

# Implementing a Planned AlltoAllv operation

Stephen Booth (EPCC)

*s.booth@epcc.ed.ac.uk*

Kenton D'Mellow (EPCC)

*Kenton@epcc.ed.ac.uk*

Adrian Jackson (EPCC)

*a.jackson@epcc.ed.ac.uk*

## Why is AlltoAllv important

---

- Large scale changes of data decomposition are usually implemented using the `MPI_Alltoall` or `MPI_Alltoallv` calls
  - `Alltoallv` is equivalent to each process sending a different message to every other process.
  - The `Alltoall` operation is similar but all messages are the same size.
- They are particularly important in the implementation of parallel multi-dimensional fourier transforms where a series of data transposes are required.

- All messages are unique
  - On the face of it very little room to optimise
- We can exploit the clustered SMP architecture
  - Messages within a node can be sent with lower latency, higher bandwidth.
  - May be possible to redistribute data within a node.
    - Send smaller number of larger messages between nodes
    - Reduce message latency costs.
- Problems
  - Cost of data re-distribution must be less than the saved latencies. This approach will break down for sufficiently large messages.
  - Need to know how to redistribute the data.

# Optimising the local redistribution.

---

- **Two options:**
  1. Use MPI point to point
  2. Communicate via a shared memory segment (SMS).
- Though shared memory point-to-point is fast we may be able to do better for collective operations where the communication pattern is known in advance. (No need to match send/recv calls)
- To investigate these options we built on previous work where we optimised tree based communications using a SMS.
  - Code is written to be switchable between MPI and SMS to allow comparisons.

- With the Alltoall call all participating processes know the full communication pattern.
  - This can therefore be implemented directly.
- With Alltoally only the communications involving the local process are known.
  - Additional communication required to distribute information.
  - This could negate any advantage of the optimisation.
  - However most application codes will repeat a small number of data redistributions many times.
    - Leads us to a “planned” alltoally approach

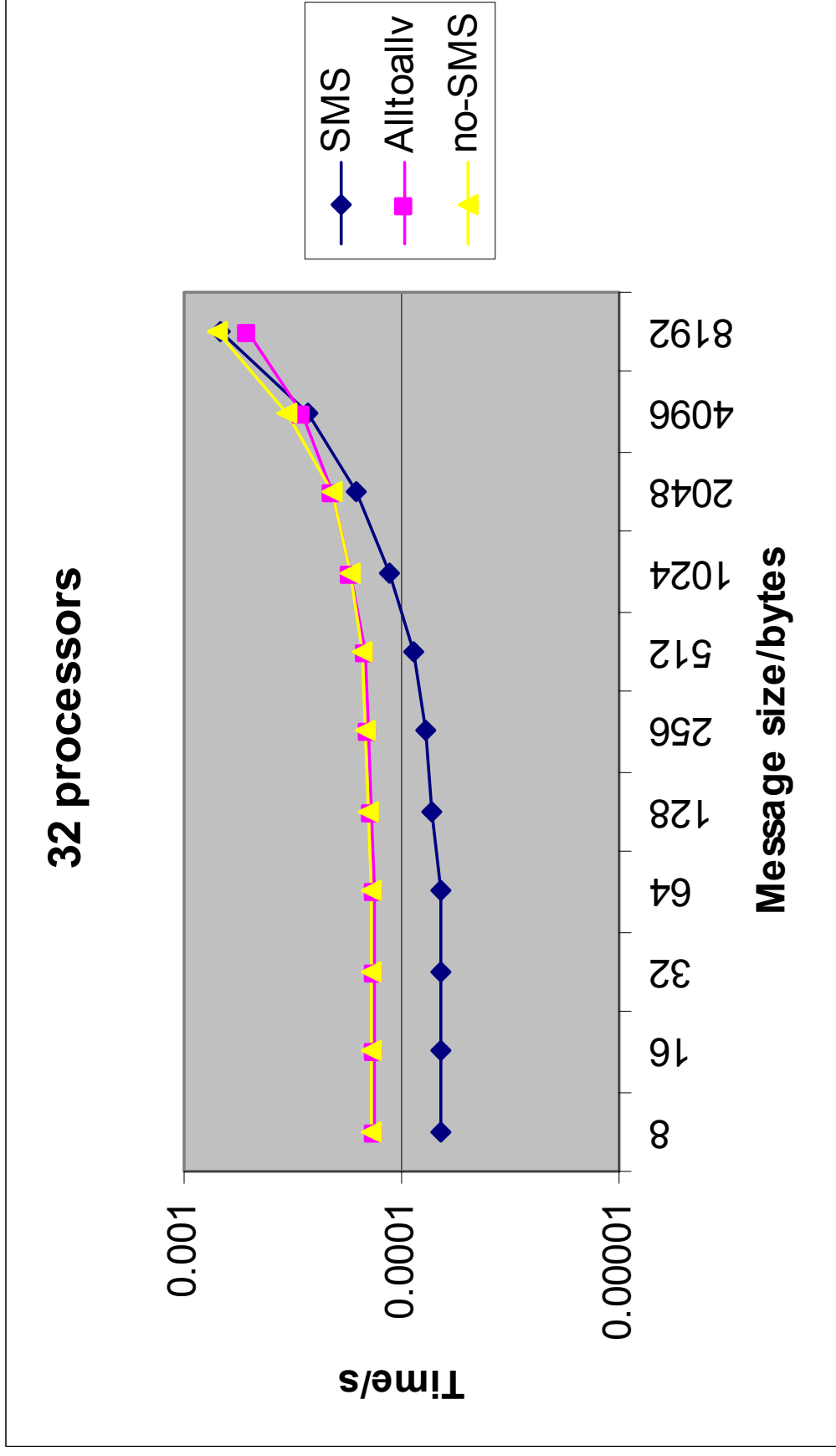
- Use the Alltoally parameters to construct a Plan object.
  - This is collective.
  - Relatively expensive.
- Plan can be re-used multiple times to perform the communication step.
- Additional communications and computation only performed once.

