HPC at EPCC

The University of Edinburgh

Adrian Jackson
Technical Architect
a.jackson@epcc.ed.ac.uk

- Edinburgh Parallel Computing Centre
  - founded in 1990 at the University of Edinburgh
- Now an Institute within the School of Physics and Astronomy
- High performance and novel computing for academia and industry

James Clerk Maxwell Building, The King's Buildings
© Visual Resources, The University of Edinburgh

EPCC

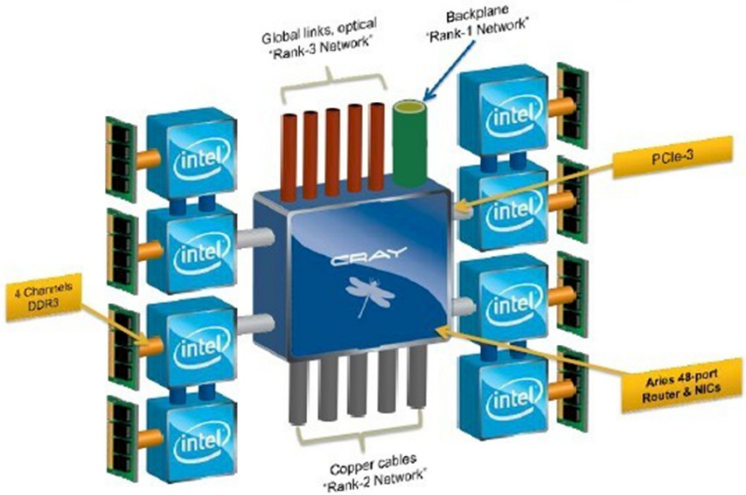
- EPCC is the HPC Centre of the University of Edinburgh
- Vital statistics:
  - ~75 staff
  - ~£4M turnover from external sources
- Multidisciplinary and multi-funded
  - with a large spectrum of activities … and a critical mass of expertise
- Supports research through:
  - Access to facilities
  - Training courses
  - Visitor programmes
  - Collaborative projects

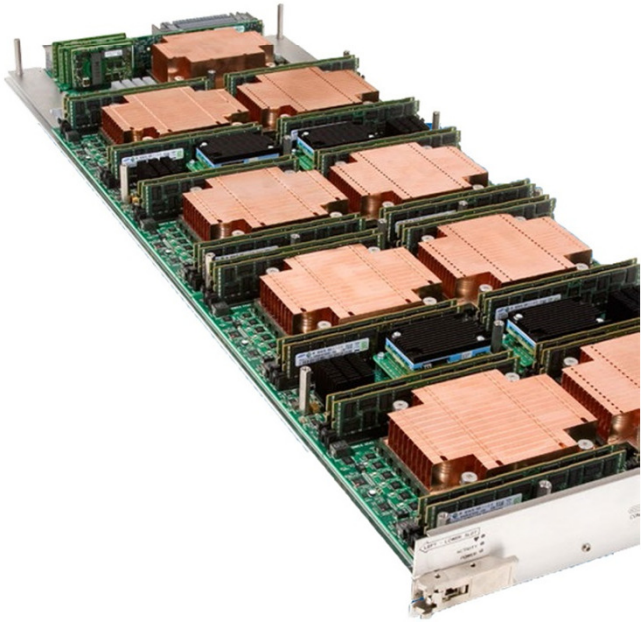
Facilities: HECToR

- UK National HPC Service
- Currently 30-cabinet Cray XE6 system
  - 90,112 cores
- Each node has
  - 2 × 16-core AMD Opterons (2.3GHz Interlagos)
  - 32 GB memory
- Peak of over 800 TF and 90 TB of memory

ARCHER

- Next generation UK National Service
  - Cray XC30
  - 1.6 Pflop/s machine
  - 16 cabinets
    - 376 nodes per cabinet
    - 2 cabinets high memory
  - Node:
    - 2 x Intel Xeon E5-2697 processor, 12-cores 2.7 GHz
    - 64 GB DDR3 1866 MHz
  - High memory nodes:
    - 2 x Intel Xeon E5-2697 processor, 12-cores 2.7 GHz
    - 128 GB DDR3 1866 MHz





Facilities: DiRAC and Indy

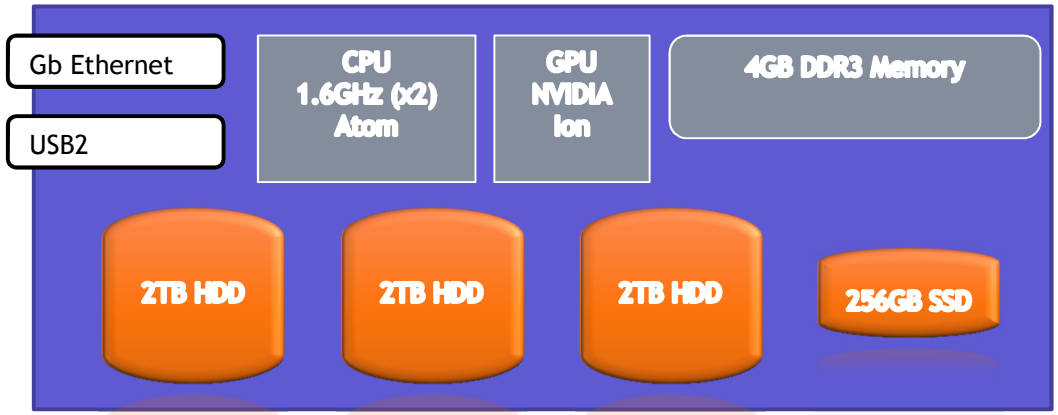
- Indy: Linux and Windows HPC cluster
  - 1536 cores
  - 24 nodes: 64 cores, 256GB
  - 128 cores
  - 2 nodes: 64 cores, 512GB
  - Commercial usage focus
    - No job length or queue restrictions
- DiRAC: BlueGene/Q
  - 6144 compute nodes
  - 98304 compute cores
  - 1.26PFlop/s

Facilities: EDIM1

- A machine for Data Intensive Research
- Commissioned by EPCC & Informatics
- Designed for I/O-intensive applications
  - Use commodity components
  - Combine them in a novel way
  - Use cheap low-power processors

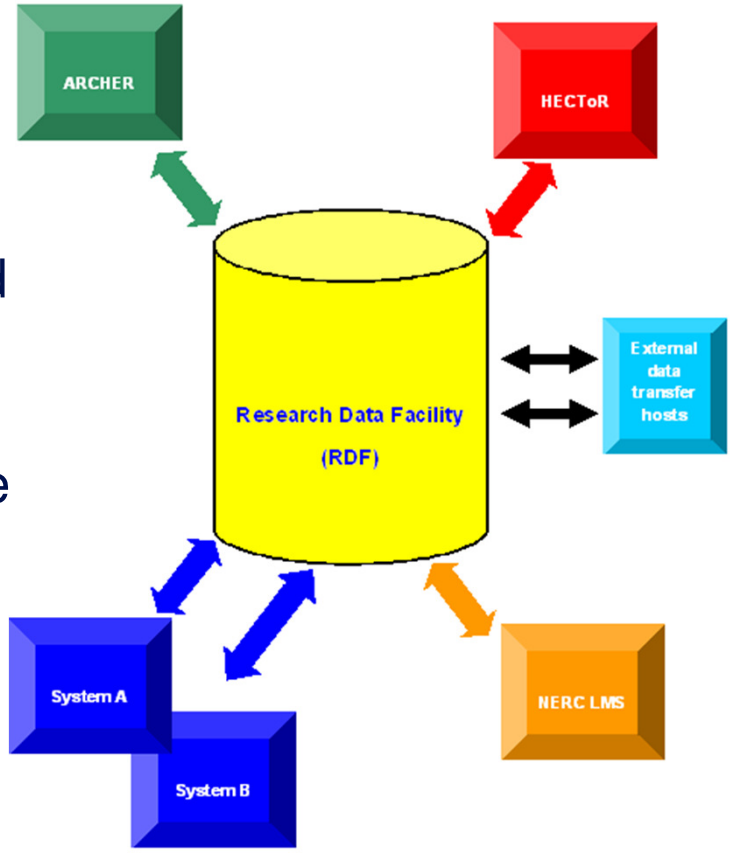
Facilities: EDIM1

- EDIM1
  - 120 nodes
  - Dual-Core Intel 1.6 GHz ATOM processor
  - NVIDIA ION GPU
  - 1 x 256 MB SSD
  - 3 x 2TB HDD
  - Data staging node for hot-plugging SATA hard disks for direct data upload



