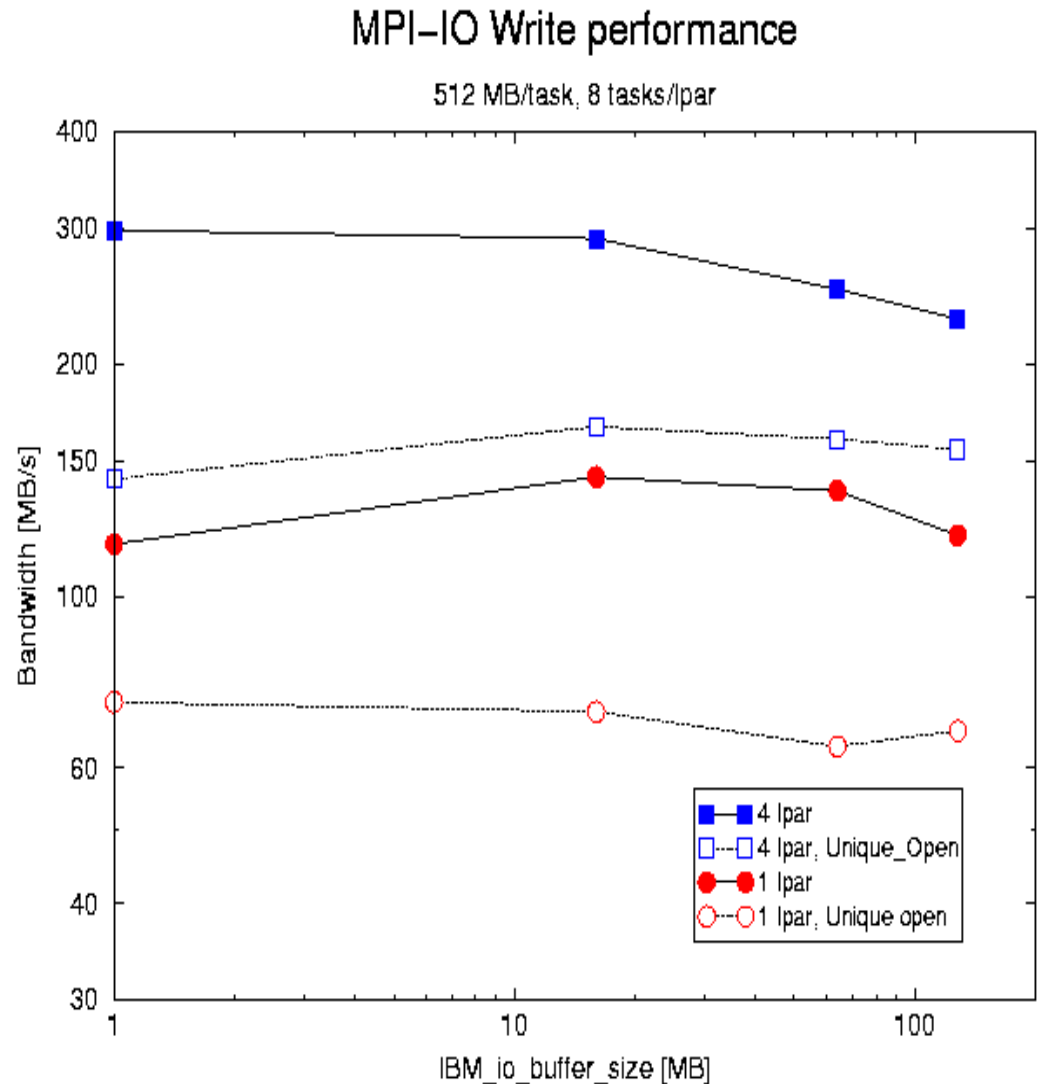
- Simple and convenient way to write/read data from a domain decomposition code
  - Single call
  - Decomposition independent
  - Strip off the halos
- Our benchmark
  - 3 dimensional decomposition of 3 dimensional array
  - Each processor writes (reads) 512 MB of data
  - Use MPI\_File\_write\_all (MPI\_File\_read\_all)
- Full write up available:  
[www.hpcx.ac.uk/research/hpc/technical\\_reports/](http://www.hpcx.ac.uk/research/hpc/technical_reports/)

# MPI2-I/O Write performance

- Writing 512 MB per task
- Saturates at 8 or more lpar
- Independent of task/lpar
- About 2x slower than if each task does own I/O
  - Convenience vs performance
- Performance superior to having single lpar write
- **Master/slave** not a good idea with present network