- Aim for only a single message to be passed between separate SMPs.
- Inter-box sends and receives equally distributed across processes within a box.
- Three stage process:
  - Collect data to sending process.
  - Exchange data between SMPs
  - Distribute data to correct end-point.
- Collect/Distribute performed by either
  1. MPI\_Pack/MPI\_Unpack to SMS
  2. MPI point-to-point

- Plan objects contain lists of communications for each stage.
  - Point to point communications stored as MPI persistent communication requests.
  - Pack/Unpack stored as linked list of structures.
- **SMS version uses only a single segment per SMP**
  - Designed to work robustly with split-communicators
  - Same segment used for the Tree based communications.
  - Wherever possible use normal MPI calls, overriding via the profiling interface.

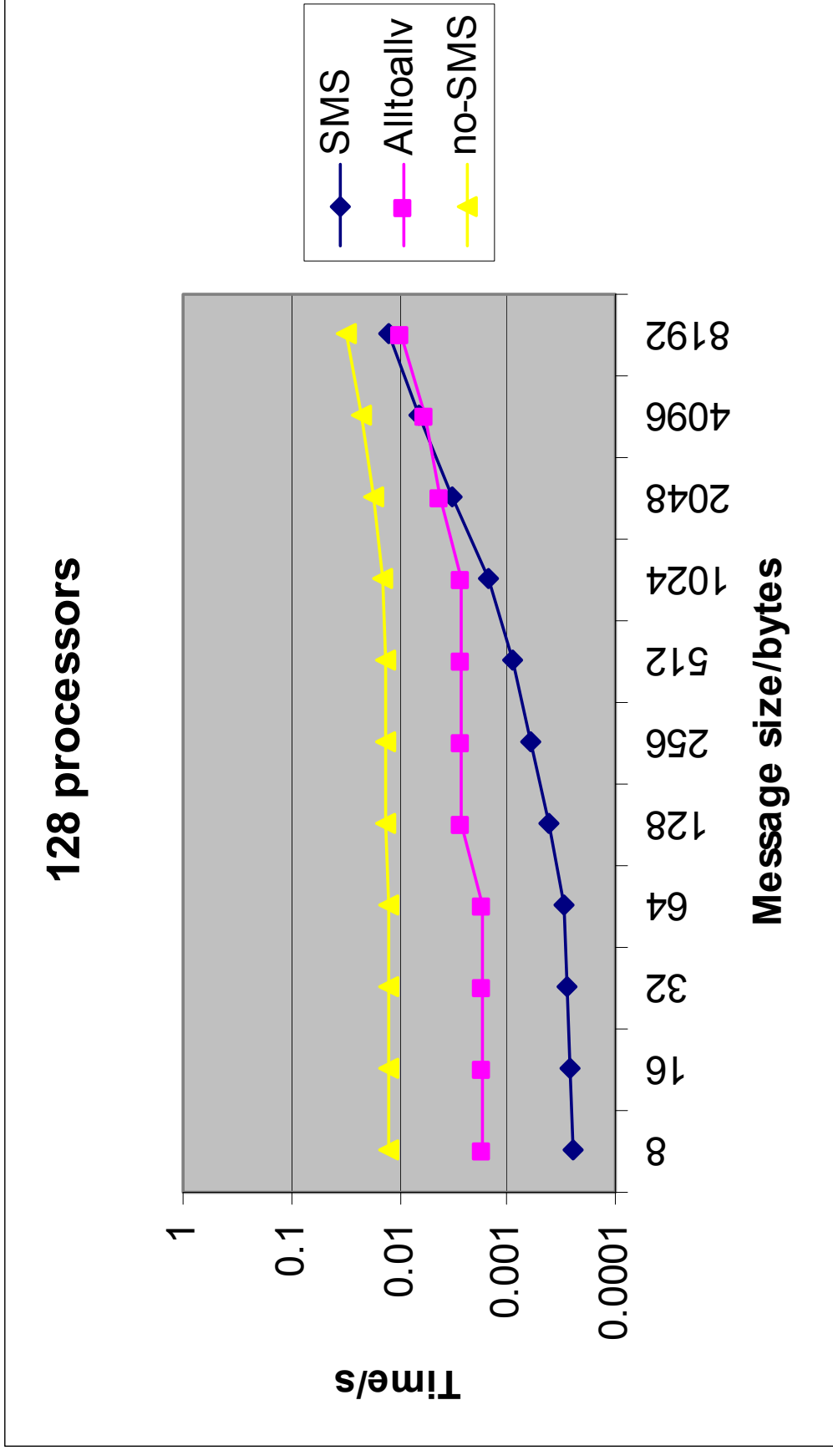
- Optimisation effect should be greatest for large number of small messages.
- Benchmark the planned Alltoall code using Alltoall equivalent communication patterns at a variety of message sizes.
- Initial benchmarks run on HPCx phase2
  - 32 way p690+ nodes
  - Service Pack 12

