Improvements in I/O for DL_POLY_3

Ian Bush
Numerical Algorithms Group Ltd, HECToR CSE

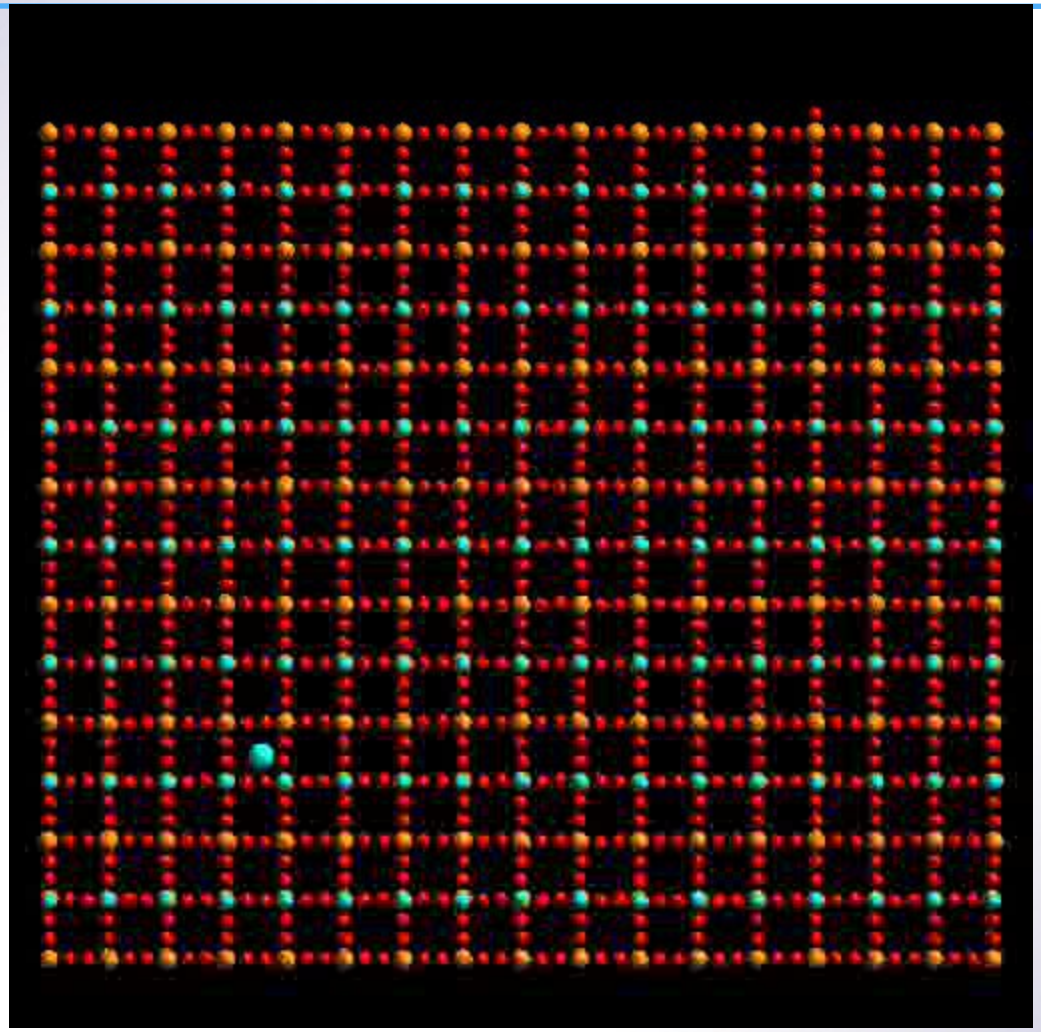
The DL_POLY_3 MD Package

- General purpose MD simulation package
- Written by Ilian Todorov and Bill Smith at STFC Daresbury Laboratory
- Written in modularised free formatted Fortran 95 - FORCHECK and NAGWare verified
- Generic parallelisation (for short-ranged interactions) based on spatial domain decomposition (DD) and linked cells (LC)
- Long-ranged Coulomb interactions are handled by SPM Ewald employing 3D FFTs for k-space evaluation
- Full force field and molecular description but no rigid body description yet (as in DL_POLY_2)

HECToR

RESEARCH COUNCILS UK

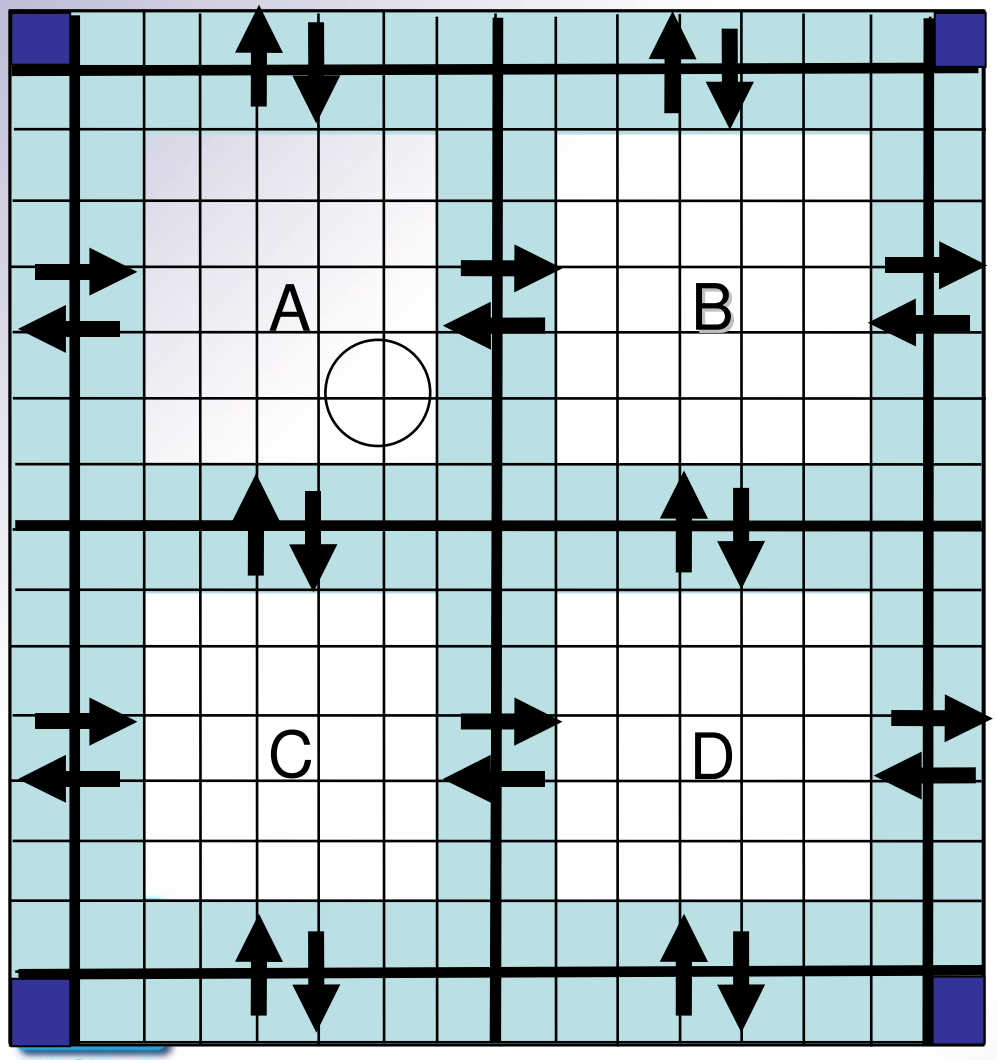
What is Classical MD ?