Facilities: UK Research Data Facility

- RDF is designed for long term data storage
- RDF consists of
  - 7.8PB disk
  - 19.5 PB backup tape
  - Provide a high capacity robust file store;
  - Persistent infrastructure - will last beyond any one national service;
  - Will remove end of service data issues - transfers at end of services have become increasingly lengthy;
  - Will also ensure that data from the current HECToR service is secured

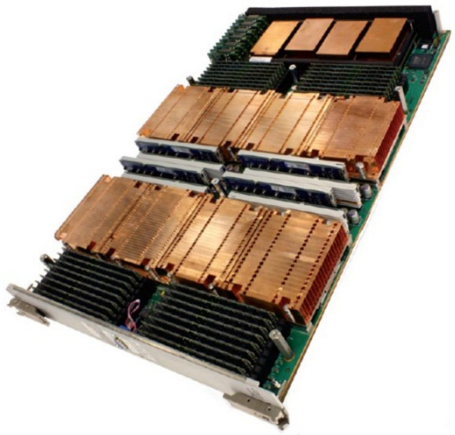


Other Facilities

- GPU testbed

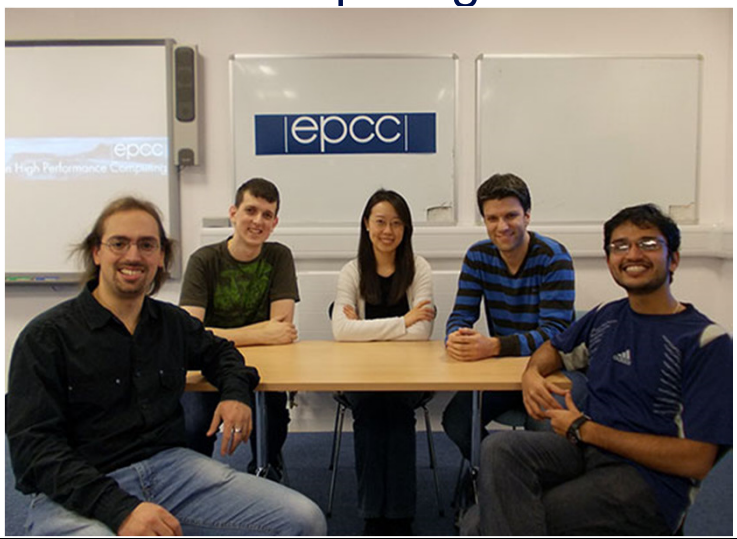
GPGPUs
NVIDIA Fermi C2050
NVIDIA Fermi C2070
AMD FireStream 9270
NVIDIA K20

- Hydra
  - NVIDIA Fermi GPGPUs
  - Intel Xeon Phi
  - NVIDIA K20's



Training

- MSc in HPC
  - One year masters, 30-50 students annually
  - Message-Passing Programming
  - Shared-Memory Programming
  - HPC Architectures
  - Programming Skills
  - Software Development
  - Threaded Programming
  - Parallel Numerical Algorithms
  - HPC Ecosystem
  - Parallel Design Patterns
  - Performance Programming
  - Parallel Programming Languages
  - Dissertation/Project (3 Months)
- Short courses in parallel and novel computing
  - Academia and Industry
- PhD Students



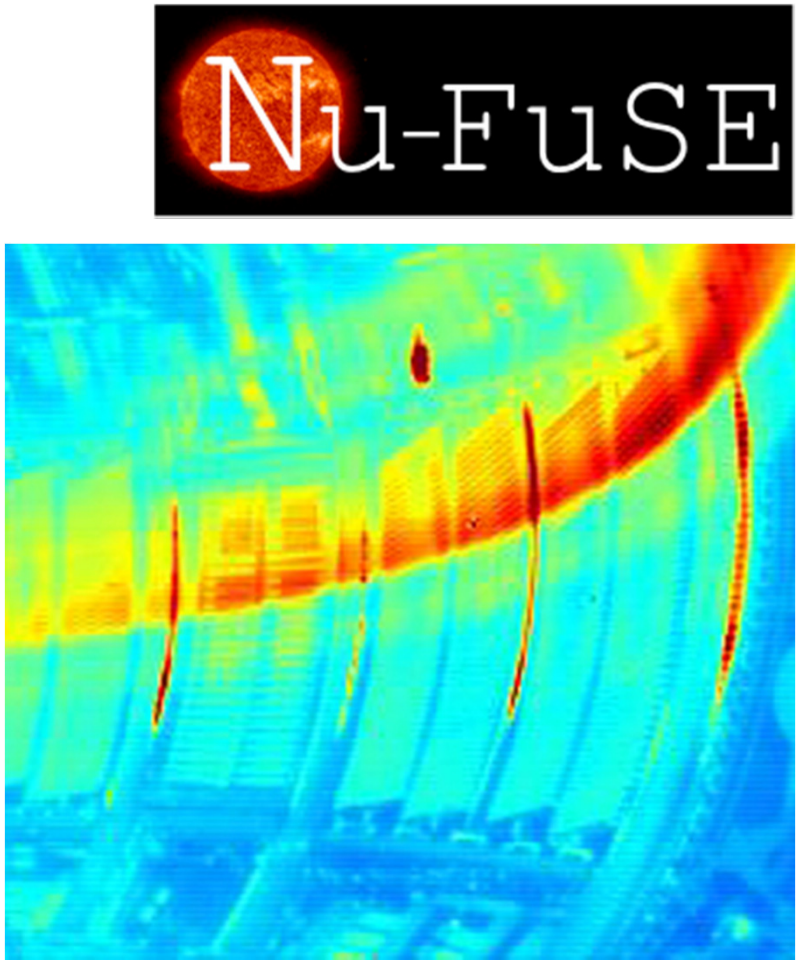
Projects


epcc
PRACE
Nu-FuSE
EUDAT
SUPERCOMPUTING SCOTLAND
BUSINESS INNOVATION THROUGH HIGH PERFORMANCE COMPUTING
Software Sustainability Institute
Rolls-Royce
JARDINE
ASSET & RISK MANAGEMENT SOLUTIONS
UKAEA
Restoring our Environment
NAIS
Centre for Numerical Algorithms & Intelligent Software
Golden Crumb
C&G Cheltenham & Gloucester
CREST
Kwik-Fit
IN DEFENS
BAE SYSTEMS
FUJITSU
dti
Unilever
Met Office
ARRAN AROMATICS
BonFIRE

Nu-FuSE: Nuclear Fusion Simulation at Exascale

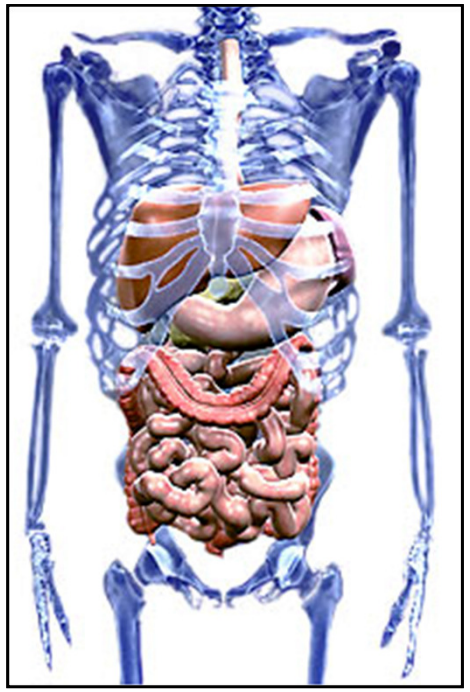
http://www.nu-fuse.com

- Nu-FuSE is an international project (funded through the G8 Research Councils Initiative on Multilateral Research Funding) looking to significantly improve computational modelling capabilities to the level required by the new generation of fusion reactors.
- The focus is on three specific scientific areas:
  - fusion plasma
  - the materials from which fusion reactors are built;
  - physics of the plasma edge
- This will require computing at the “exascale” level across a range of simulation codes, collaborating together to work towards full integrated fusion tokamak modelling.