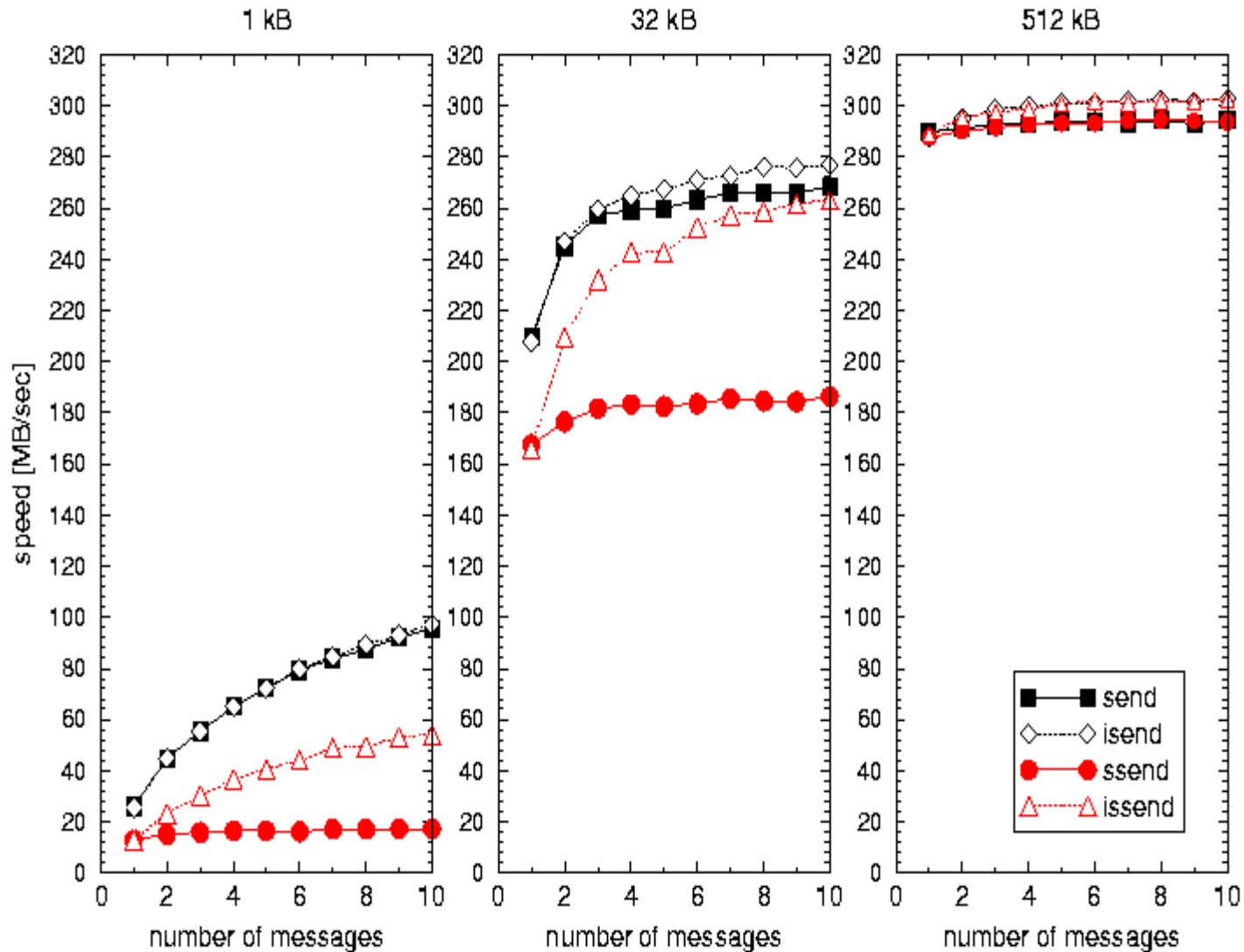


- `IBM_io_buffer_size` has only minor effect
  - The 16 MB default is very reasonable
- `MPI_MODE_UNIQUE_OPEN` has bad effect on write performance (neutral for read)
- `IBM_largeblock_io` **very bad** (not on plot) for multi processor run (30.5 MB/s 4 lpar)