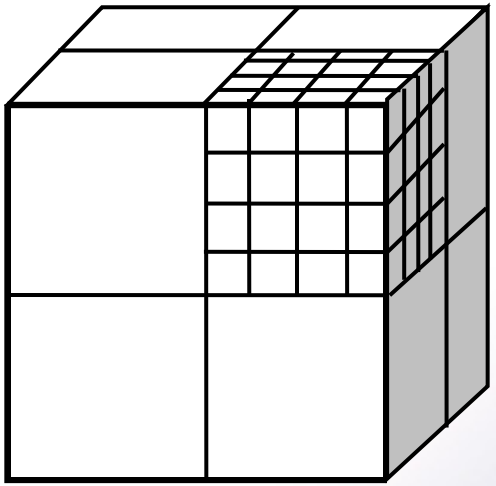


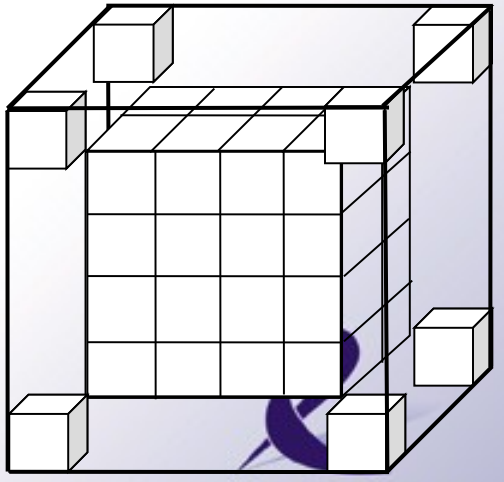
HECToR

RESEARCH COUNCILS UK

Domain Decomposition Parallelisation



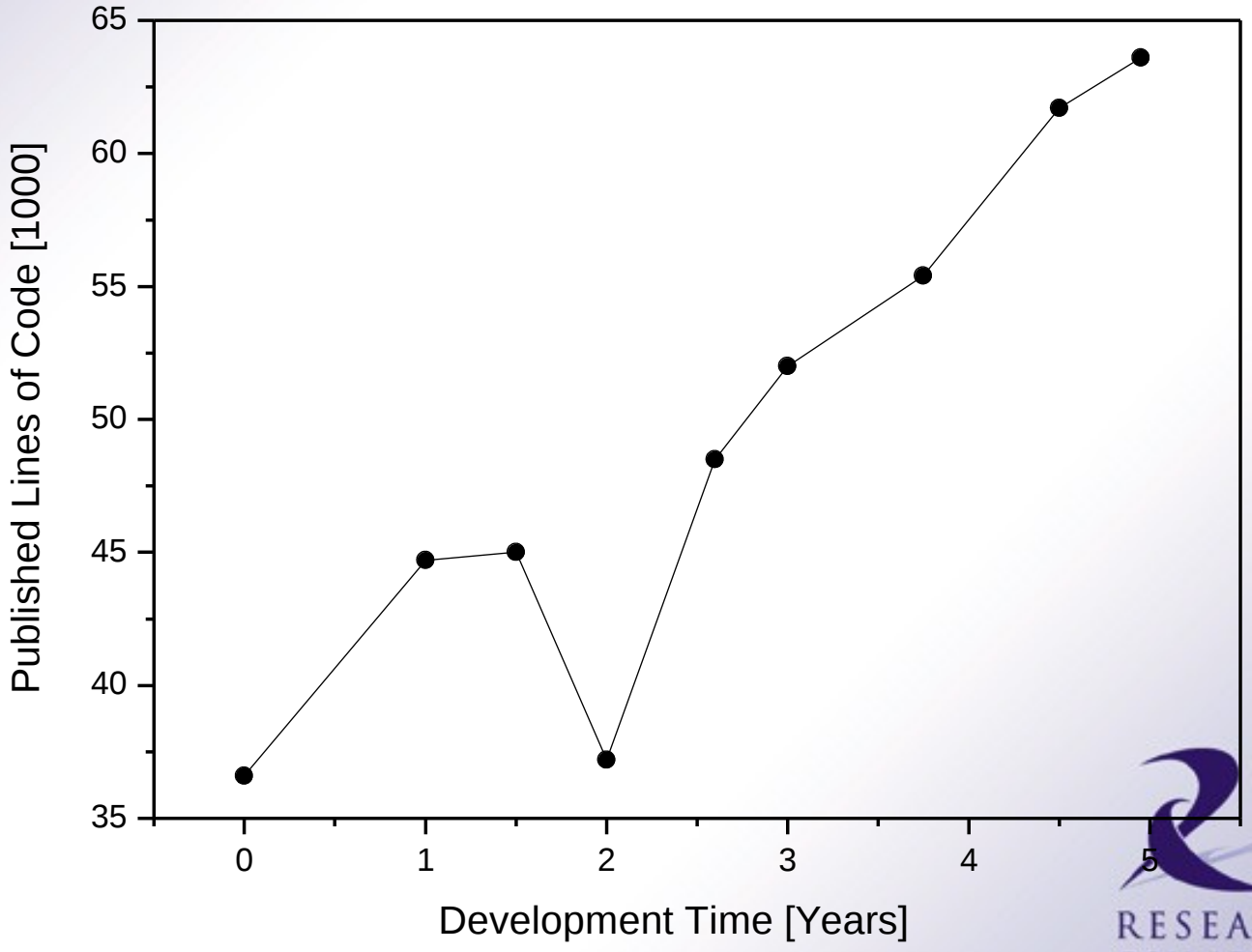




HECToR

RESEARCH COUNCILS UK

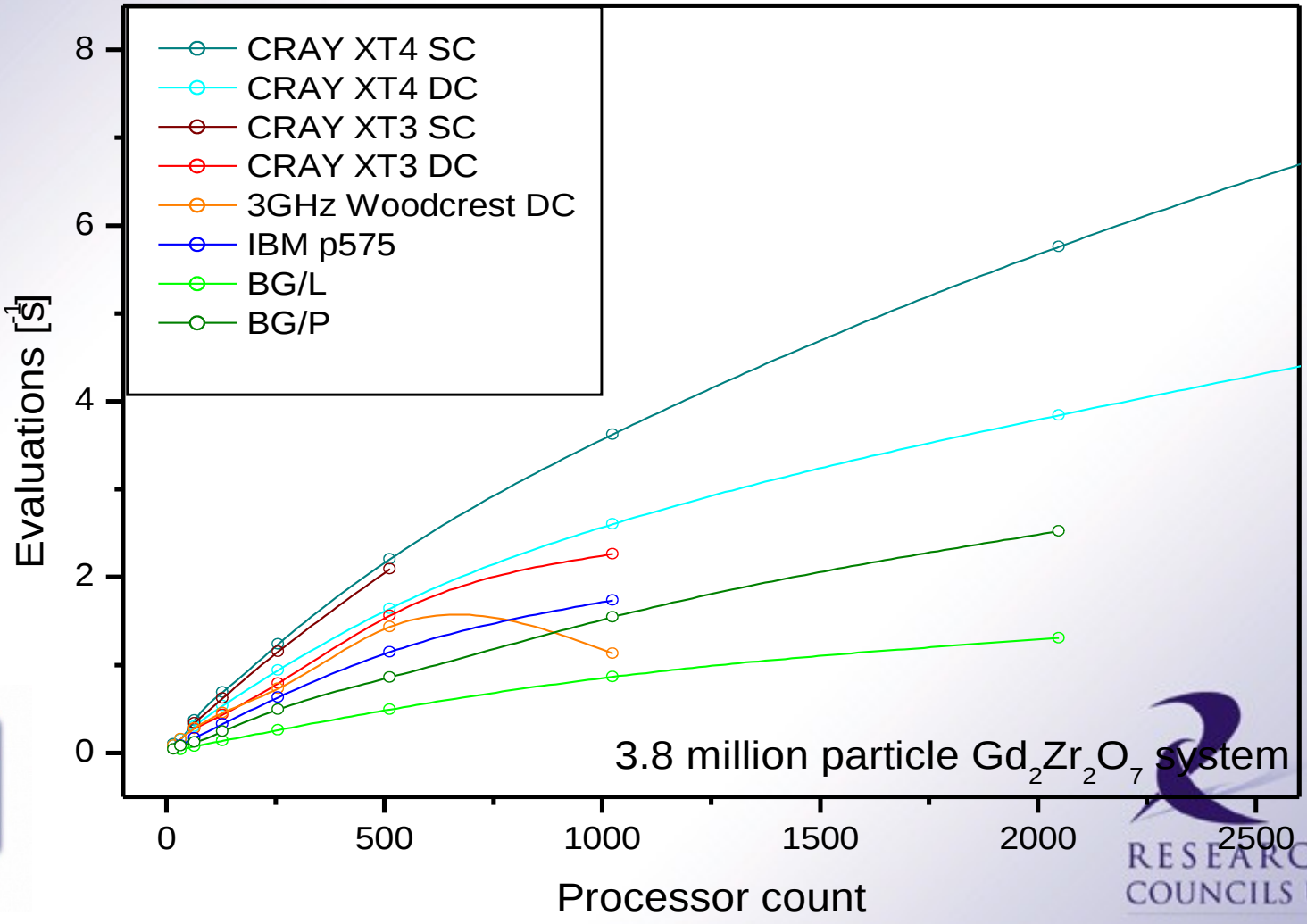
Development



HECToR

RESEARCH COUNCILS UK

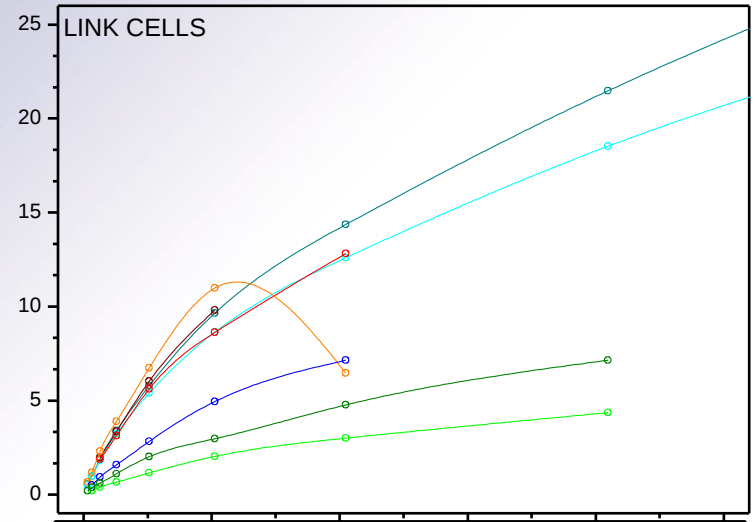
Benchmarking Main Platforms

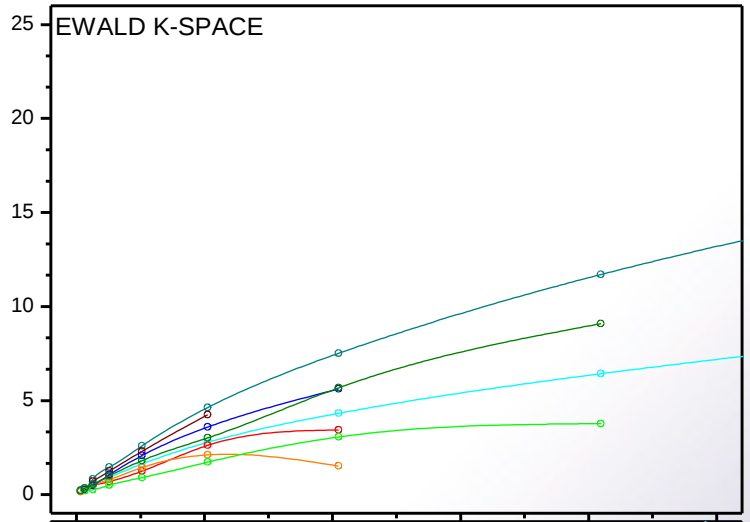


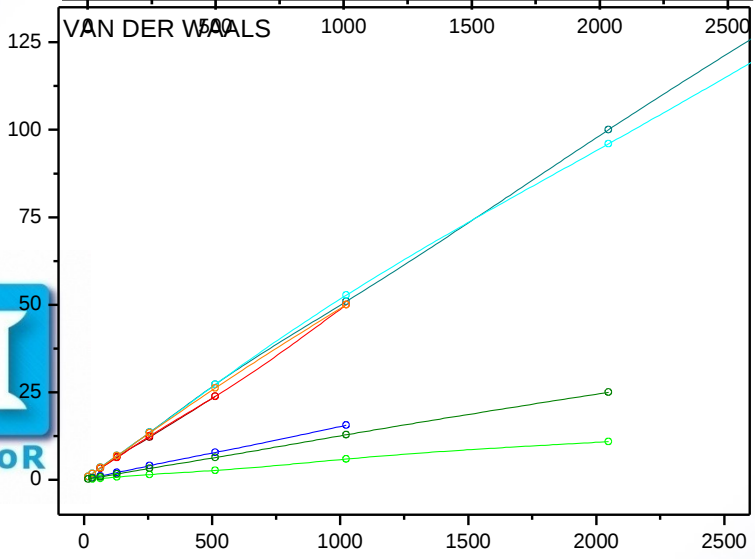
HECToR

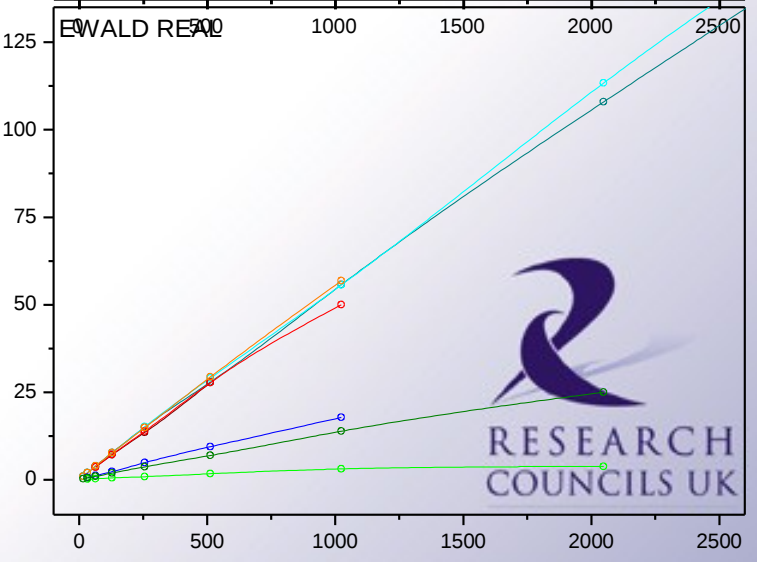
RESEARCH COUNCILS UK

Benchmarking Main Platforms





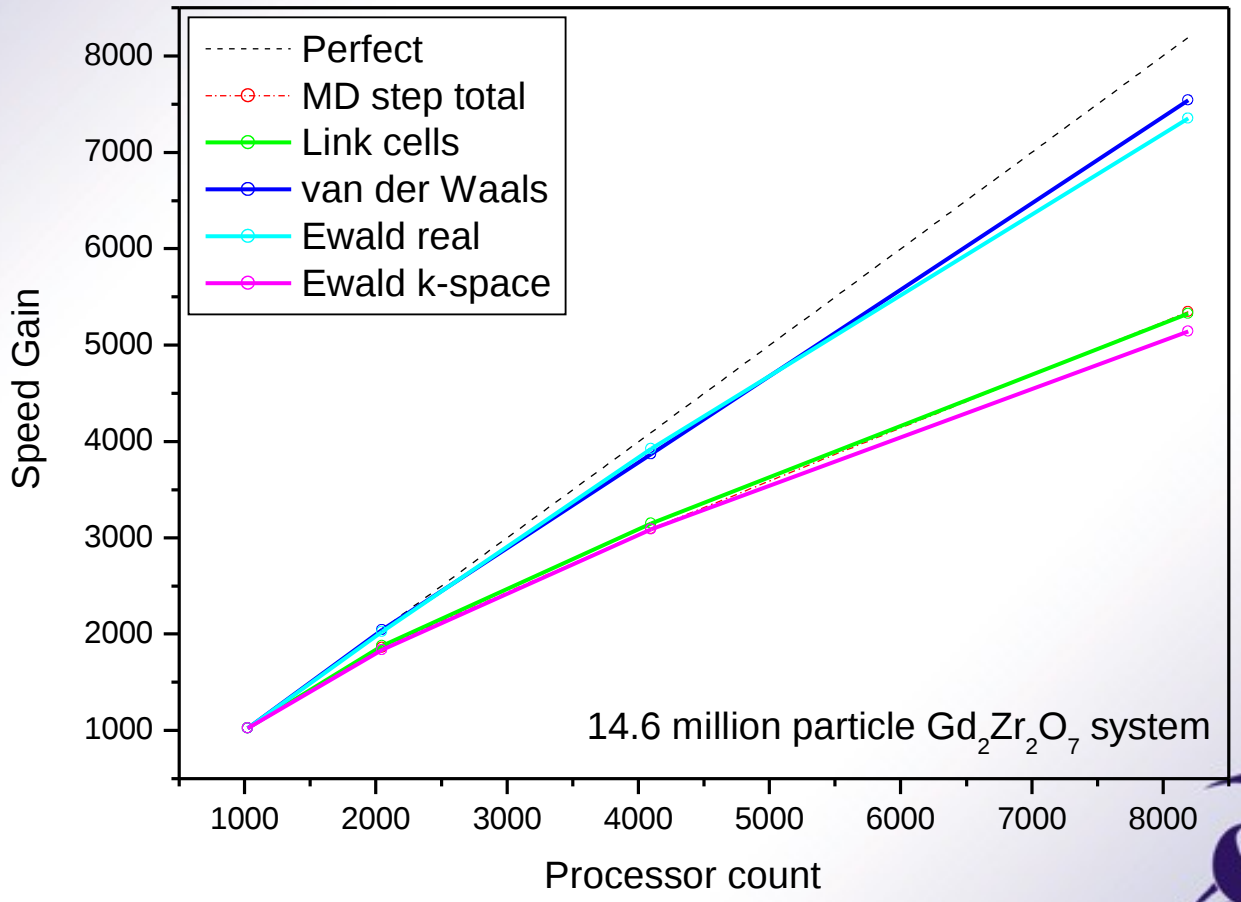




HECToR

RESEARCH COUNCILS UK

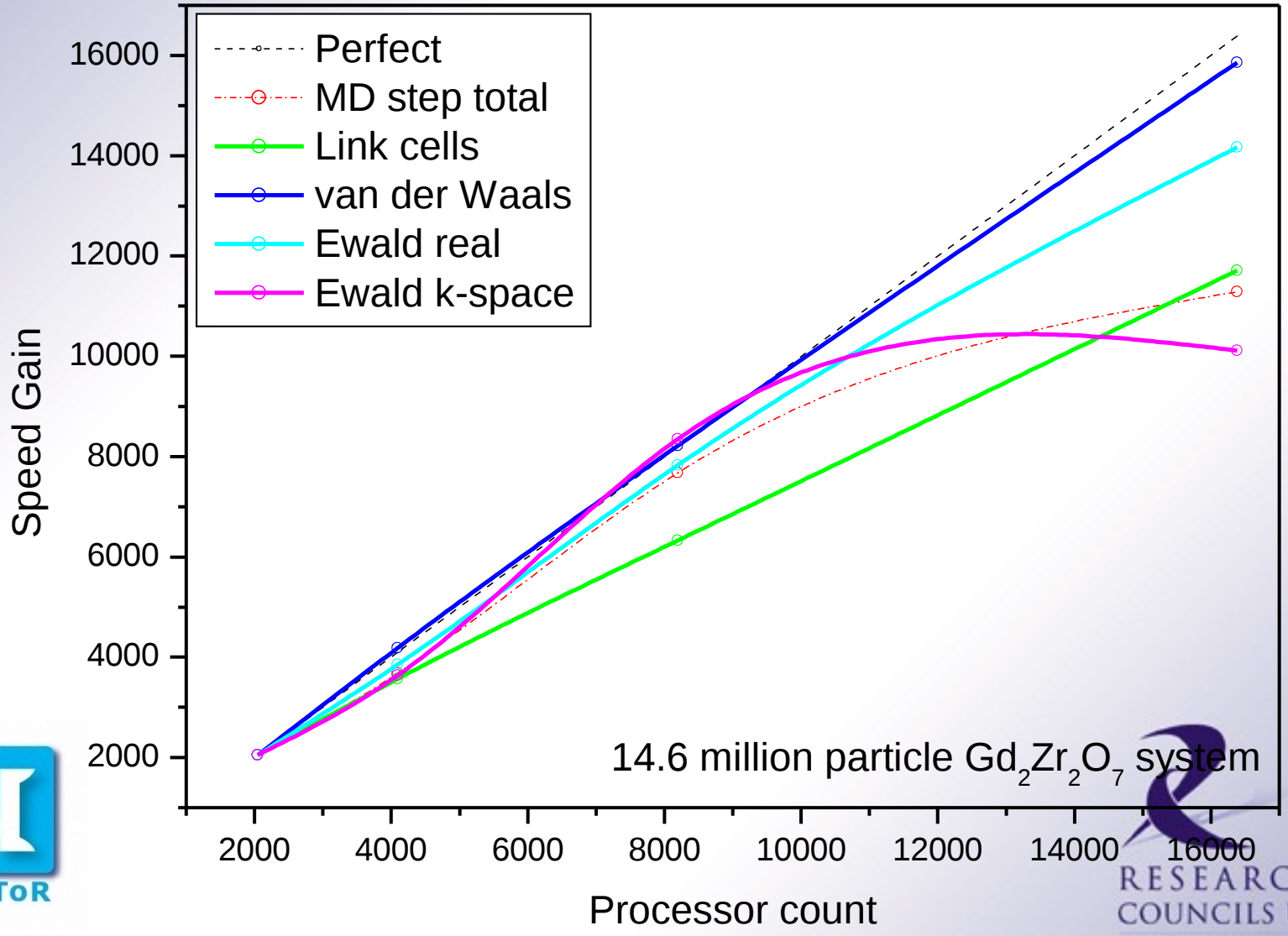
A Bit Bigger (On HECToR Phase 1)



HECToR

RESEARCH COUNCILS UK

A Bit Bigger (On BG/L)



HECToR

RESEARCH COUNCILS UK

Bigger Still (IBM P575)

300,763,000 NaCl with full SPME electrostatics evaluation on 1024 CPU cores

Start-up time	≈ 1 hour
Timestep time	≈ 68 seconds
FFT evaluation	≈ 55 seconds

In theory ,the system can be seen by the eye. Although you would need a very good microscope – the MD cell size for this system is 2µm along the side and as the wavelength of the visible light is 0.5µm so it should be theoretically possible.

But this starts to show the problem ...

HECToR

RESEARCH COUNCILS UK

The Problem

- For the 14,600,000 particle system on 16,384 processors of the the Jülich BG/L system it takes ~0.5s for a MD timestep
  - Fast enough to do science !
- ~1800s to write the coordinates
  - Not fast enough to do science !
- Want to write the coordinates every ~100-1000 timesteps

So while the compute is fast enough the I/O prohibits any useful science being done

HECToR

RESEARCH COUNCILS UK

It’s Not Just BG/L

