HECToR

HIGH END COMPUTING TERASCALE RESOURCE

A Research Councils UK High End Computing Service

Using the XT6: case studies

Numerical Algorithms Group Ltd, HECToR CSE

XT6 Workshop

13th October 2010

Outline

- Micro-benchmarking
- CASINO
- GloMAP mode MPI
- CABARET and Incompact3D
- DL_POLY
- CASTEP

Micro-benchmarking tests

Chris Armstrong

HECToR

RESEARCH COUNCILS UK

Questions

1. What is the performance penalty for accessing data stored in non-local memory?
2. What is the effect of having a single link (which is the same as in the XT4) to the interconnect?

Micro-benchmarking tests to help understand the architecture...

Testing non-local memory accesses

- Looked at two ways of accessing non-local memory:
  - MPI
    - Cray's MPI library performs intra-node communications using shared memory.
    - Indicates the performance of accessing fixed-size chunks of data.
    - Should say something about latency and bandwidth of HT links within the node.
  - OpenMP
    - Data is moved as needed by the application.
    - No direct control over "message" size.
    - Should say something about the performance of direct memory addressing.

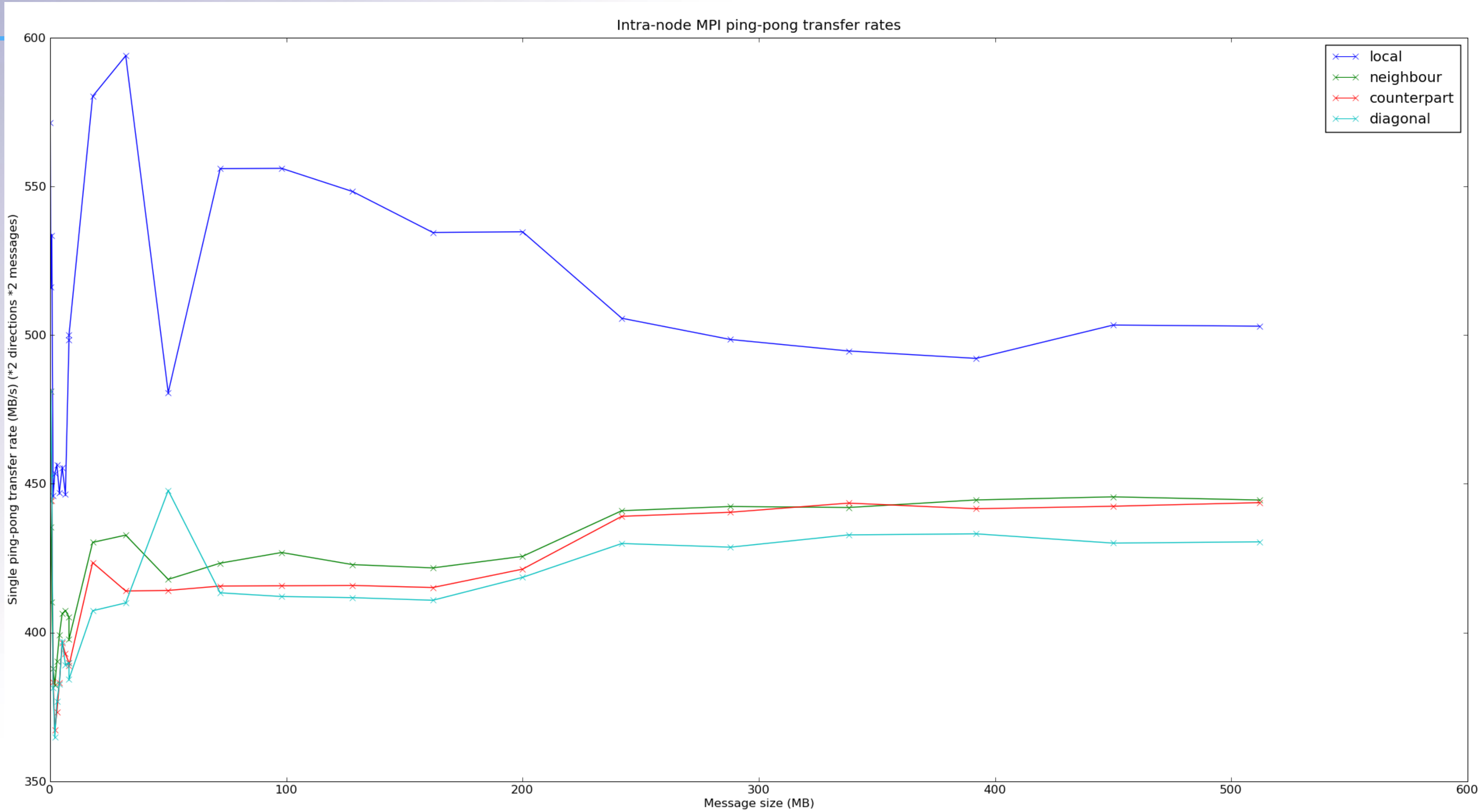
MPI Test

- Time 100 MPI_Sendrecvs for a range of message lengths using the hierarchy of intra-node communication:
  - In die 0:
    - 2 sendrecv messages to a core in the neighbouring die.
    - 2 sendrecv messages to a core in the counterpart die (of the neighbouring socket).
    - 2 sendrecv messages to a core in the diagonally opposite die.
  - All cores not involved in one of the above communications send messages to self (i.e. timing local memory accesses).