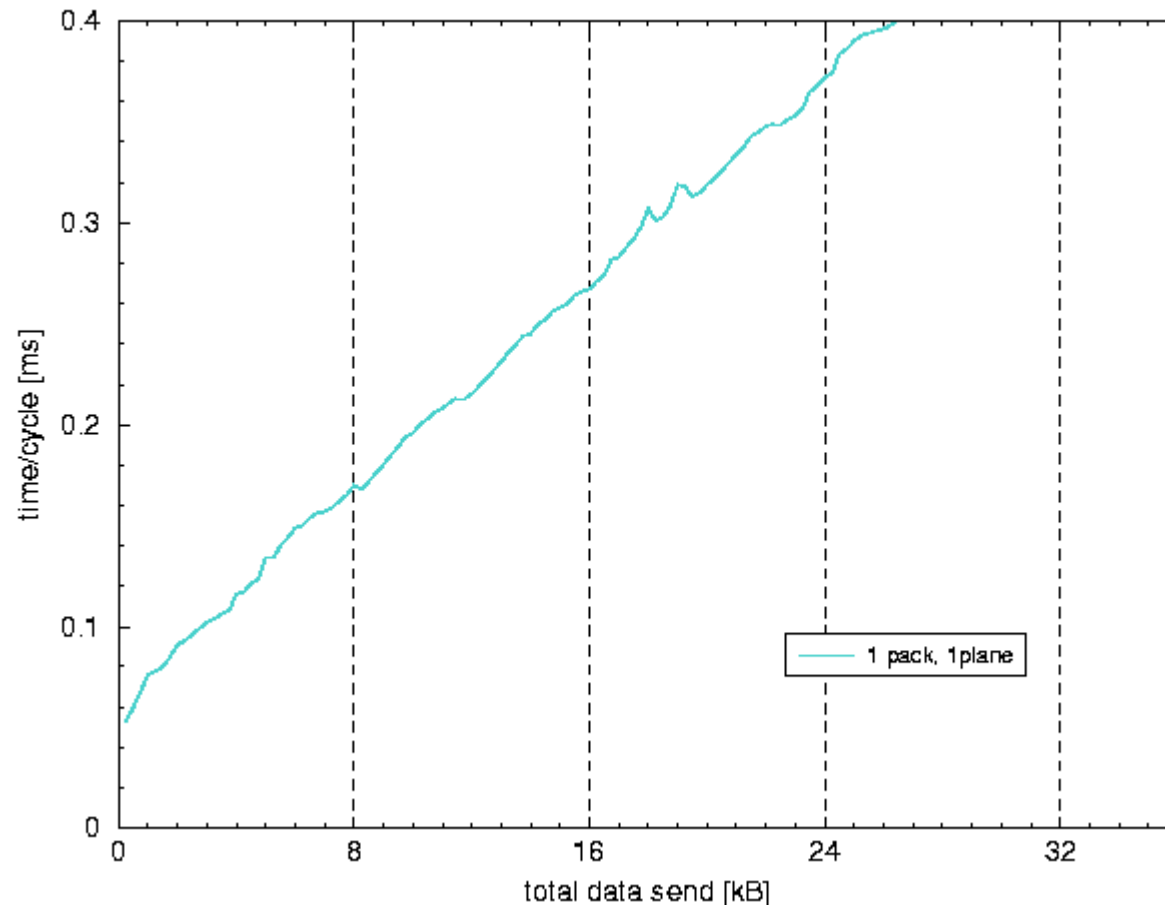


- Processor 0 sends  $n$  messages to Processor 1
- After receiving **all**, Processor 1 sends them back
- 4 implementations
  - a)  $n$  calls `MPI_Isend`, `MPI_Waitall(n)`
  - b)  $n$  calls `MPI_Send`
  - c)  $n$  calls `MPI_Issend`, `MPI_Waitall(n)`
  - d)  $n$  calls `MPI_Ssend`
- Timings for a whole cycle
- Start: keep message size fixed