- 15 million system on 2048 processors of HECToR Phase 1
  - MD time per timestep ~0.7 seconds on Cray XT4
  - Configuration read ~100 seconds (once during the simulation)
  - Configuration write ~600 seconds
- I/O in native binary is only 3 times faster and 3 times smaller
  - So just using binary is not a solution

Some Unpopular Solutions

- Saving only the important fragments of the configuration
  - What's important ?
  - How does the user specify that ?
- Saving only fragments that have moved more than a given distance between two consecutive dumps
  - Rewrite of analysis programs required
- Distributed dump – separated configuration in separate files for each MPI task
  - The Files ! The Files !
  - Restart on a different number of processors ?

HECToR

RESEARCH COUNCILS UK

So What Do We have To Write ?

```
pyrochlore
         2          3    3773000         50      0.00003125      0.00156250
       378.0382791976         0.0000000000         0.0000000000
         0.0000000000       378.0382791976         0.0000000000
         0.0000000000         0.0000000000       378.0382791976
GD               3
     -186.2697242          -188.9656799          -186.3793036
     0.2315100734          -1.673201463          0.9363383539
      13210.65286          -235052.7542           44828.56133
GD               4
     -188.9764926          -186.3753017          -186.3328710
    -0.2949178501          0.9443083034           2.428692460
     -254542.5135           49396.61430           67986.12075
GD               5
     -189.0096634          -183.5772665          -183.4873639
      1.344516913          0.3640837776E-01      -1.250456823
     -21153.56476           1492.614280           949.9063469
GD               6