# Single SMP results

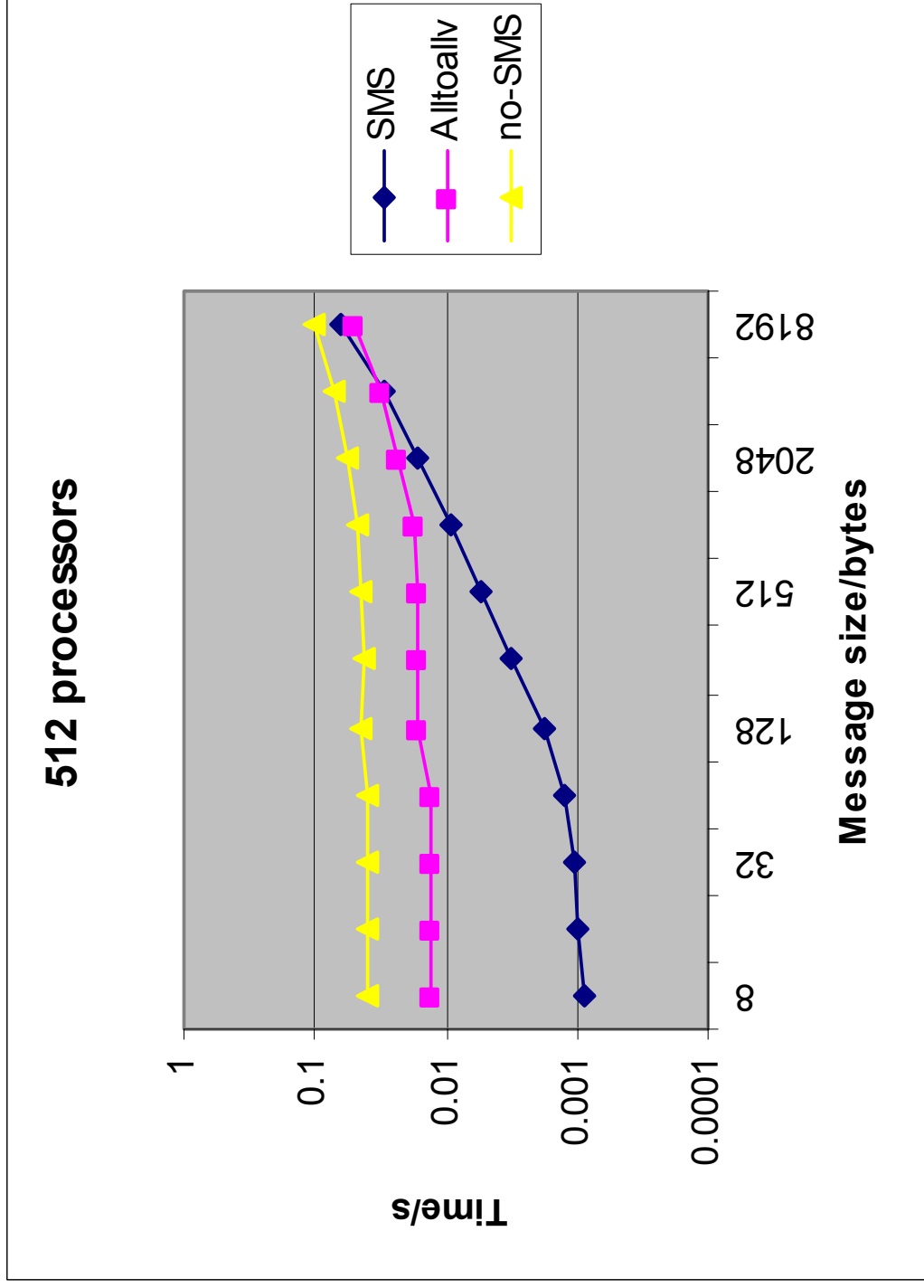


- Within a single box only a 2 stage copy is used.
  - SMS version Packs/Unpacks from the segment
  - Pt-to-pt version uses single message send.
- Pt-to-pt version very similar to native Alltoally
  - Actually slightly slower, Alltoally may optimise the send to the local processor as a copy.
- SMS approach faster for message sizes up to 4K