# Performance of Multi-Ping-Pong

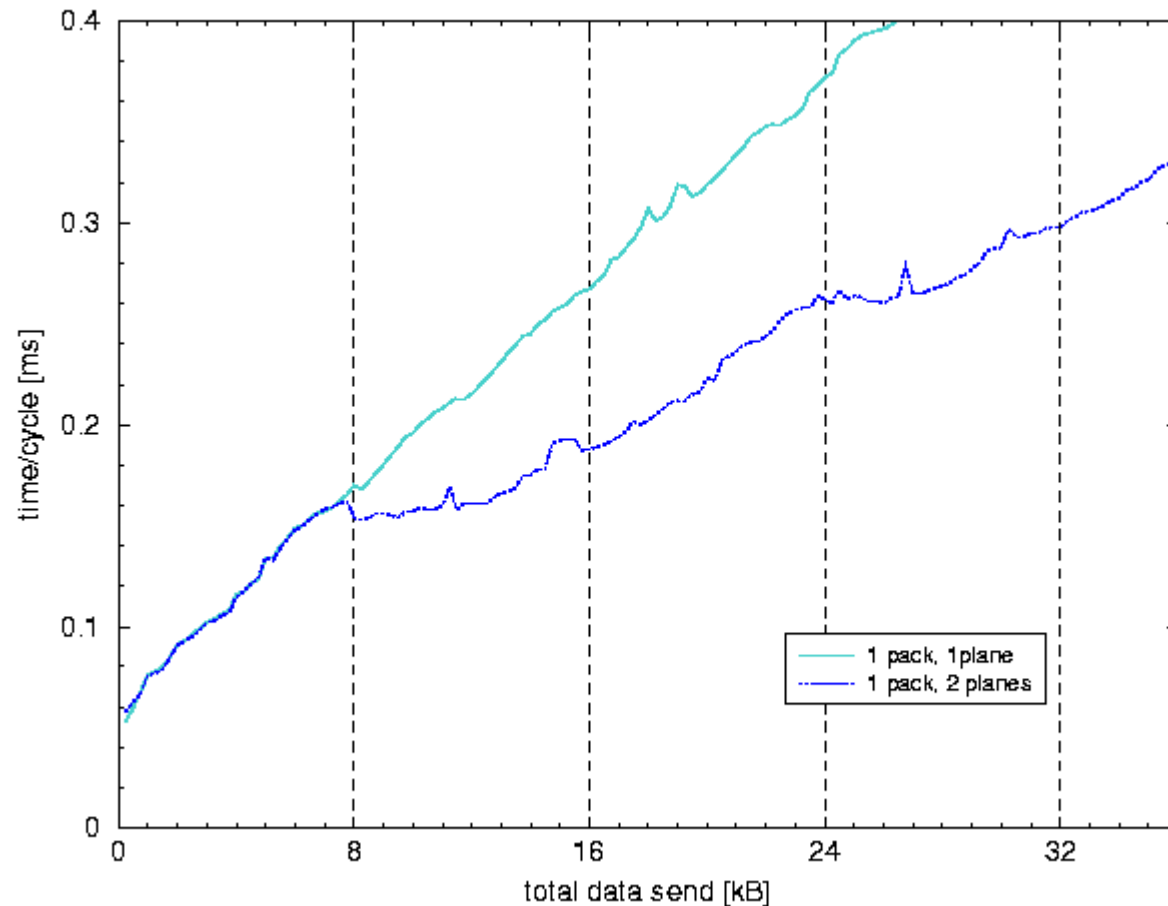


# 1 message via 1 plane



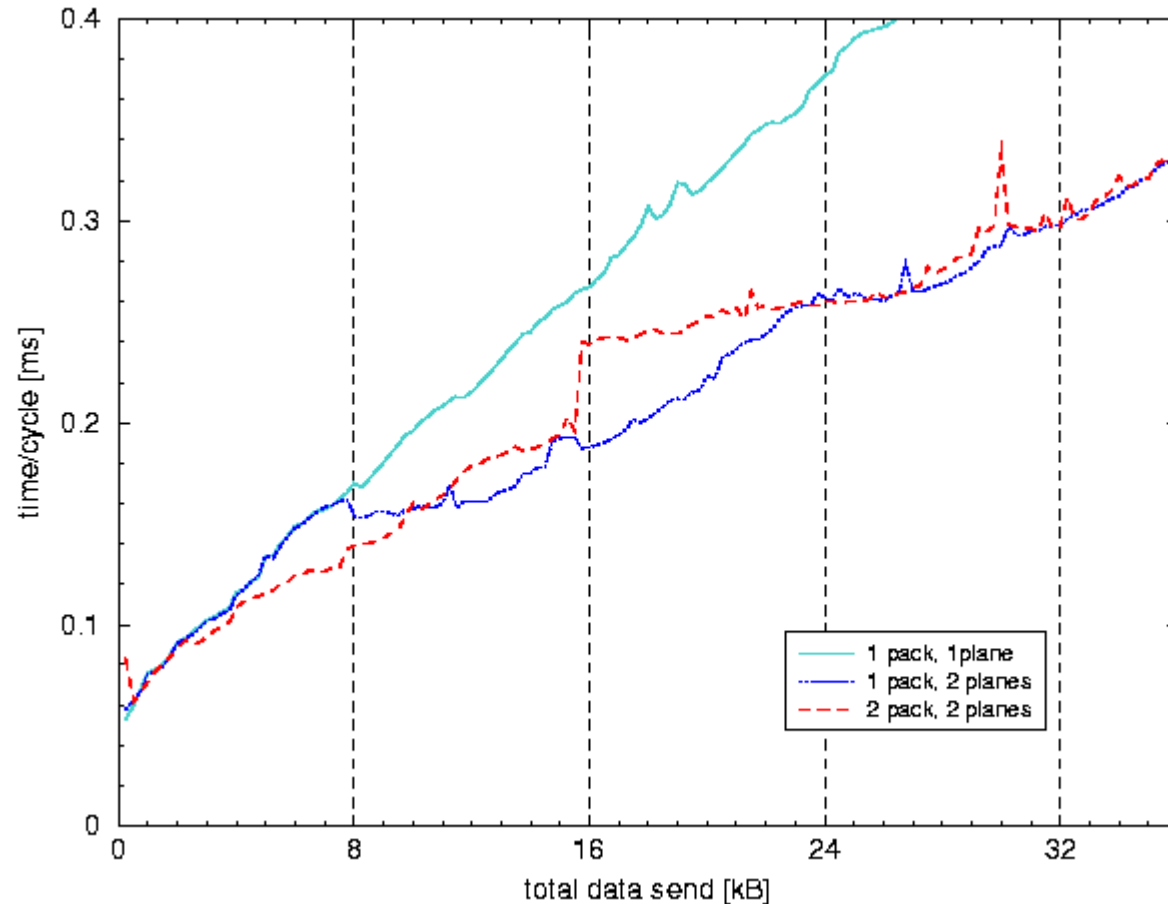
- Fits latency-bandwidth model

# 1 message via 2 planes



- 