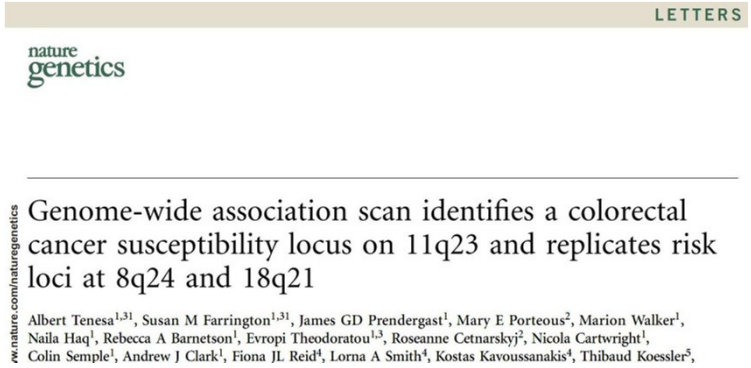


Example: Oncology

- Aim: investigate genetic causes of bowel cancer
- Collaborative project between EPCC and the Colon Cancer Genetics Group (CCGG)
- Vast amount of data
  - Over 500,000 genetic markers from 2000 people
- Two-stage study
- Stage 1: investigated effect of each individual marker
  - Required ~565,000 computations, O(N) problem
  - Predicted serial runtime ~4 months on a single cpu
  - Parallel code took 6.5 hours on 128 processors



(www.sanofi-aventis.com)



LETTERS

nature genetics

Genome-wide association scan identifies a colorectal cancer susceptibility locus on 11q23 and replicates risk loci at 8q24 and 18q21

Albert Tenesa[1,31], Susan M Farrington[1,31], James GD Prendergast[1], Mary E Porteous[2], Marion Walker[1], Naila Haq[1], Rebecca A Barnetson[1], Evropi Theodoratou[1,3], Roseanne Cetnarskyj[2], Nicola Cartwright[1], Colin Semple[1], Andrew J Clark[1], Fiona JL Reid[4], Lorna A Smith[4], Kostas Kavoussanakis[4], Thibaud Koessler[5],

Example: Oncology

- Stage 2: investigated interactions between the gene markers
  - Every pair of markers must be tested
  - $O(N^2)$ problem

$$\frac{565{,}000 \times 565{,}000}{2} = 1.5 \text{ billion gene interactions!}$$

  - Key challenges: **runtime, memory, scaling & sorting**

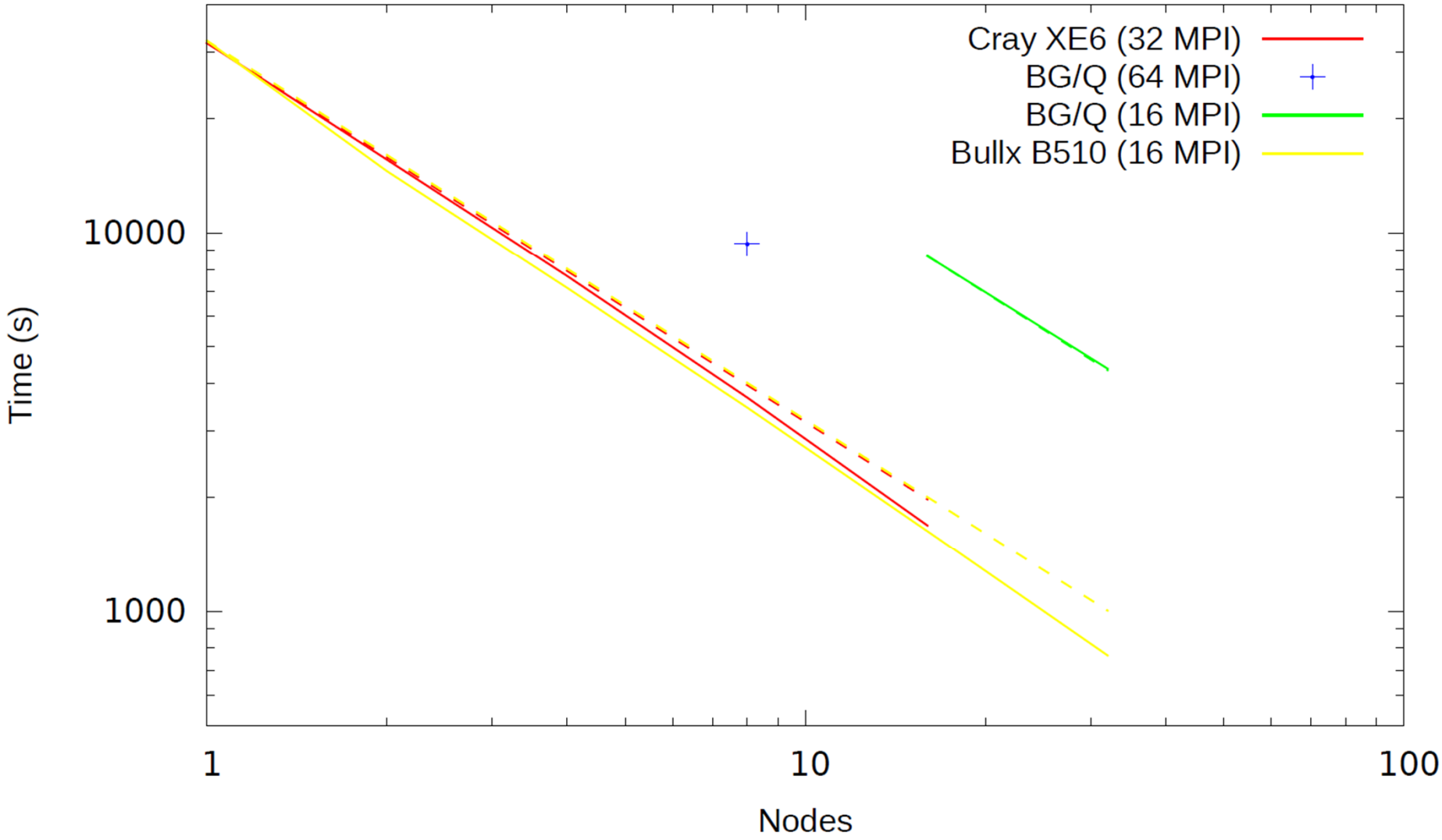
Oncology

- **Runtime**
  - code expected to take 400 days, optimisation reduced this to 130 days but still too long
  - Need a parallel code
- **Memory**
  - Impossible to fit all the data into memory
  - However, we only actually need 5% of the results
- **Scaling**
  - 2D decomposition used with a “task farm”
  - More chunks than processors
- **Sorting**
  - Parallel sorting algorithm used
- Computed interactions between all pairs of markers
  - $565{,}000^2$ computations
- Runtime reduced from 400 days to 5 hours on 512 CPUs on HECToR
- 8.5×10^9 (192GB) probability values obtained
- Sorting performed in 5 minutes