HECToR

RESEARCH COUNCILS UK

MPI Test: ping-pong transfer rates (averages)



- Clear benefits to accessing data locally.
- Non-local differences comparatively small.
- Results roughly follow the hierarchy of HT links.
- If it's not possible to keep comms local it is not hugely significant which non-local memory location data comes from.
- Not worth creating complex intra-node topologies, in either bandwidth or latency dominated regimes.

OpenMP test

- Perform 24 matrix multiplications in parallel
  1. Each thread initialises 2 matrices to use in a 3000x3000 matrix multiplication.
     - This places the data locally – sets affinity.
  2. Each thread calculates a separate 3000x3000 matrix multiplication. In each die...
     - 3 threads work on data initialised by themselves;
     - 1 thread works on data init in neighbouring die;
     - 1 thread works on data init in counterpart die;
     - 1 thread works on data init in diagonally opposite die.
- How do you know where threads run?
  - By default, threads are placed sequentially...

OpenMP test: ordered results

Thread ID	Data Init Thread ID	Time
1	1	22.517
8	8	22.519
12	12	22.521
13	13	22.523
7	7	22.523
14	14	22.525
20	20	22.525
6	6	22.527
19	19	22.529
0	0	22.536
2	2	22.538
18	18	22.561
3	9	22.824
21	15	22.832
22	10	22.836
15	21	22.837
9	3	22.843
4	16	22.845
16	4	22.845
10	22	22.855
11	17	22.879
17	11	22.890
5	23	22.896
23	5	22.900

- We get roughly the ordering expected from the hierarchy of HT links.
- It's always best to work on local data.
- Working on neighbouring data same as working on counterpart data.
- It's always worst to work on "diagonally opposite" data.

OpenMP Test: Levels of optimisation, XT4 Results

Compile	XT4	XT6 local data			XT6 non-local data			XT6 non-local – local worst case cost
		Min	Med	Max	Min	Med	Max	
PGI –O0	**120.35**	116.71	**116.89**	116.91	117.09	**117.12**	117.14	**0.43**
PGI –O	**21.46**	22.52	**22.53**	22.56	22.82	**22.84**	22.90	**0.38**
PGI -fast	**7.52**	7.94	**7.95**	7.97	8.25	**8.28**	8.39	**0.45**

- Comparing with XT4:
  - Un-optimised: memory bound. Higher memory bandwidth of XT6 wins, even when working on non-local data.
  - Local optimisations: working from cache more. Higher clock speed of XT4 wins.
  - Aggressive optimisations: vectorized code. XT4 wins again.
- Non-local cost is constant (final column)
  - The same amount of data is being retrieved
  - For optimised code this becomes a more significant ratio.

Questions

1. What is the performance penalty for working data stored in non-local memory?
2. What is the effect of having a single link (which is the same as in the XT4) to the interconnect?

Micro-benchmarking tests...