2<sup>nd</sup> plane used from 8 kB, 2<sup>nd</sup> plateau at 24 kB

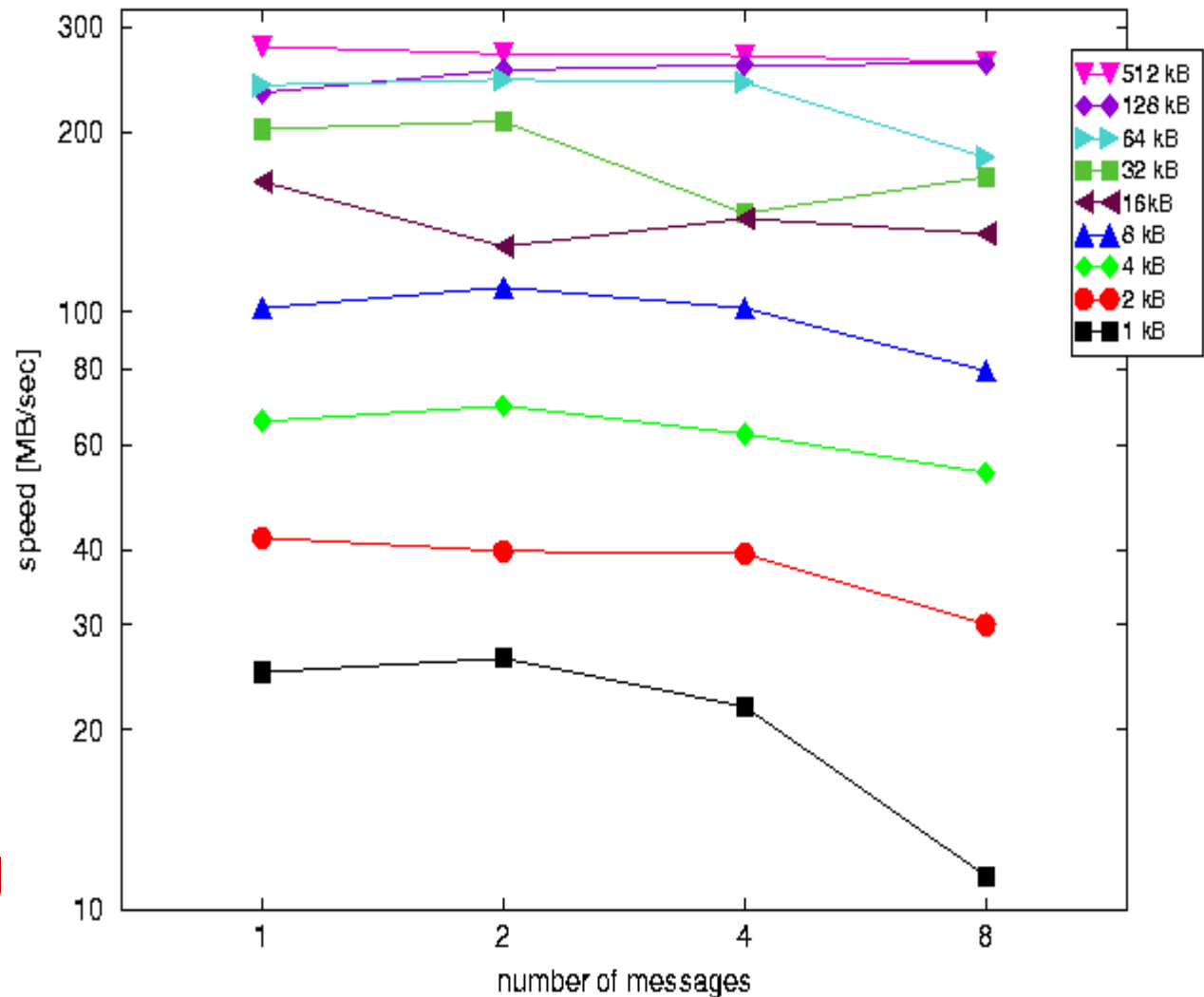
# 2 messages via 2 planes



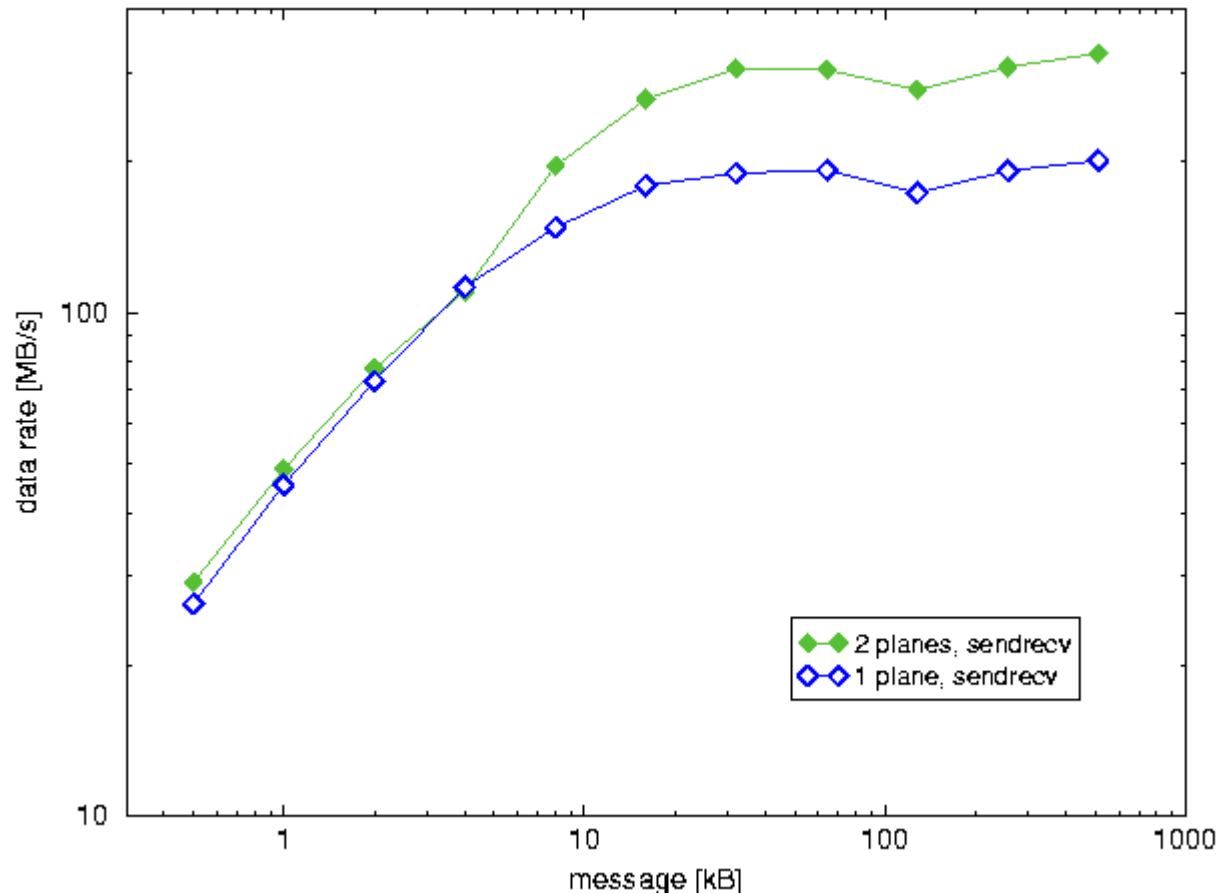
- Same latency, faster 4kB - 10kB, plateau 16kB - 28 kB

# Sending message in pieces

- Quite flat:
  - multiple mesg. about same speed as single message of total size
- Dip when pieces about 8kB
- Dividing to get EAGER sending advantageous