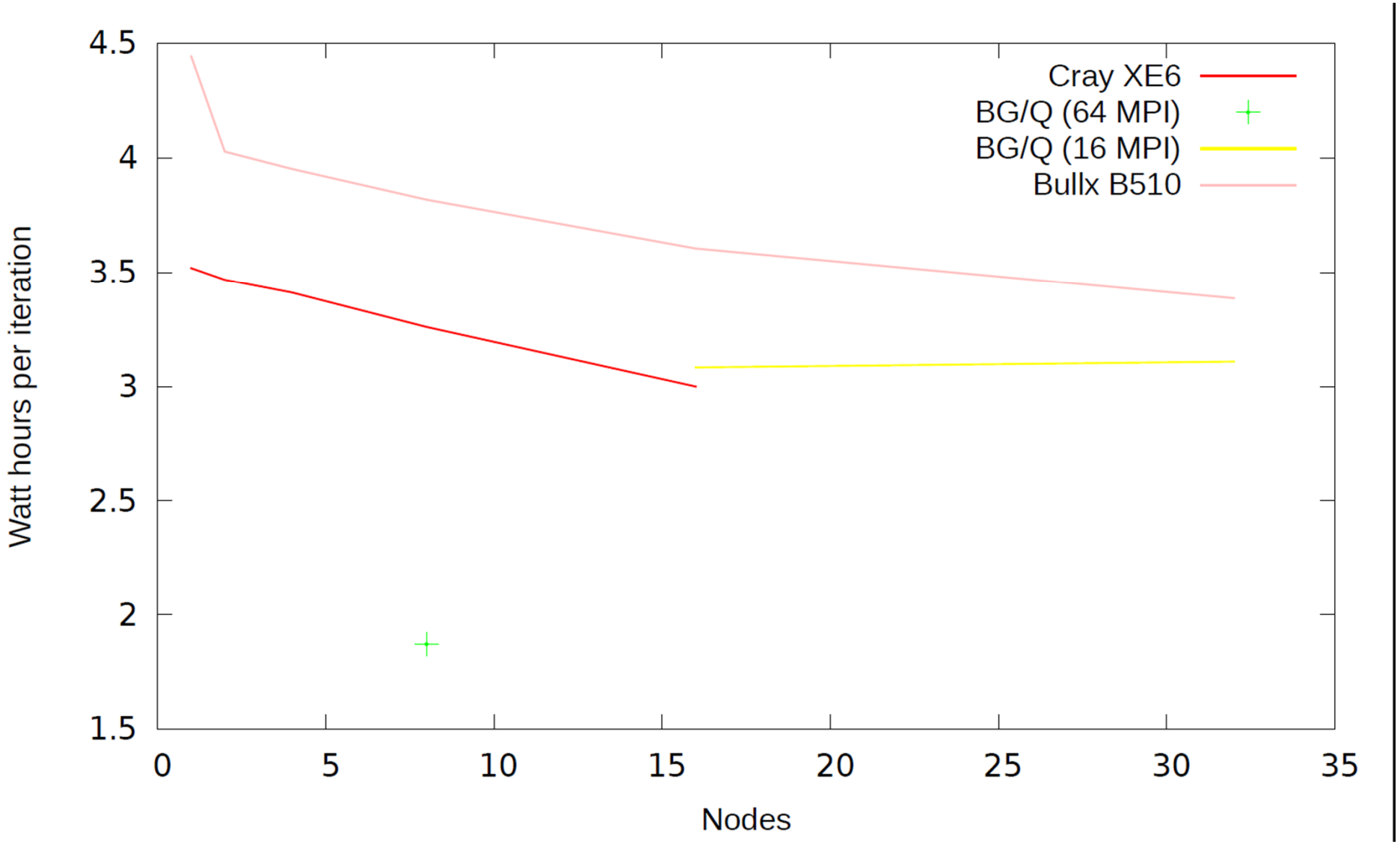
Example: CFD Code COSA

- COSA CFD simulation code
  - Serial code
  - M. Sergio Campobasso author
  - FORTRAN code
  - Currently 2D, 3D under development
  - Demonstrate performance of harmonic balance vs time domain for periodic unsteady flows
- Single code incorporates multiple functionality
  - Euler, viscous laminar and turbulent equations
  - Structured multi-block solver using explicit multigrid integration
  - Time-domain solution based on dual-time-stepping
  - Harmonic Balance solver with variable number of harmonics
- Serial code required significant computational time
  - Weeks for a reasonable dataset
  - Code developed using multi-block format