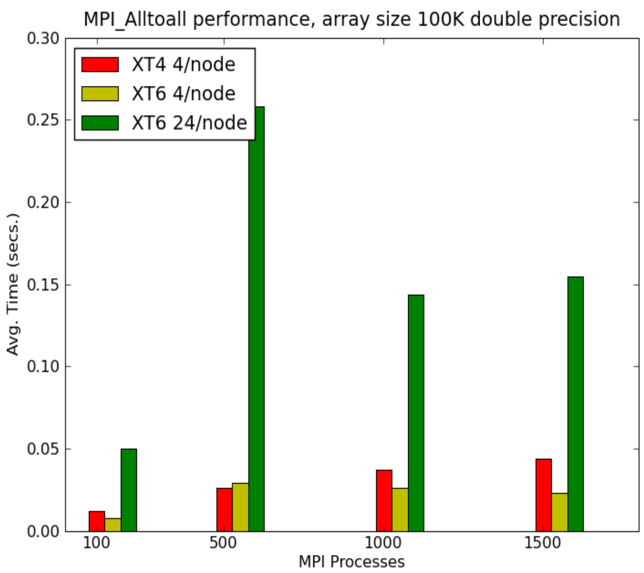
HECToR

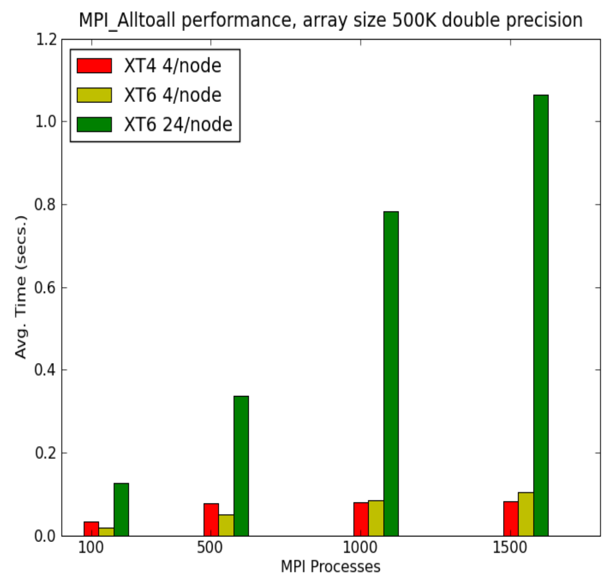
RESEARCH COUNCILS UK

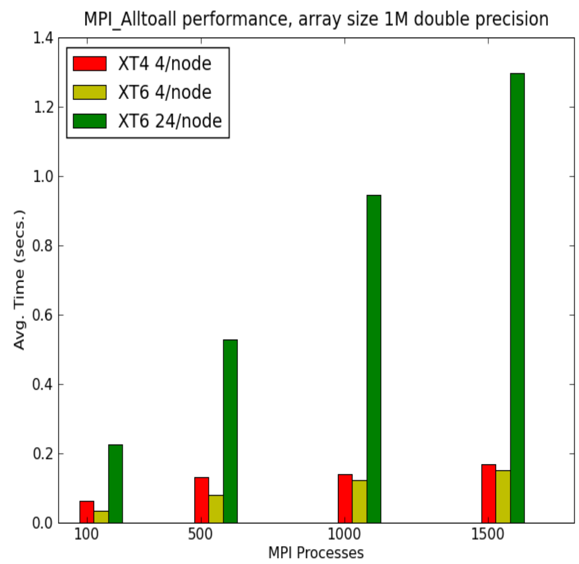
Inter-node communications

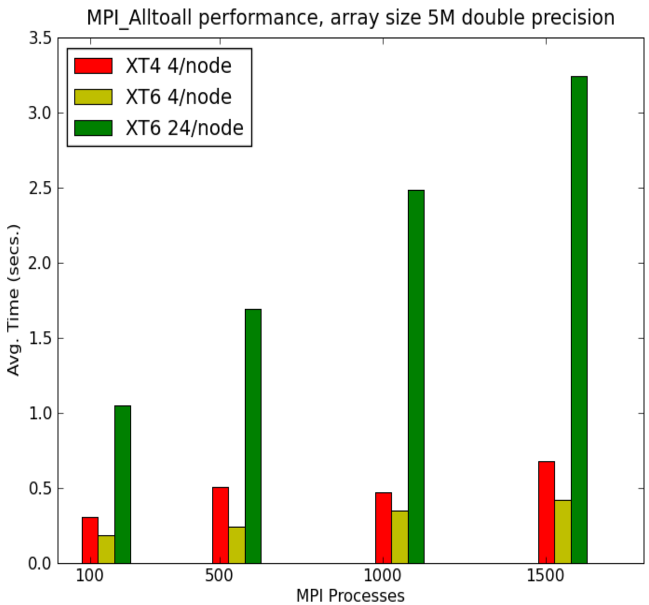
- The easiest way to stress this is with MPI_Alltoall.
- How does the performance compare with the XT4 for various job sizes, message lengths?

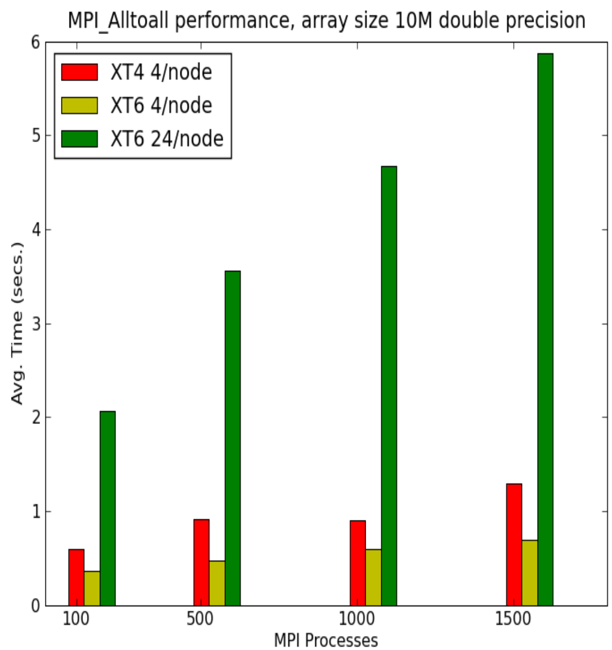
MPI_Alltoall performance: XT4, XT6 (4/node), XT6 (24/node)