     -186.2854413          -180.8116309          -183.7179432
    -0.3272091542         -0.3909127980          -2.407327182
     -5003.623307          -288.9791458           5327.259472
GD               7
     -186.1640453          -183.6603272          -181.1469216
     0.4695334076         -0.6539792816          0.2197201872
     -7260.018574           9301.762012           24438.07425
```

And What's The Problem Writing It ?

- **The atoms move!**
- An atom can migrate from one processor to another, so the ordering of atoms is **not preserved**.
- But users' analysis programs (e.g. for visulization) often assume that the ordering is preserved.
- So have to rearrange data so that it can be written out in the form the users require.
- Also files need to be **portable** – often the analysis is done on a different machine from that upon which the MD is performed. So we need a portable format.

HECToR

RESEARCH COUNCILS UK

Initial Attempts At A Solution

1. **Serial direct access write** (abbreviated SDAW) – where only a single node, the master, prints it all and all the rest communicate information to a master in turn while the master completes writing a configuration of the time evolution.
2. **Parallel direct access write** (PDAW) – where all nodes print in the same file in an orderly manner so no overlapping occurs using Fortran direct access files. However the behaviour of this method is not defined by the Fortran standard.
3. **MPI-I/O write** (MPIW) which has the same concept as the PDAW but is performed using MPI-I/O rather than direct access. This is portable.
4. **Serial NetCDF** write (SNCW) using NetCDF libraries for machine-independent data formats of array-based, scientific data (widely used by various scientific communities)