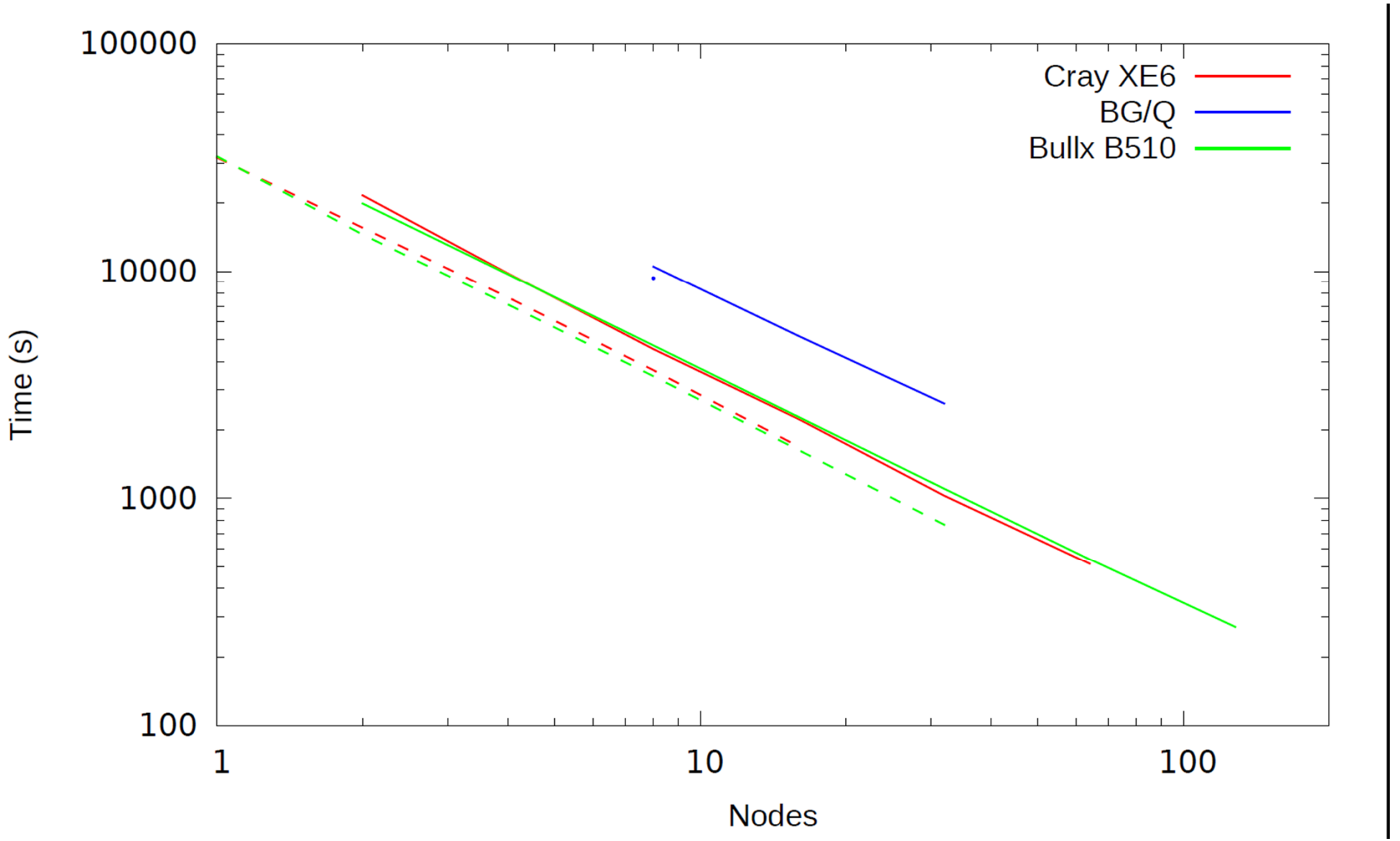
Pure MPI Performance



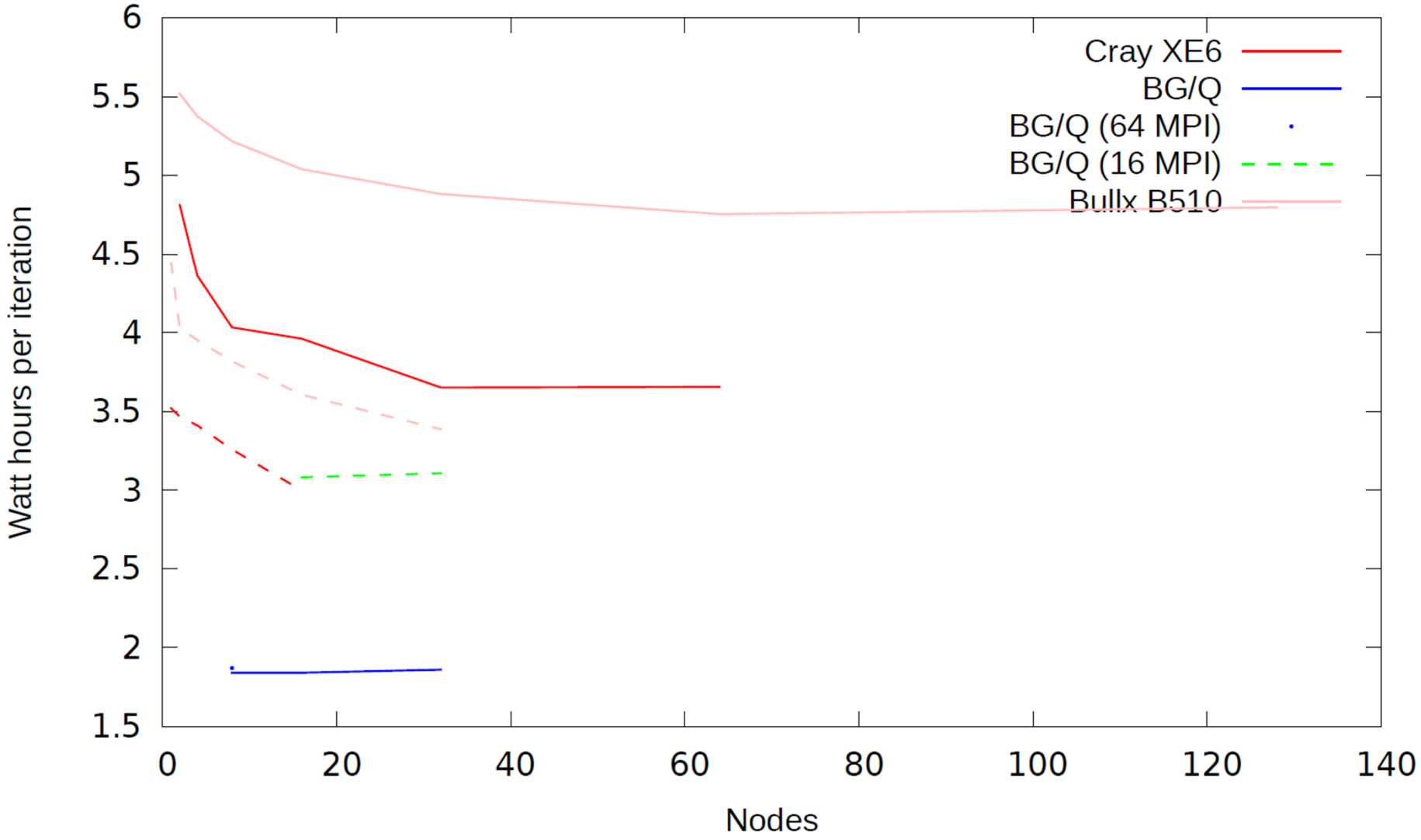
Efficiency – Power used



Hybrid Parallelisation



Hybrid Efficiency

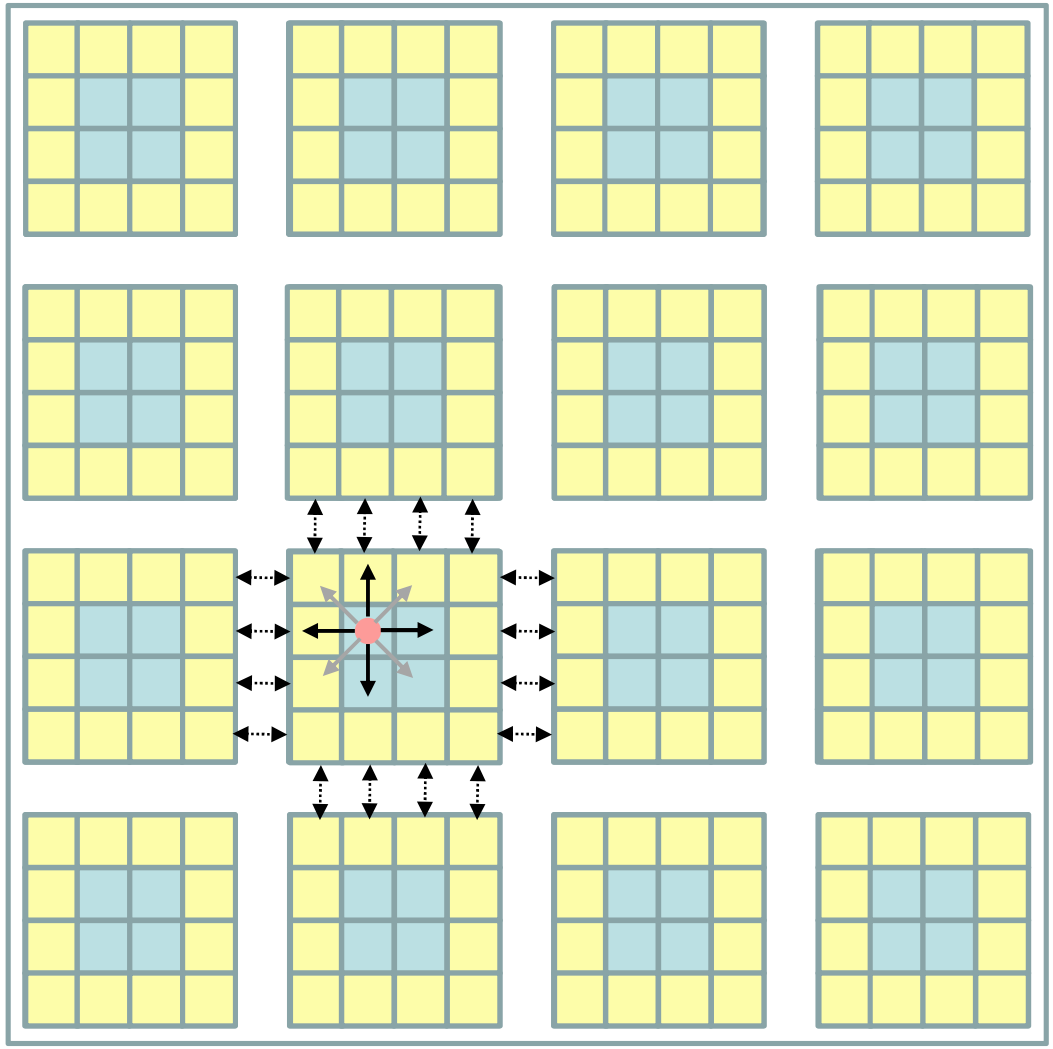


Example: MDMP

- New parallel programming approach
  - Optimise parallel performance
    - Assuming computing free, memory expensive, communication very expensive
  - Optimising communications
    - Similar to memory optimisations
    - High communication latency
    - Cache or pre-fetch optimisations
    - Not quite as simple
- Parallel Programming Challenges
  - Work with MPI based codes
    - Generate MPI calls for parallelism
  - Enable easy new development
    - Directives based/Compile based implementation
  - Provide framework for optimisation communications
    - Runtime monitoring and scheduling of communications
    - Enable different communication layers to be used

Optimise standard patterns

- Standard computational and communication
- Calculate over array or matrix
- Communicate halo data at the end of the calculations