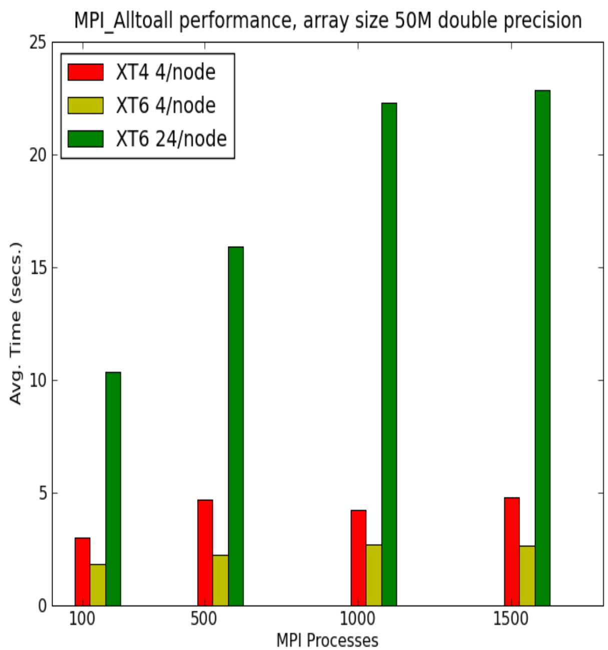












Micro-benchmarking summary

- On-node memory:
  - Accessing die-local memory faster than non-local
    - OpenMP: More significant for highly optimised code.
  - Differences between non-local times small.
  - Either restrict SMP to a die or a whole node – no benefit to creating more complex hierarchies.
- Off-node collective communication disastrous due to same link to the interconnect as XT4
  - Gemini should hopefully fix this!
- Let's look at some real codes...

HECToR

RESEARCH COUNCILS UK

CASINO

Lucian Anton

Mixed Mode

- Reduce network traffic

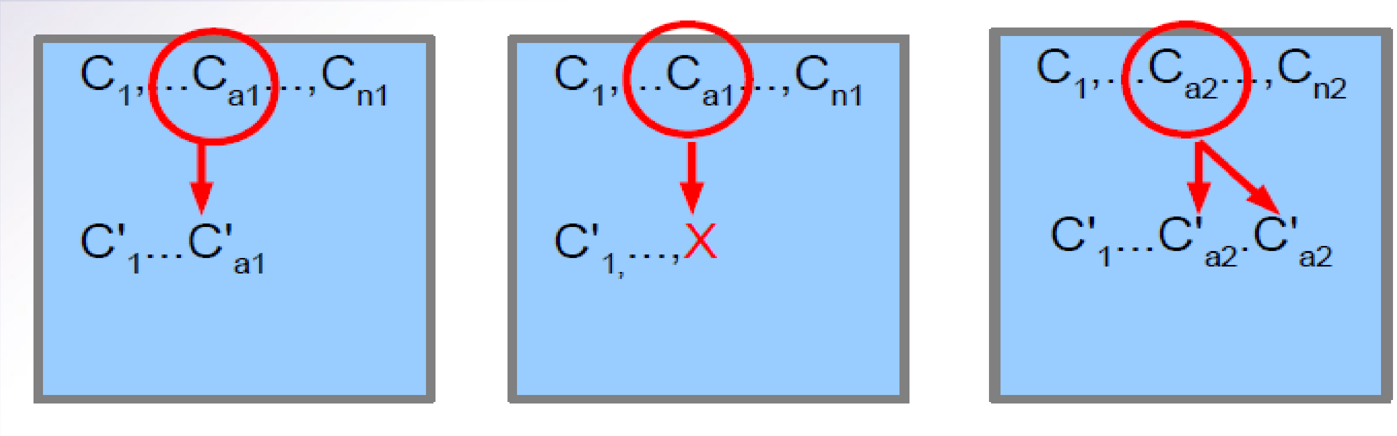
$$S(v_l(N), t_{comm}(N)) < S(v_l(N/t + x(N - N/t)), t_{comm}(N/t))$$

- Load balancing, parallelism based on tasks
- Additions of new levels of parallelism.