HECToR

RESEARCH COUNCILS UK

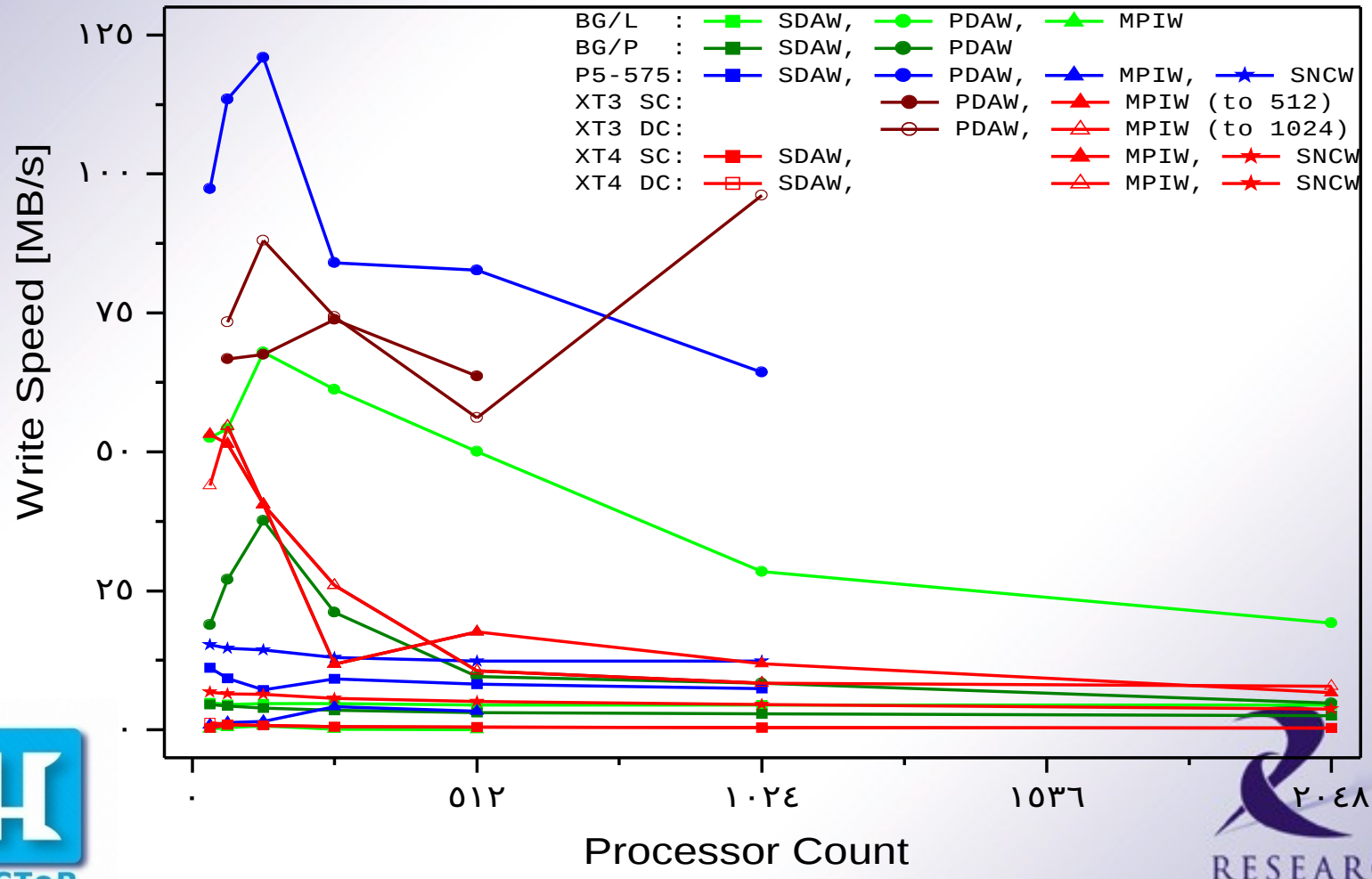
Comparison of The Methods

Method	Parallel	Portable	Human Readable
SDAW	No	Yes	Yes
PDAW	Yes	No	Yes
MPIW	Yes	Yes	Yes
SNCW	No	Yes	No

HECToR

RESEARCH COUNCILS UK

Performance of the Methods



HECToR

RESEARCH COUNCILS UK

Initial DL_POLY_3 I/O Conclusions

- In general parallel methods achieve a higher I/O bandwidth
- PDAW performs markedly superiorly to the SDAW or MPIW where supported by the platform
- MPIW performs consistently well for Cray XT3/4 architectures, and for this architecture MPIW is much better than SDAW, which is extremely slow on the XT3 (9 hours !! results not shown).
- MPIW performs badly on IBM platforms.
- While on the IBM P5-575 SNCW was only 1.5 times faster than SDAW on average, on the Cray XT4 it was 10 times. May be more a reflection of the slowness of SDAW on the Cray machines.

It's all a very sad story!

But What About The Scaling

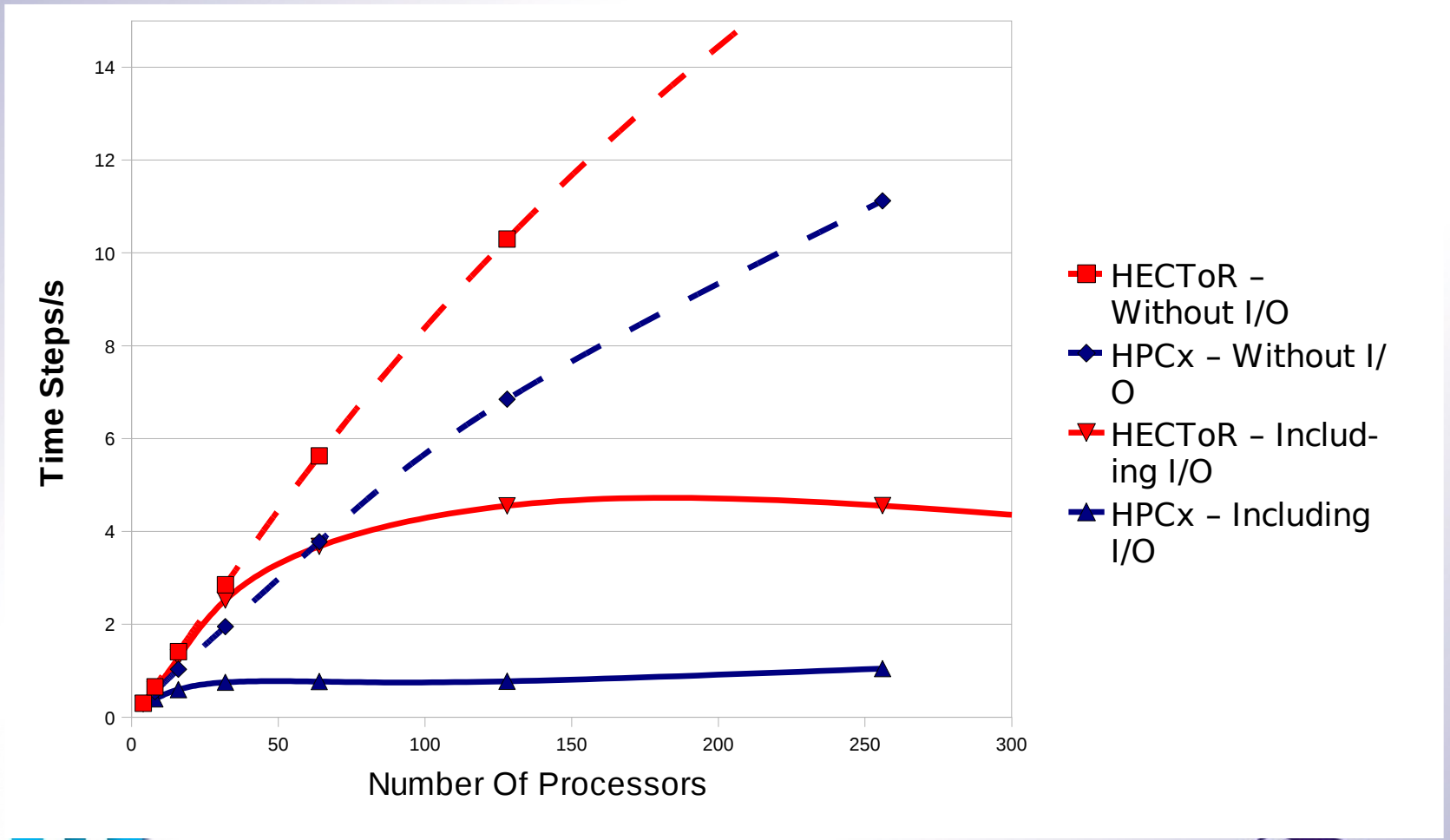
Overall the best of a very bad bunch seems MPIW, which became the default method in DL_POLY_3.

But how does this affect the scaling of the whole code?

To test used a benchmark of 216,000 ions of NaCl. Ran the job for 500 time steps, then dumped a single configuration. Chosen as representative of an “average” job for DL_POLY_3 (though probably slightly toward the tough end for this as the force field is so simple). Benchmark from now on unless otherwise stated.

All environment variables set to default values (because that is what the users will use)

The Scaling of MPIW



HECToR

RESEARCH COUNCILS UK

Can’t we do better ?

Can't we do better? Potentially two major problems.

3. All the processors are writing
   - Probably will exhaust the available I/O bandwidth
   - Might be a serialization somewhere
4. All the I/O transactions are very short
   - Each transaction writes the data for one atom only

To circumvent this a new method MPIW_SORTED based on MPIW was developed, and implementing this as production quality code is the first part of the dCSE.