MDMP Fundamentals

```
#pragma send(…)

#pragma recv(…)
```

```
for (iter=1;iter<=maxiter; iter++){
#pragma recv(old[0][0], NP, prev)
#pragma recv(old[MP+1][1], NP, next)
#pragma send(old[MP][1], NP, next)
#pragma send(old[1][1], NP, prev)
  for (i=1;i<MP+1;i++){
            for (j=1;j<NP+1;j++){
                new[i][j]=0.25*(old[i-1][j]+old[i+1][j]+
                    old[i][j-1]+old[i][j+1] - edge[i][j]);
            }
        }
        for (i=1;i<MP+1;i++){
            for (j=1;j<NP+1;j++){
                old[i][j]=new[i][j];
            }
        }
  }
```

MDMP Fundamentals

- Communicating Regions

```
#pragma commregion
#pragma commregionfinished
```

- Defines the scope to consider for the data evaluation and communication optimisation
  - Count read and writes of data to be communicated
  - Communicate once; last write occurs (sends), last read/write occurs (receives)

MDMP Fundamentals

```
#pragma commregion
for (iter=1;iter<=maxiter; iter++){
#pragma recv(old[0][0], NP, prev)
#pragma recv(old[MP+1][1], NP, next)
#pragma send(old[MP][1], NP, next)
#pragma send(old[1][1], NP, prev)
  for (i=1;i<MP+1;i++){
            for (j=1;j<NP+1;j++){
                new[i][j]=0.25*(old[i-1][j]+old[i+1][j]+
                   old[i][j-1]+old[i][j+1] - edge[i][j]);
            }
        }
        for (i=1;i<MP+1;i++){
            for (j=1;j<NP+1;j++){
                old[i][j]=new[i][j];
            }
        }
  }
#pragma commregionfinished
```

MDMP Fundamentals

- Combined these enable runtime scheduling of communications to optimise network usage
  - Auto-tuning possible
  - Potential for errors
  - Adds computational overheads
  - May damage memory performance
  - Relies on users placing communicating regions correctly

MDMP Performance

- STREAMS benchmarks
  - Communicating region

Operation	Original Time	MDMP Time	Optimised MDMP Time
Int Assign	0.000007	0.000189	0.000048
Db Assign	0.000009	0.000173	0.000057
Db Copy	0.000009	0.000300	0.000066
Db Scale	0.000017	0.000297	0.000063
Db Add	0.000035	0.000427	0.000071
Db Triad	0.000035	0.0004333	0.000076

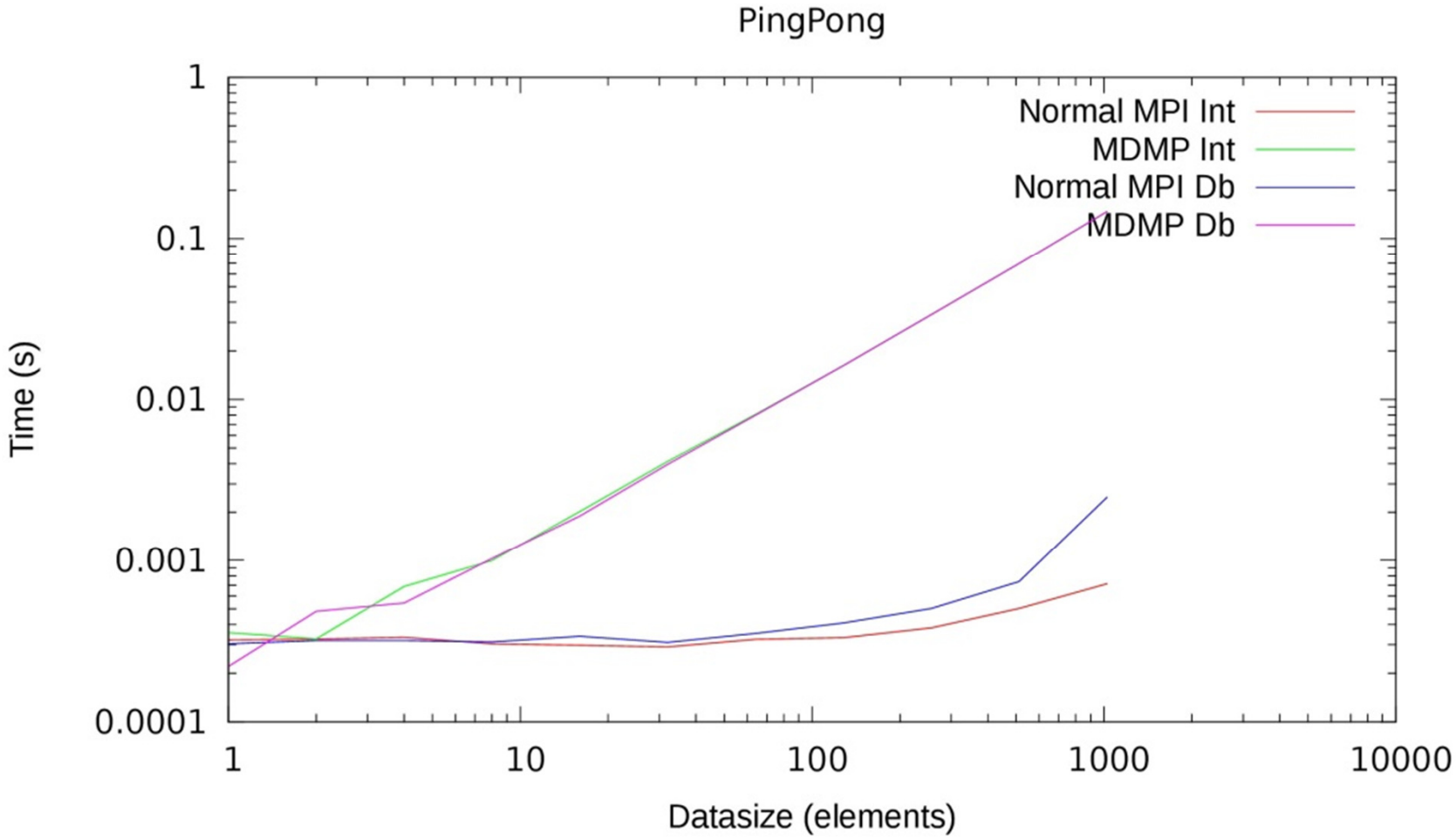
MDMP Performance

- STREAMS benchmarks
  - No communicating region

Operation	Original Time	MDMP Time	Optimised MDMP Time
Int Assign	0.000007	0.000113	0.000020
Db Assign	0.000009	0.000083	0.000024
Db Copy	0.000009	0.000128	0.000027
Db Scale	0.000017	0.000167	0.000025
Db Add	0.000035	0.000196	0.000033
Db Triad	0.000035	0.000195	0.000037