HECToR

RESEARCH COUNCILS UK

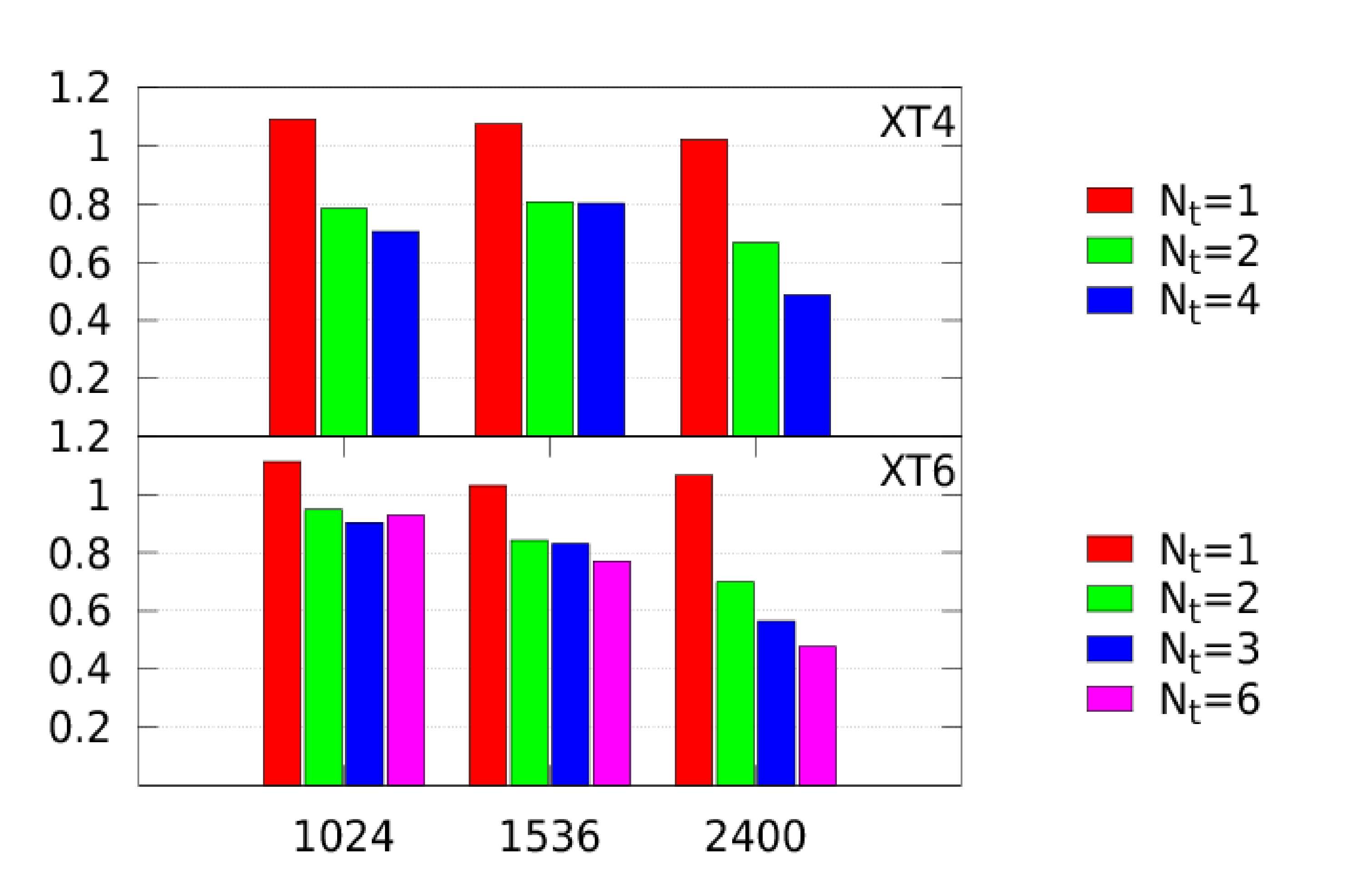
CASINO mixed mode



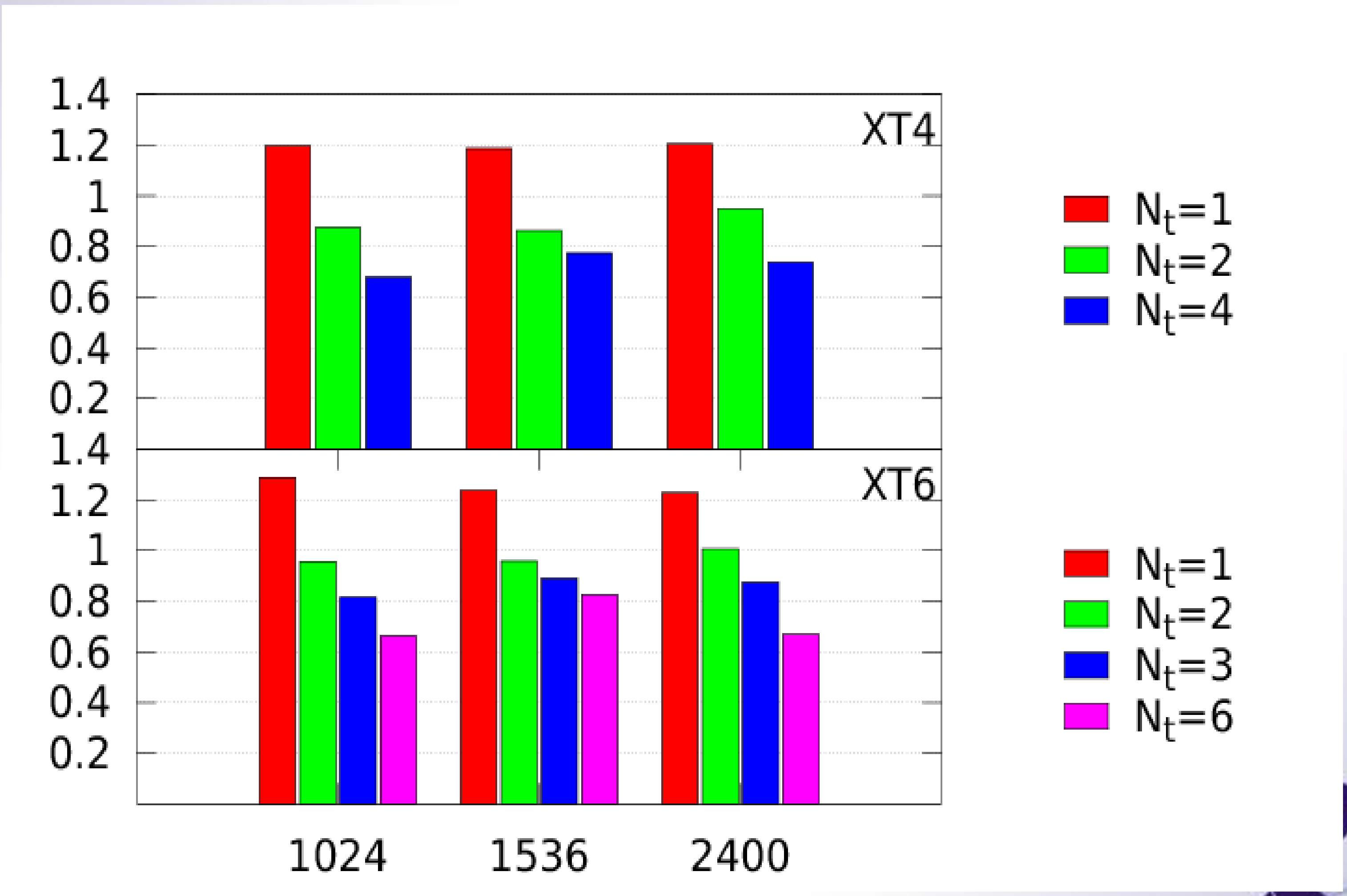
$$t = N_{step} \times \frac{N_{pop}}{P} \times \frac{t_{config}}{N_{thread}}$$

$$t_{config} \sim N_e^2$$

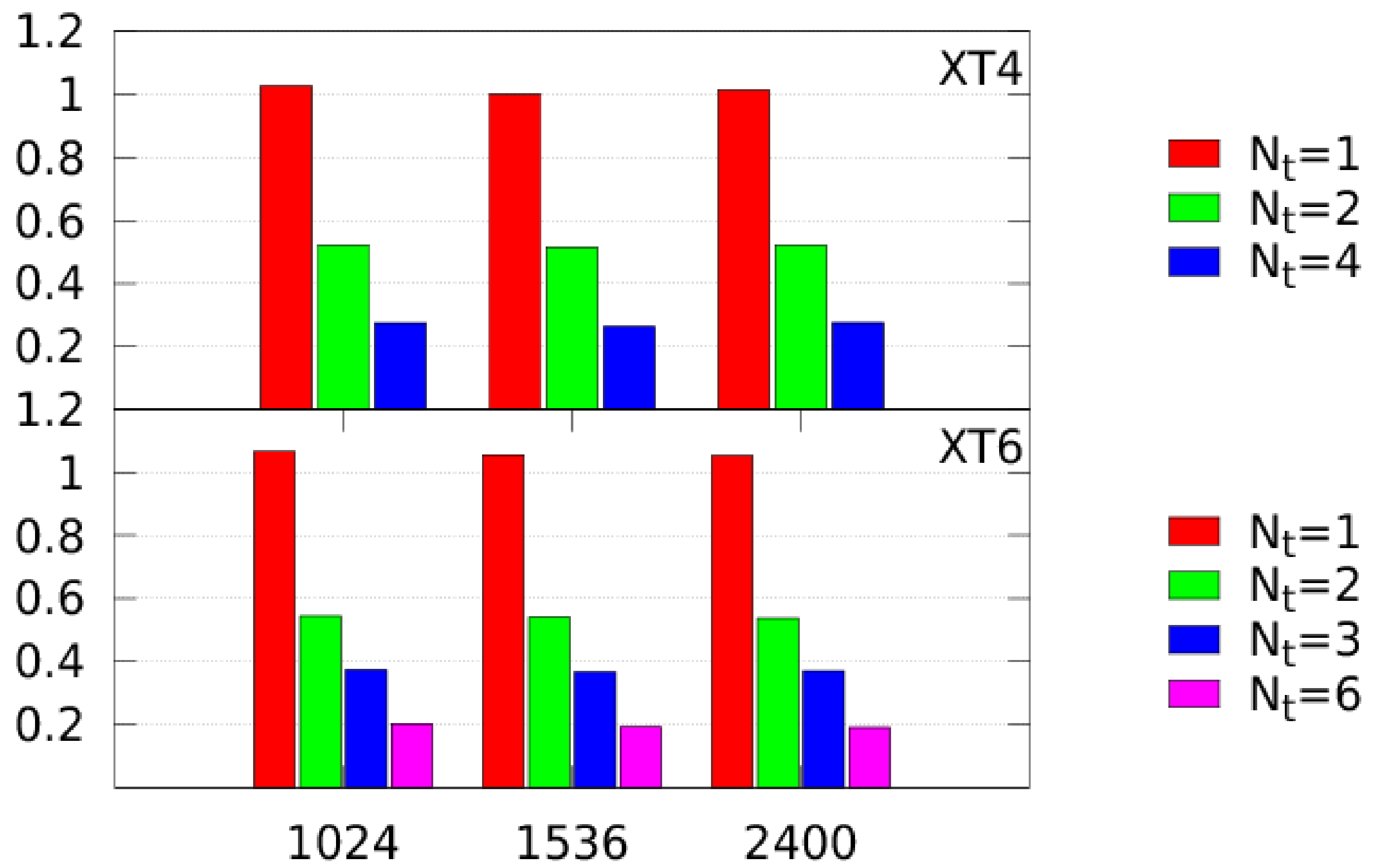
OPO kernel OpenMP scaling for three system sizes