# Multiple SMP results

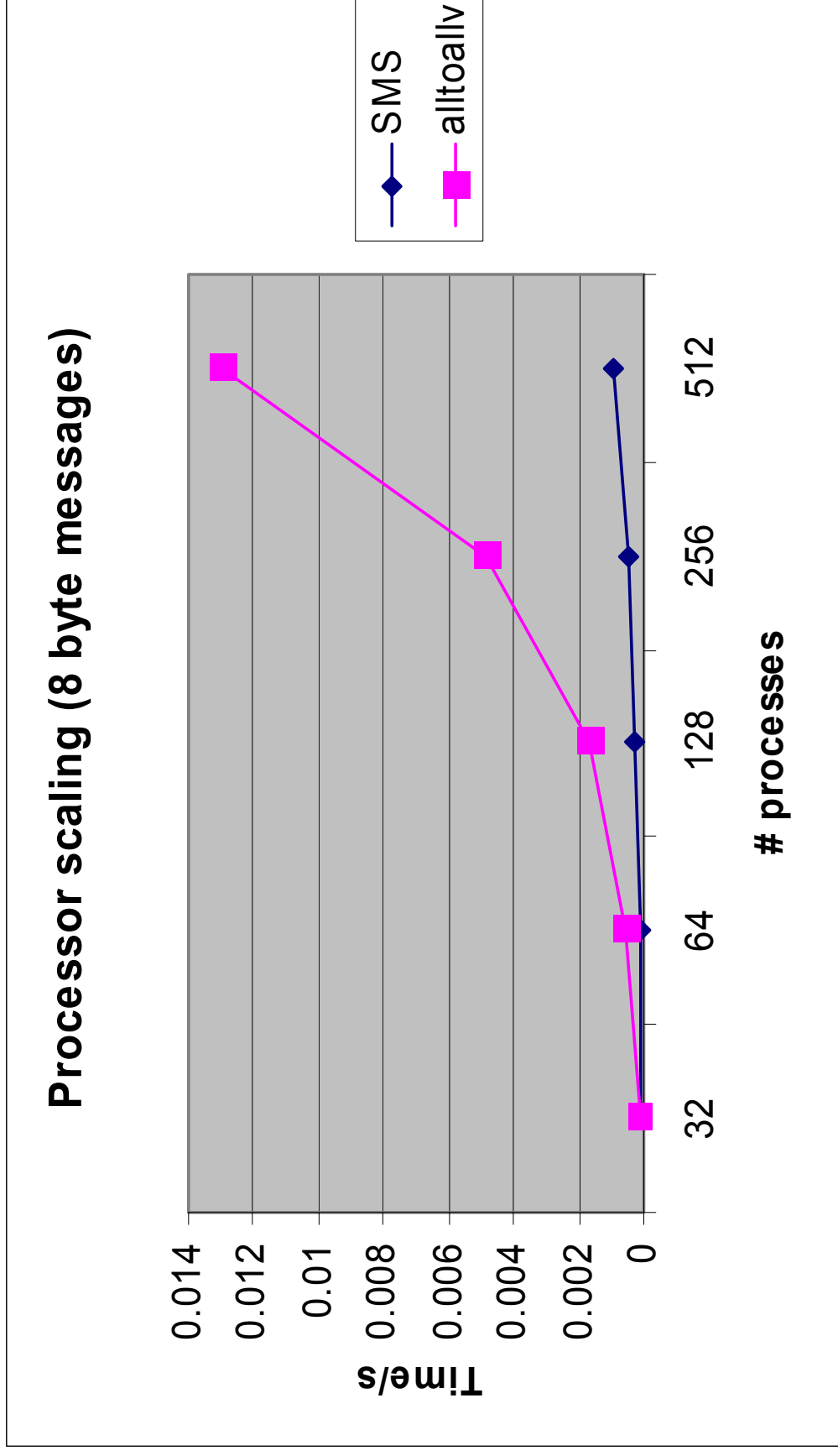


- Pt-to-pt version always slower
  - Under SP12 difference between shared memory and switch latencies does not seem great enough for this optimisation to work. Only a couple of microsecond difference in latencies much reduced from initial install.
- SMS version faster up to 2K messages
- With 32 way nodes we are combining  $32 * 32 = 1024$  messages
  - Large number of message latencies to save



# Small message scaling

---



- Message aggregation is a workable optimisation for Altoallv IF:
  - Aggregation performed via a Shared memory segment.
  - Messages are small  $\ll \sim 2K$ .
  - Communication pattern is calculated in advance.
- Optimisation is much more significant at large processor counts.