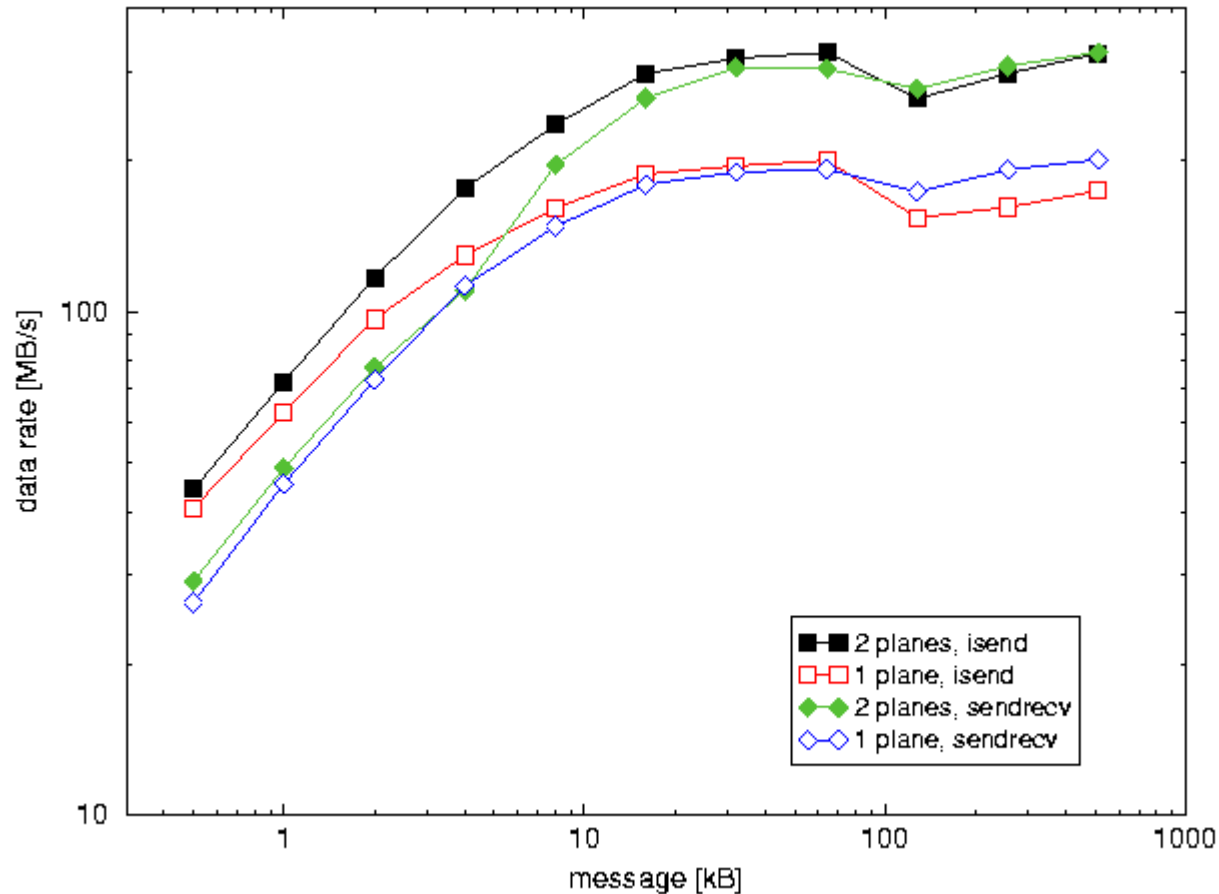


- So far all messages to the same processor
- But: the network is overspecified compared to the switch adapters.
- Do the Ipars care where the messages go?
- Arrange processors in a ring
- Each sends a message to both neighbours
  - Send to the left and receive from the right
  - Send to the right and receive from the left



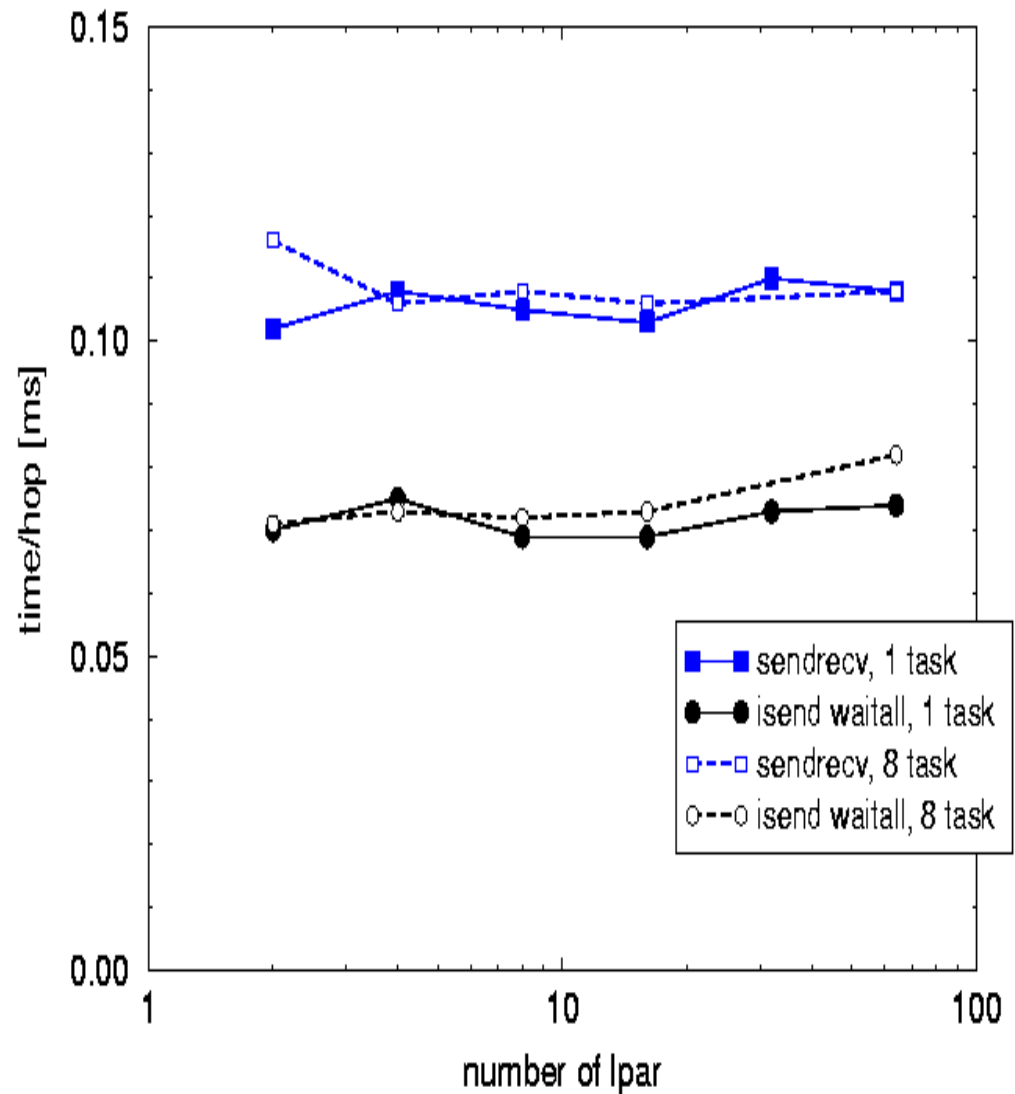
- Irecv-Isend-Waitall(2): similar performance
- 2<sup>nd</sup> plane used for more than 8 kB only