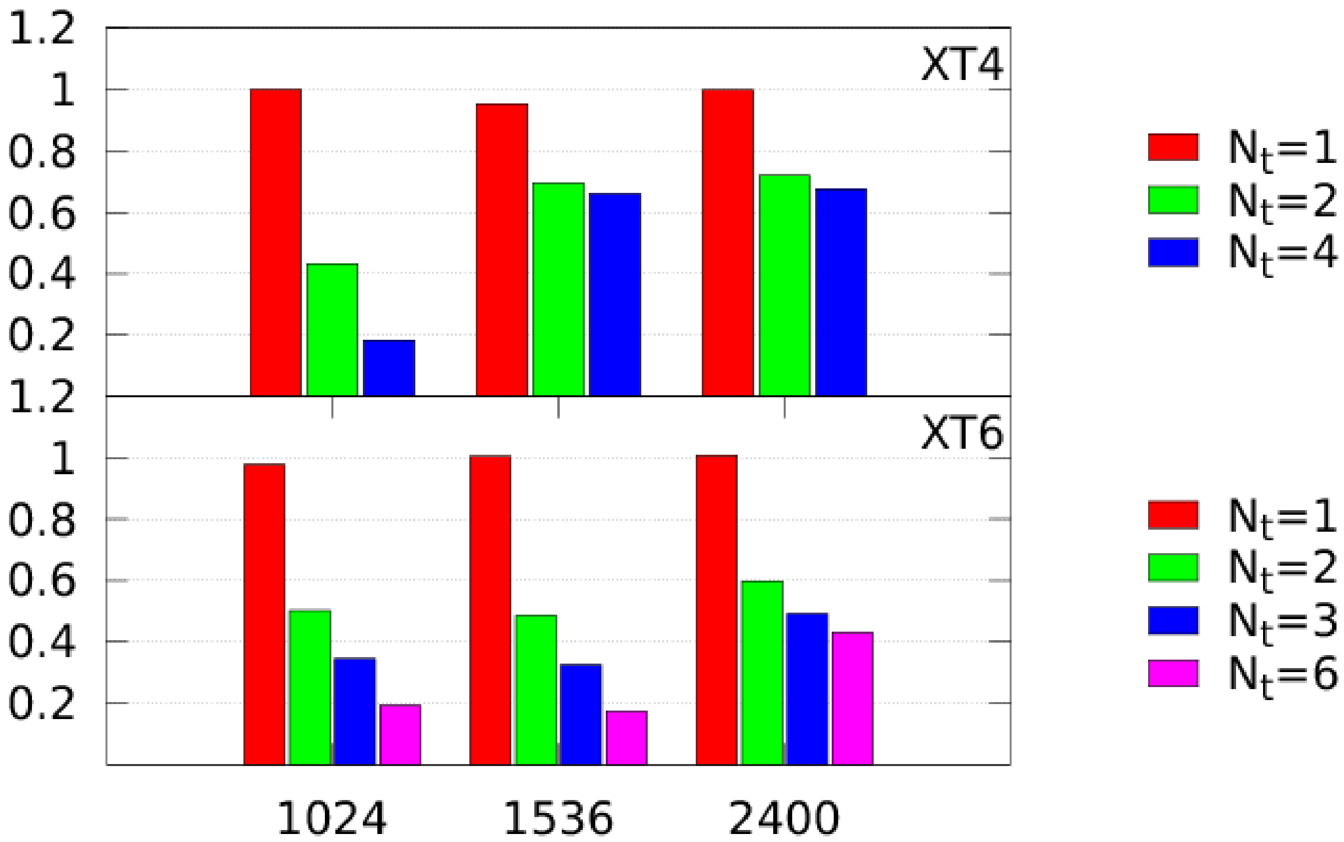
Jastrow kernel OpenMP scaling for three system sizes