MPIW_SORTED

- Gather the data onto a subset of the processors (the “I/O processors”)
- Perform a parallel, distributed sort of the data on the I/O processors into the original ordering
  - Index the local data (O((N/P)log(N/P)))
  - Work out which I/O processor this atom should be on
  - Redistribute among the I/O processors
  - Sort the new local data(O((N/P)log(P)) in principle - something like merge sort)
- As the data is now in order can write many records at once. Use MPI I/O as in MPIW for portability.

Rearrange the data on the CPUs rather than the disk

MPIW_SORTED

Though the method address the two problems above there are some downsides

2) Much increased communication

3) A very appreciable memory overhead – the data structures are no longer distributed across all the processors

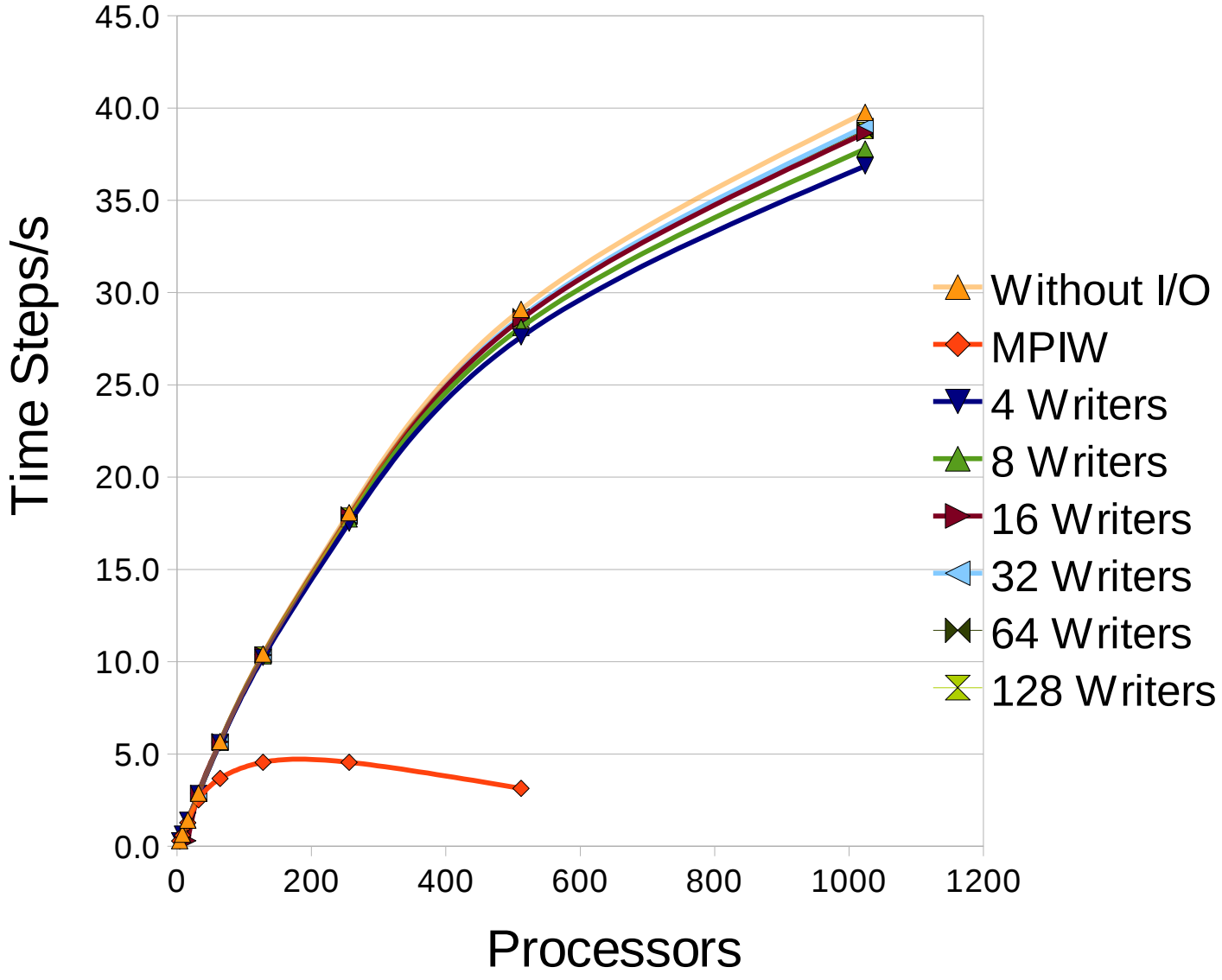
4) Some extra CPU work – but sorting/indexing is quick

First see how well it works, then discuss how we addressed the downsides

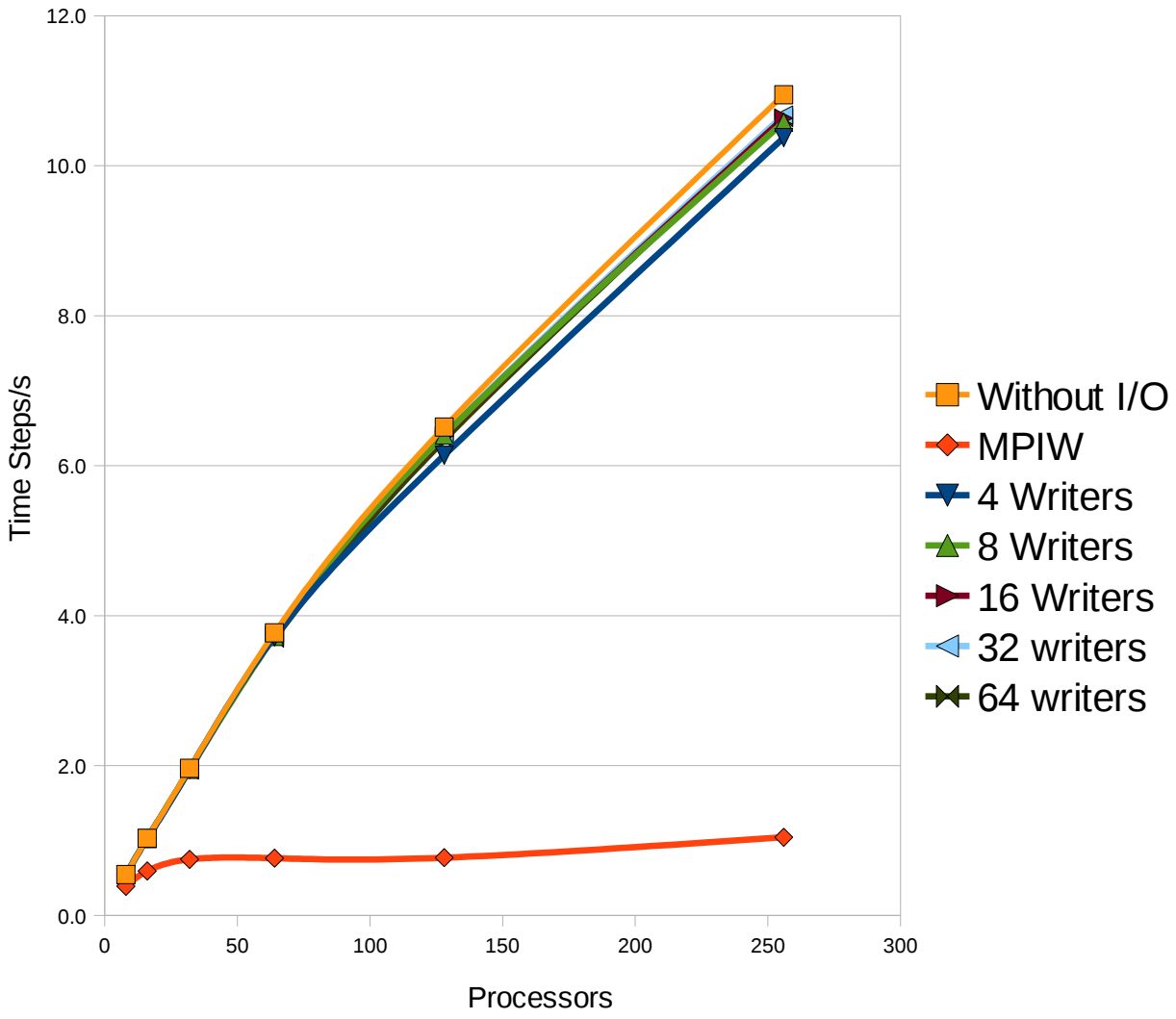
HECToR

RESEARCH COUNCILS UK

MPIW_SORTED on HECToR



MPIW_SORTED on HPCx



HECToR

RESEARCH COUNCILS UK

Why The Difference ?