# 2Irecv-2Isend-waitall(4)



- non-blocking overlaps latency and uses 2<sup>nd</sup> plane earlier

- Study dependence
  - Number of lpar
  - Number of task/lpar
  - 2 kB message on ring
- Key effect due to communication pattern
- Noise (OS, etc.) obviously not the key effect



- Simple example of a 2-dimensional domain decomposition code

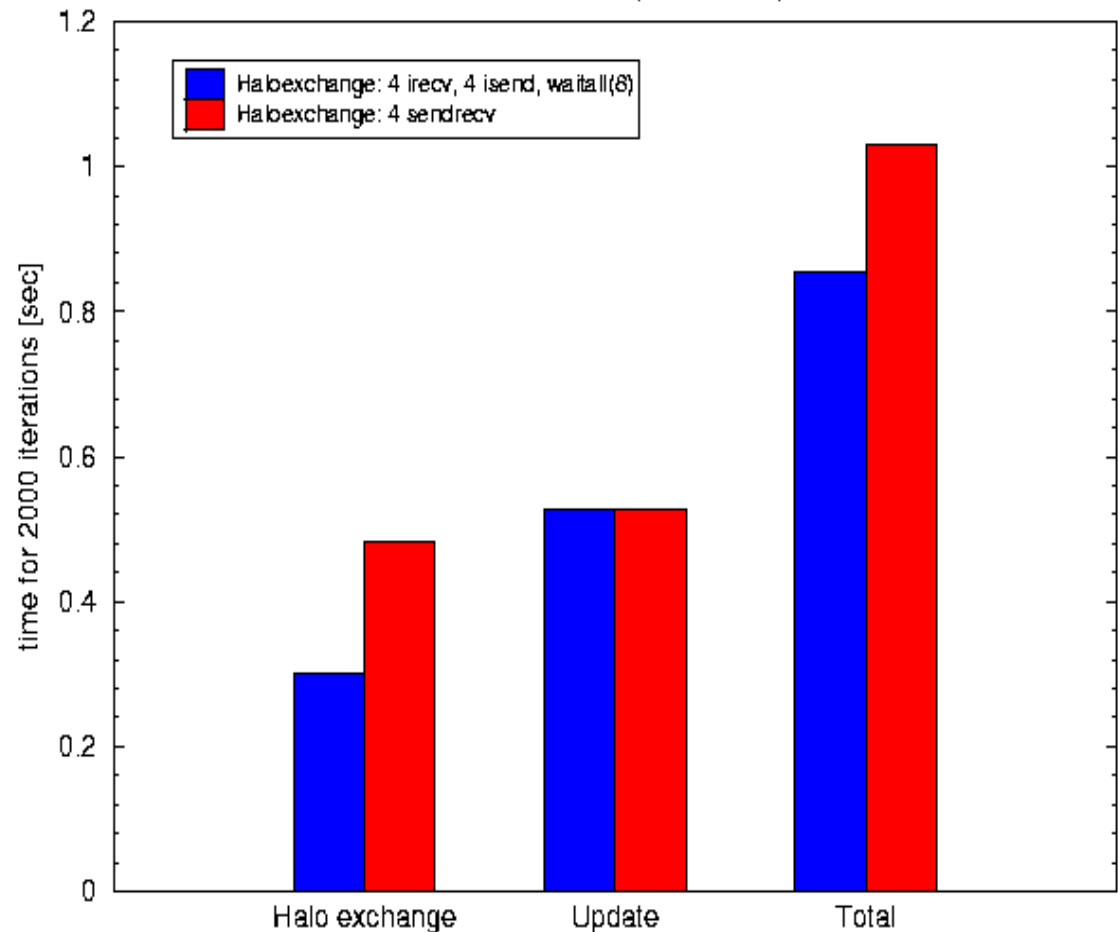
$$I_n(x_1, x_2) = \frac{1}{4} \left[ I_{n-1}(x_1 + 1, x_2) + I_{n-1}(x_1 - 1, x_2) \right. \\ \left. + I_{n-1}(x_1, x_2 + 1) + I_{n-1}(x_1, x_2 - 1) \right. \\ \left. - E(x_1, x_2) \right]$$

- Code discussed in detail at ScicomP7
- See also [www.hpcx.ac.uk/research/hpc](http://www.hpcx.ac.uk/research/hpc)

- No global communication
- Grid size:  
6720 \* 8064
- Halo sizes:
  - 848 bytes
  - 1016 bytes
- Speed up of Halo exchange:
  - 37%

## Jacobi inverter of Laplacian in 2 dimensions

6720 \* 8064, 128 lpar, 8 task/lpar



- **File I/O**
  - Use more than 1 switch adapter-pair for Write/Read
  - Master/Worker can not be recommended if performance is critical
  - Use of MPI-I/O is a trade between
    - Performance
    - Convenience
- **MPI point-to-point**
  - Sending several message non-blocking allows
    - Overlap of latencies
    - Better utilisation of dual plane architecture
    - Significant improvement for messages of a few kilo-bytes

- Discussions
  - David Henty
  - Lorna Smith
- Financial support
  - EPSRC