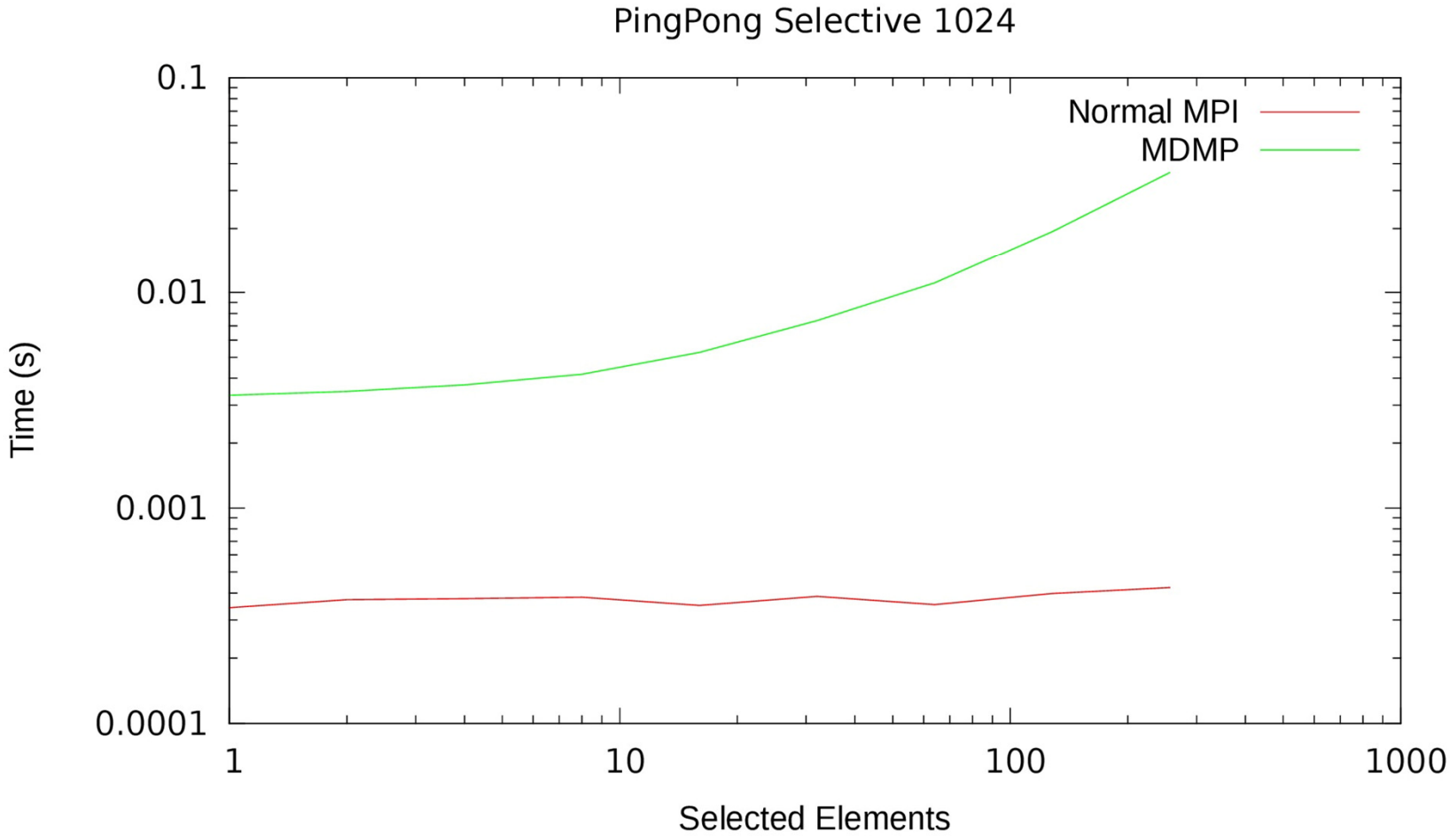
MDMP Performance

- PingPong benchmark



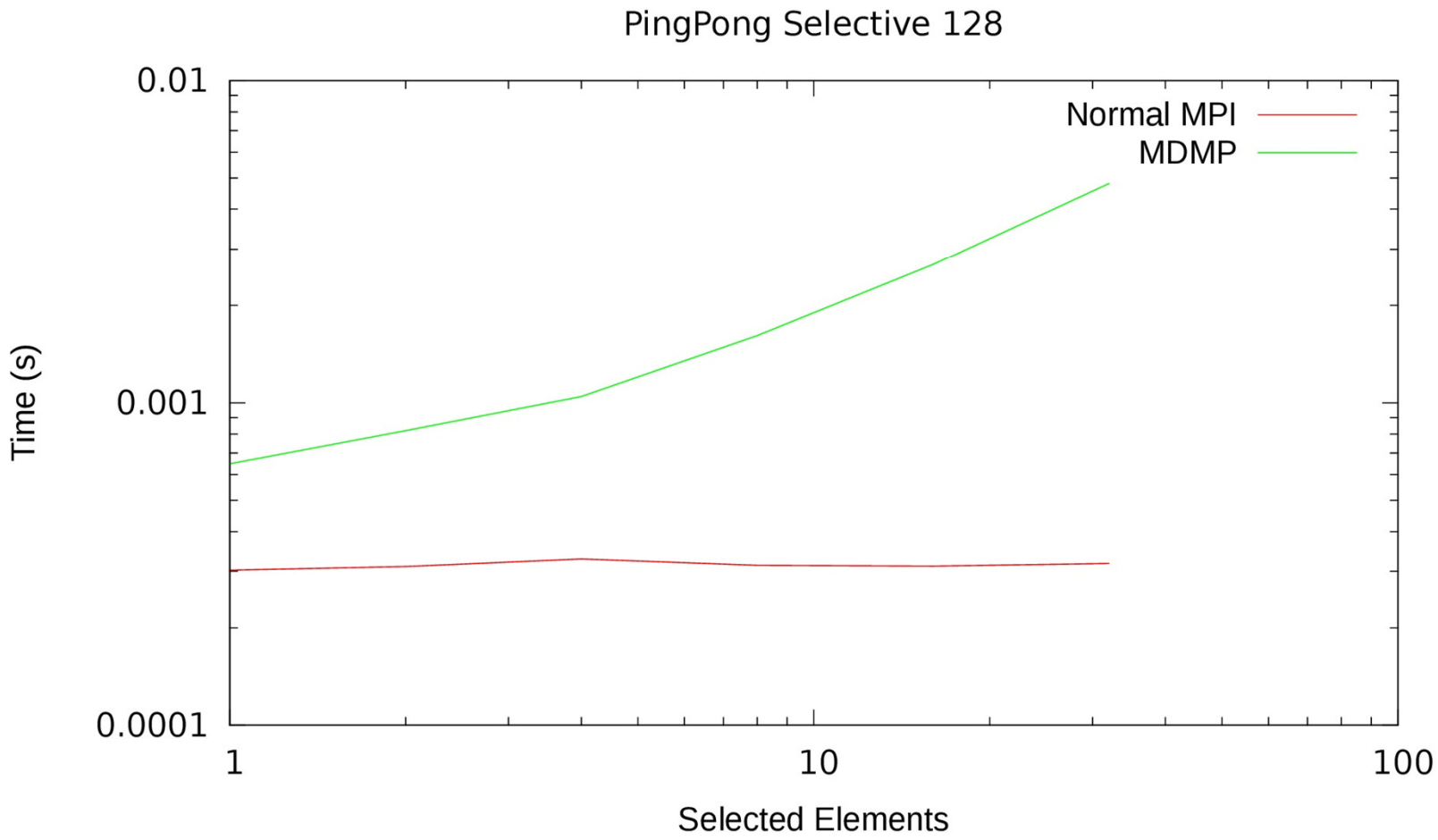
MDMP Performance

- Selective PingPong benchmark



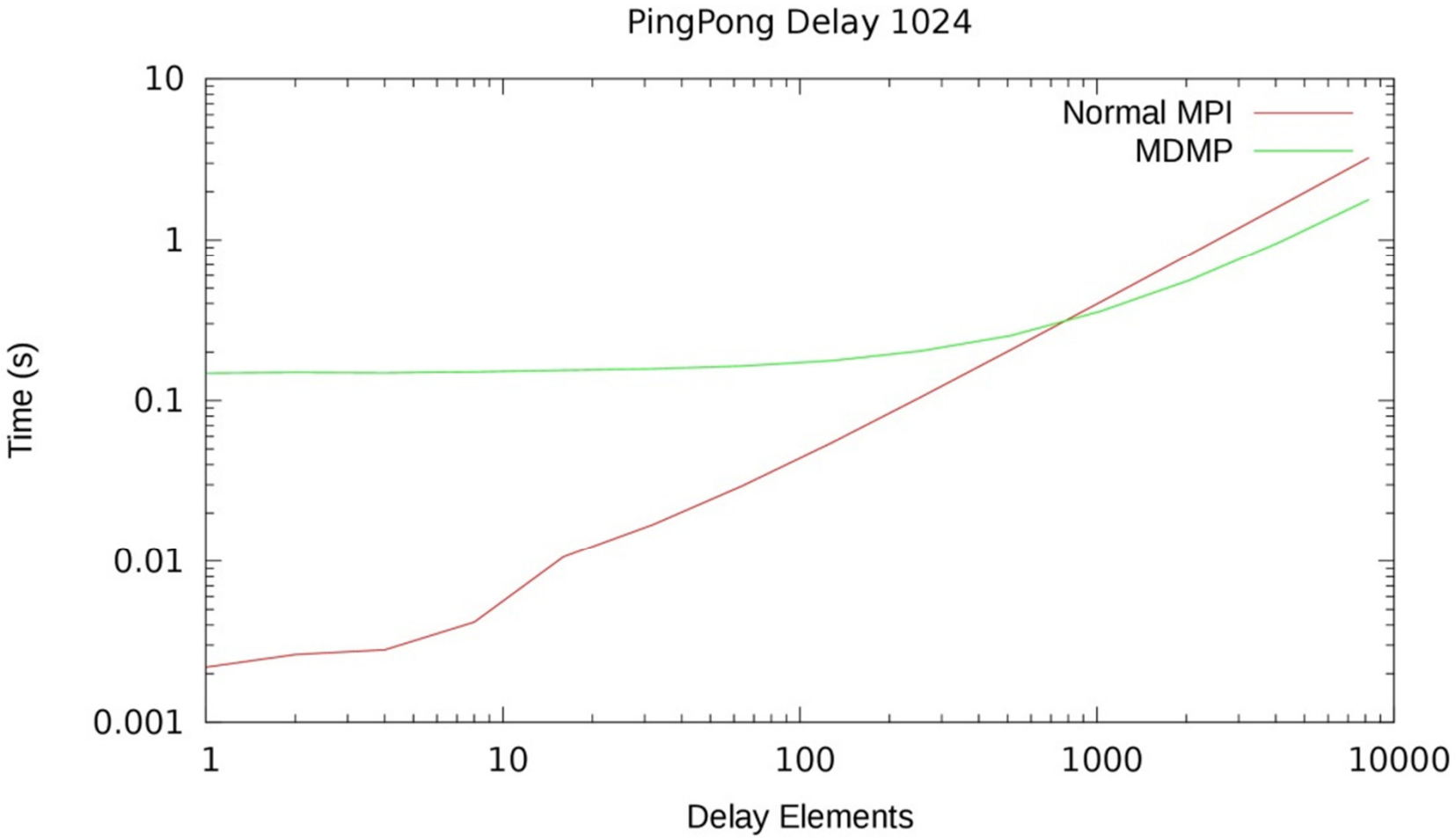
MDMP Performance

- Selective PingPong benchmark



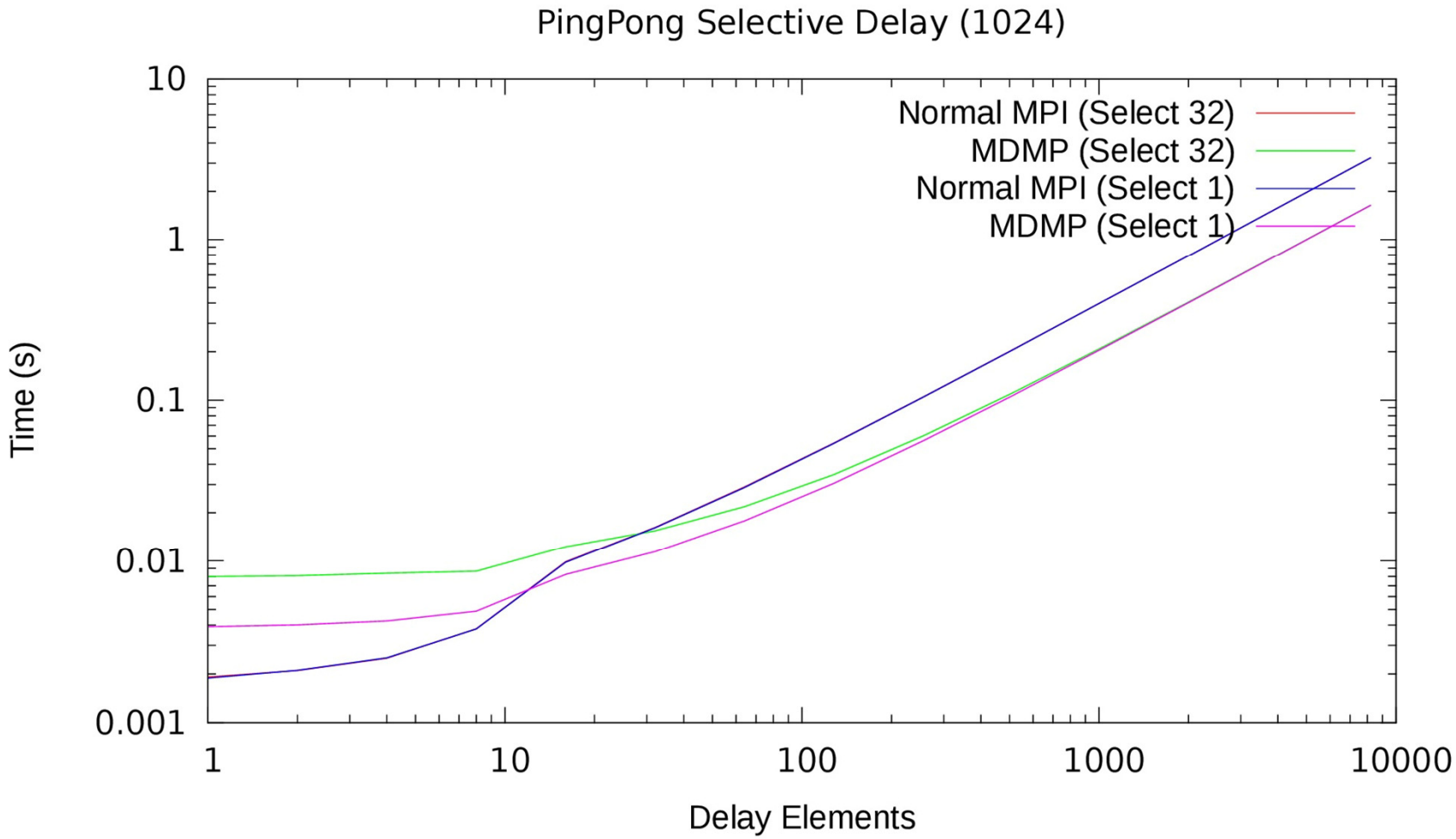
MDMP Performance

- Delay PingPong benchmark



MDMP Performance

- Delay selective PingPong benchmark



Summary

- MDMP
  - Simplified MPI programming
  - Aims to optimise communication
    - Automating evaluation and scheduling of communications
  - Won't work for everything
  - Won't necessarily give best performance
  - Requires more memory
  - Can default to mapping to MPI only
  - Can just use MPI
    - Selective optimisations after development

PGAS2013

7th International Conference on PGAS Programming Models

3rd & 4th October 2013, Edinburgh, Scotland, UK

Partitioned Global Address Space (PGAS) programming models offer a shared address space model that simplifies programming while exposing data/thread locality to enhance performance. The PGAS conference is the premier forum to present and discuss ideas and research developments in the area of: PGAS models, languages, compilers, runtimes, applications and tools, PGAS architectures and hardware features.

- **Applications**. New applications that are uniquely enabled by the PGAS model, existing applications and effective application development practices for PGAS codes.
- **Performance**. Analysis of application performance over various programming models.
- **Developments in Programming Models and Languages**. PGAS models, language extensions, and hybrid models to address emerging architectures, such as multicore, hybrid, heterogeneous, SIMD and reconfigurable architectures.
- **Tools, Compilers, and Implementations**. Integrated Development Environments, performance analysis tools and debuggers. Compiler optimisations for PGAS languages, low level libraries, memory consistency models. Hardware support for PGAS languages, performance studies and insights, productivity studies, and language interoperability.

Contact details

EPCC, The University of Edinburgh

JCMB, Mayfield Road, Edinburgh EH9 3JZ

+44 131 650 5022

info@epcc.ed.ac.uk

http://www.epcc.ed.ac.uk/