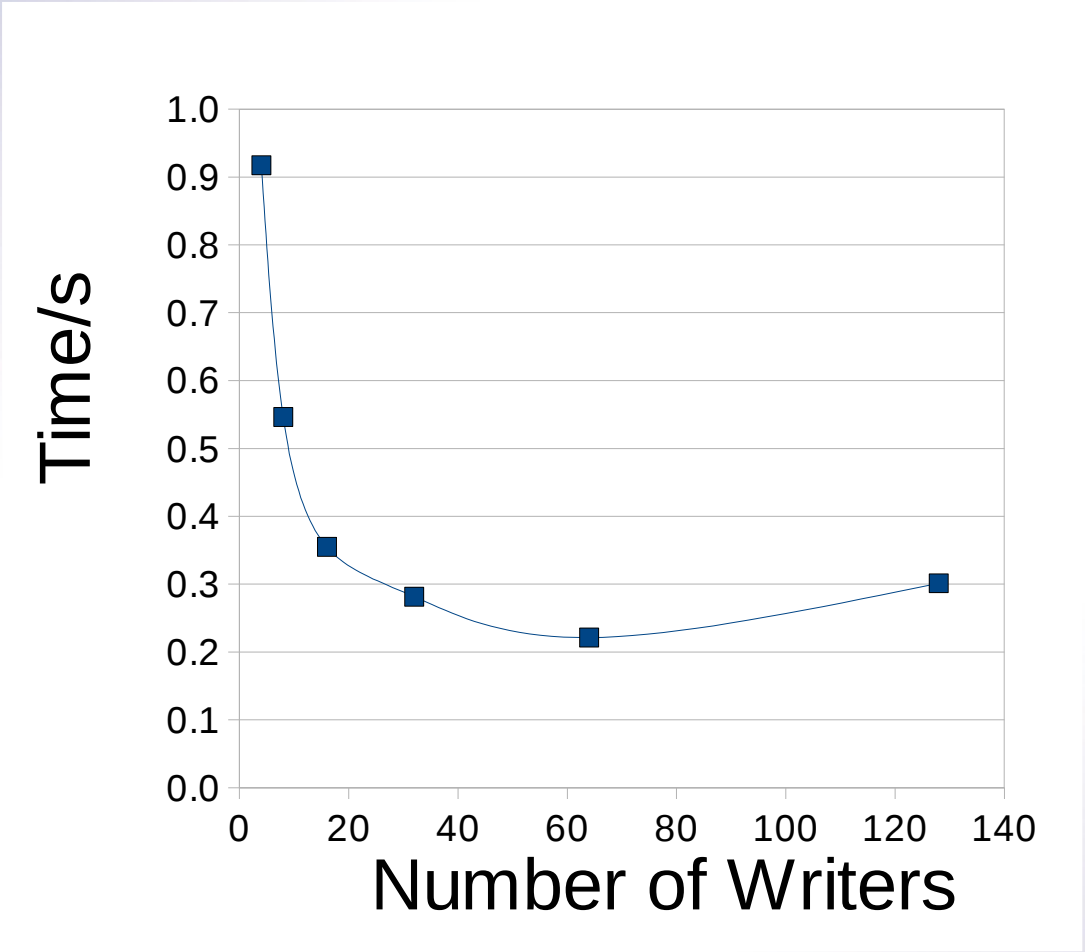
The new method spends 95%+ of it's time actually writing the file

- Communication costs are negligible
- Sorting costs are negligible

So the increased performance is due to more efficiently using the disks – the longer I/O transactions

So what is the the best number of writers ?