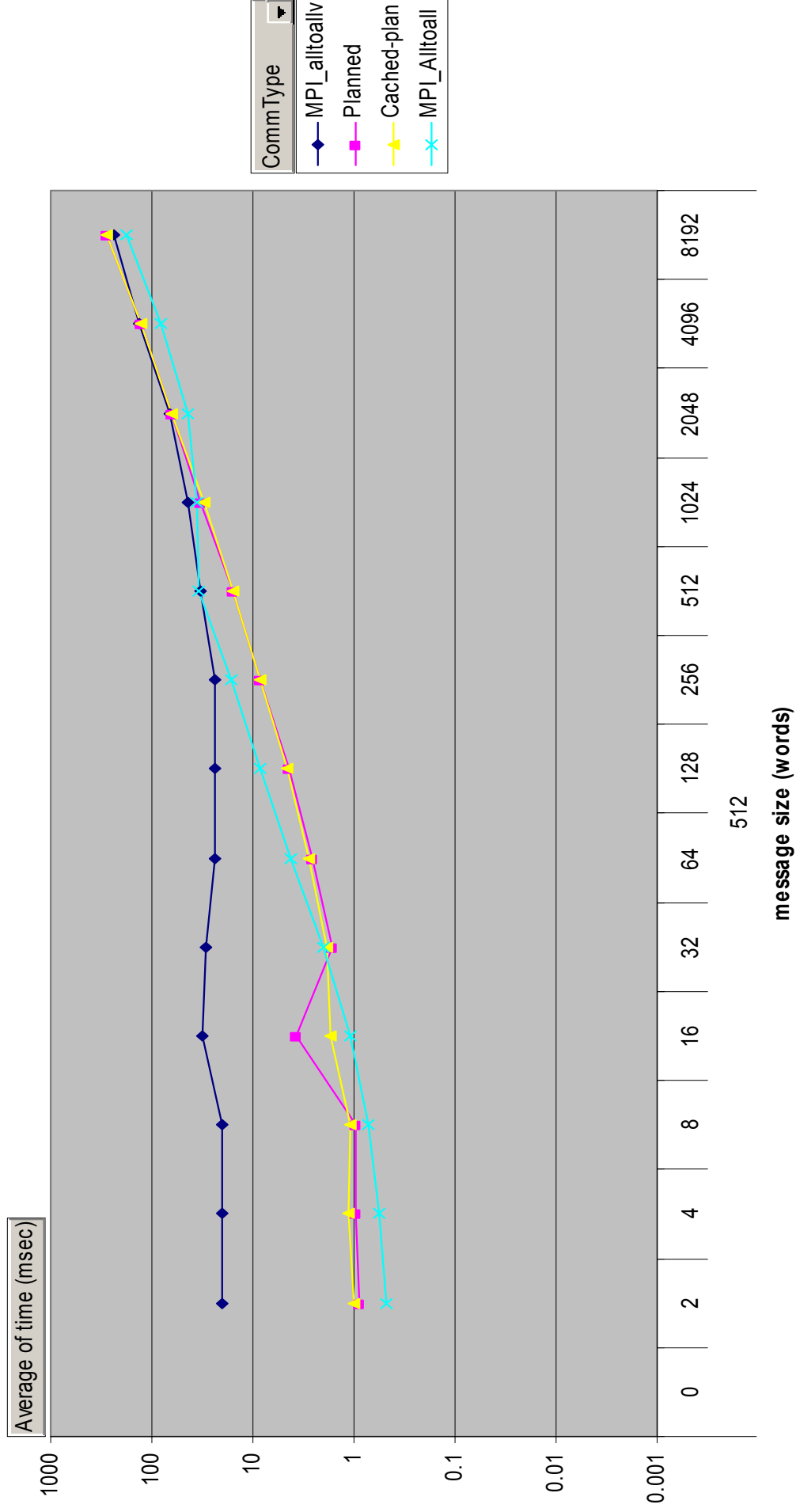
- AlltoAll is much easier to optimise in this way.
  - All message sizes are the same so the full communication pattern can be deduced from the local arguments.
- The IBM MPI library includes this kind of optimisation for the AlltoAll operation.
  - No advantage to using planned operations for true AlltoAll.
  - If your AlltoAllv operations are equivalent to an AlltoAll on a split communicator then you may want to refactor to use AlltoAll.

- Originally we implemented new communication calls with explicit calls to make and run plans.
- This proved to be difficult to integrate with existing applications.
- Really we need to make the *Standard MPI\_Alltoallv* go faster.

- Too expensive to make a new Plan each time MPI\_Alltoallv is called.
- Instead we implement a Plan cache
  - Each time MPI\_Alltoallv is called search the cache for a plan with matching arguments making a new plan only if no match found.
  - Still not enough to guarantee that the Plan is suitable, Arguments must match on *all* processors
  - Need collective operation to vote on candidate plan.

- These are early results on phase2a
  - IBM eServer 575
  - 16 1.5 Ghz Power-5 per node
- Again benchmarking using the same communication pattern as an Alltoall

# Phase2a performance



- Message combining can be made to work for both MPI\_Alltoall and MPI\_alltoall.
- Additional overhead for preserving the standard MPI\_Alltoall interface is small.
- Performance of the IBM Alltoall implementation and my cached plan implementation very similar.