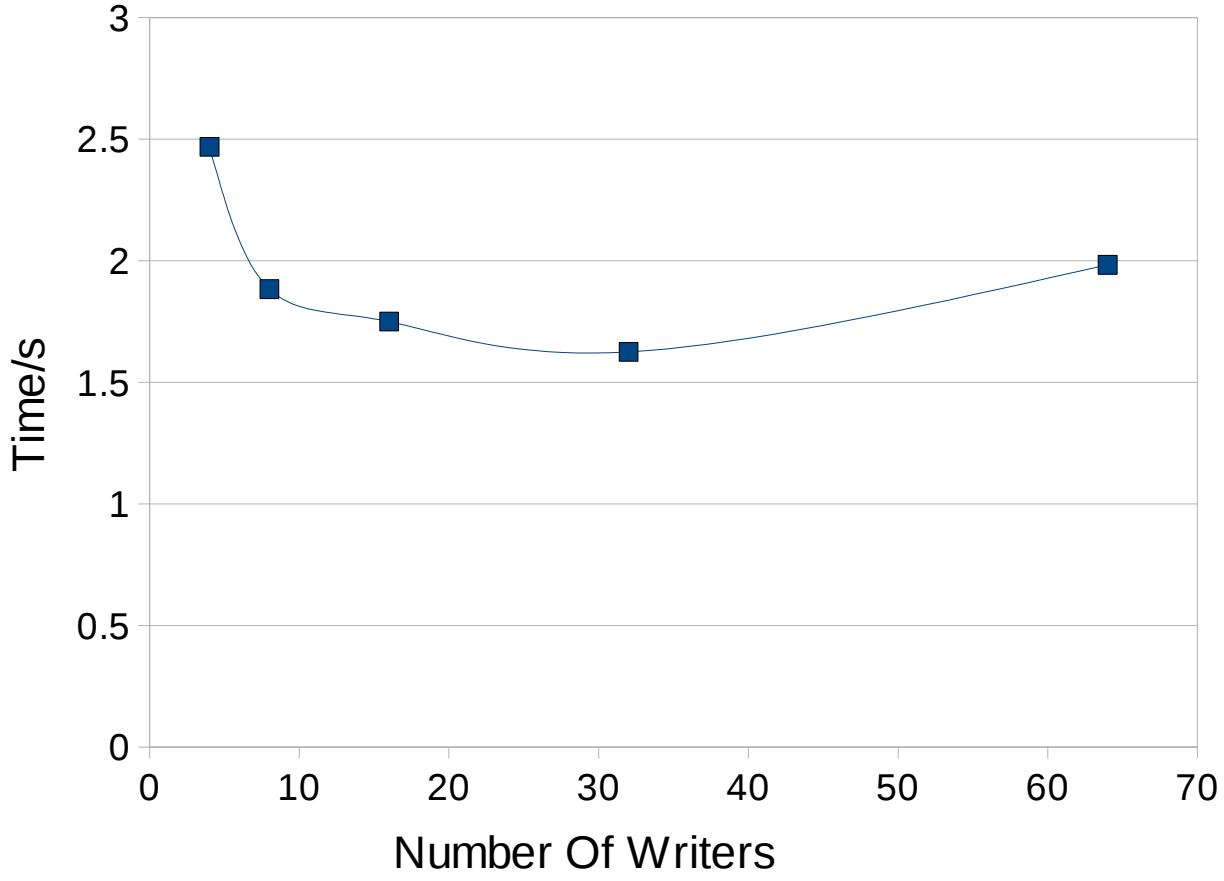
MPIW_SORTED on HECToR



HECToR

RESEARCH COUNCILS UK

MPIW_SORTED on HPCx



HECToR

RESEARCH COUNCILS UK

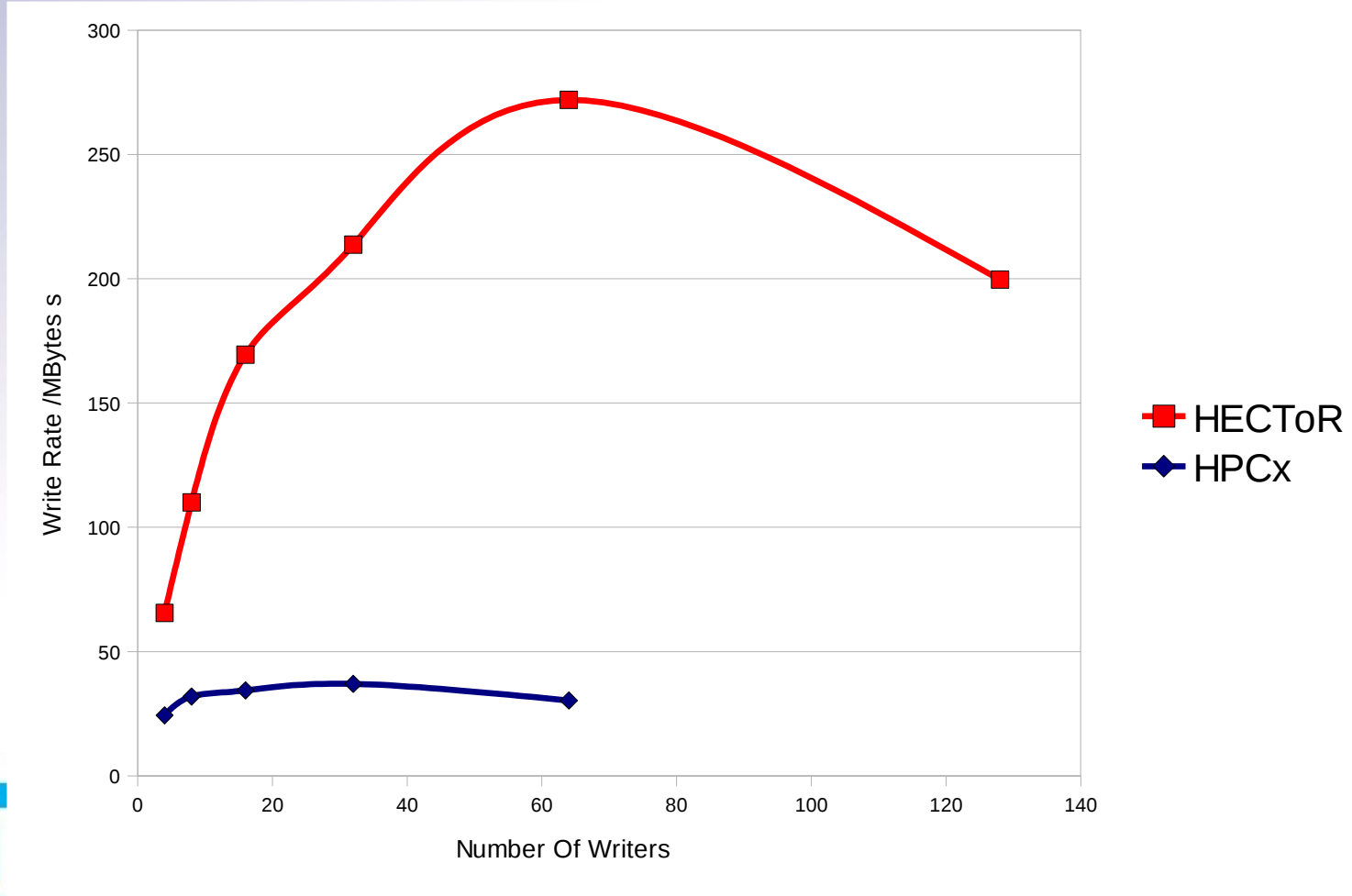
Best Number of Writers

- The best number of writers is 64 on HECToR and 32 on HPCx
- However the data is very noisy, as indicated by the error bars (but don't take them too seriously … small sample size)
- In practice it looks like a few tens of writers are the best, at least for this system
- What effective disk bandwidth are we getting ?

HECToR

RESEARCH COUNCILS UK

Disk Bandwidth



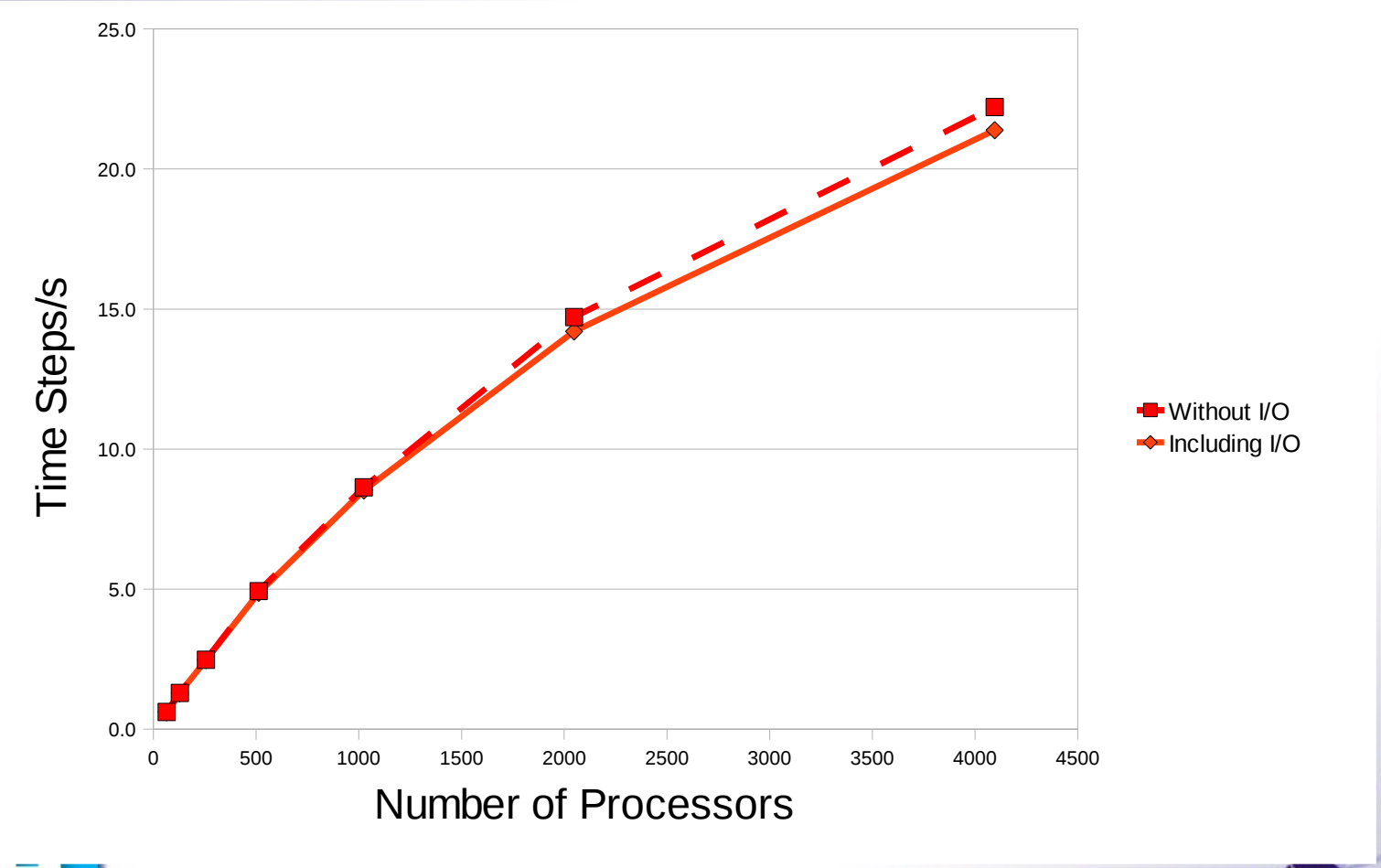
HECToR

RESEARCH COUNCILS UK

Disk Bandwidth

- Effective I/O bandwidth on HECToR very good, peaks at about 280 Mbyte s^{-1}
- Not so good on HPCx, around 40 Mbyte s^{-1}. Disappointing when compared with (the non-portable) PDAW which peaks at 120 Mbyte s^{-1}. However good enough for good scaling as shown before – the only benchmark figure of any importance is how well your whole code performs!
- How does the method scale with system size ? Take the benchmark and double it in each direction (so 8 times bigger overall). Only run on HECToR with 64 writers.

Scaling With System Size



HECToR

RESEARCH COUNCILS UK

Problems ?

The MPIW_SORTED method seems to solve the performance problem. However

- What about the memory overhead ?
  - This can be solved by writing out the data in batch
- But the I/O only scales to 64 processors – I have 132,000!
  - Much harder. Can do a bit more on software (binary formats, compression, collectives for writing) but hitting the barrier of what can be done portably. Ultimately hardware improvements are required.

Current Situation

- Batching implemented
- All (large) output is now done by the MPIW_SORTED method
  - Ouput on average 51.2 time quicker compared to MWRITE over the standard DL_POLY test cases
  - Quicker on all those cases but 1
- DL_POLY_3 picks sensible default values for the number of I/O processors and batch sizes
  - But the user may override these
- Writing now released to users
- Parallel reading now implemented using a similar method and in testing
  - Less important but harder than output
  - Over an order of magnitude quicker than before
- About to start on a netcdf version

Conclusions

- As disk access is so slloooow it may be worthwhile to completely reorganize your data simply to get the least bad performance out of the disks – we all program to minimize comms, same principles apply.
- Write long I/O transactions ! (This is news?)
- For the system sizes typically studied on HPCx and HECToR the MPIW_SORTED method effectively solves the I/O problem.
- However current I/O hardware does not scale to the whole system size – only can use a few tens of writers
- For large system sizes on larger numbers of processors the jury is out – how well does the I/O hardware (and system software) scale ?

HECToR

RESEARCH COUNCILS UK

More Details of The Method

More details in the HPCx technical report:

"DL_POLY_3 Parallel I/O Alternatives at Large Processor Counts", I.T. Todorov and I.J Bush

Which can be found at

http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0806.pdf

Acknowledgements

Thanks to

- Ilian Todorov for the code, many of the slides, helping me move house ...
- Andy Porter for NetCDF work and support
- Luican Anton and David Tanqueray for first draft of MPI-I/O writing routine

http://www.ccp5.ac.uk/DL_POLY/