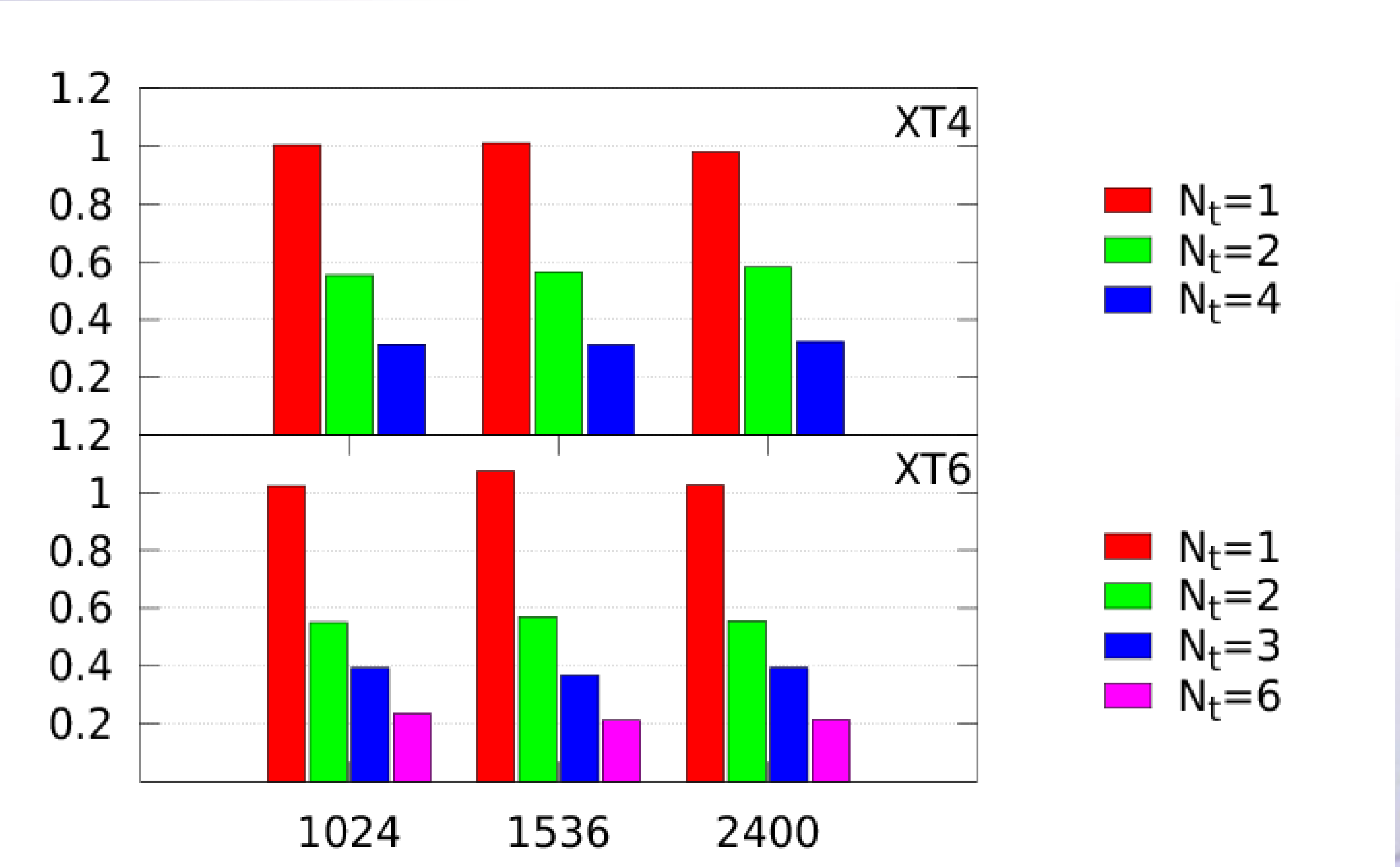
EWALD kernel OpenMP scaling for three system sizes



Update D kernel OpenMP scaling for three system sizes



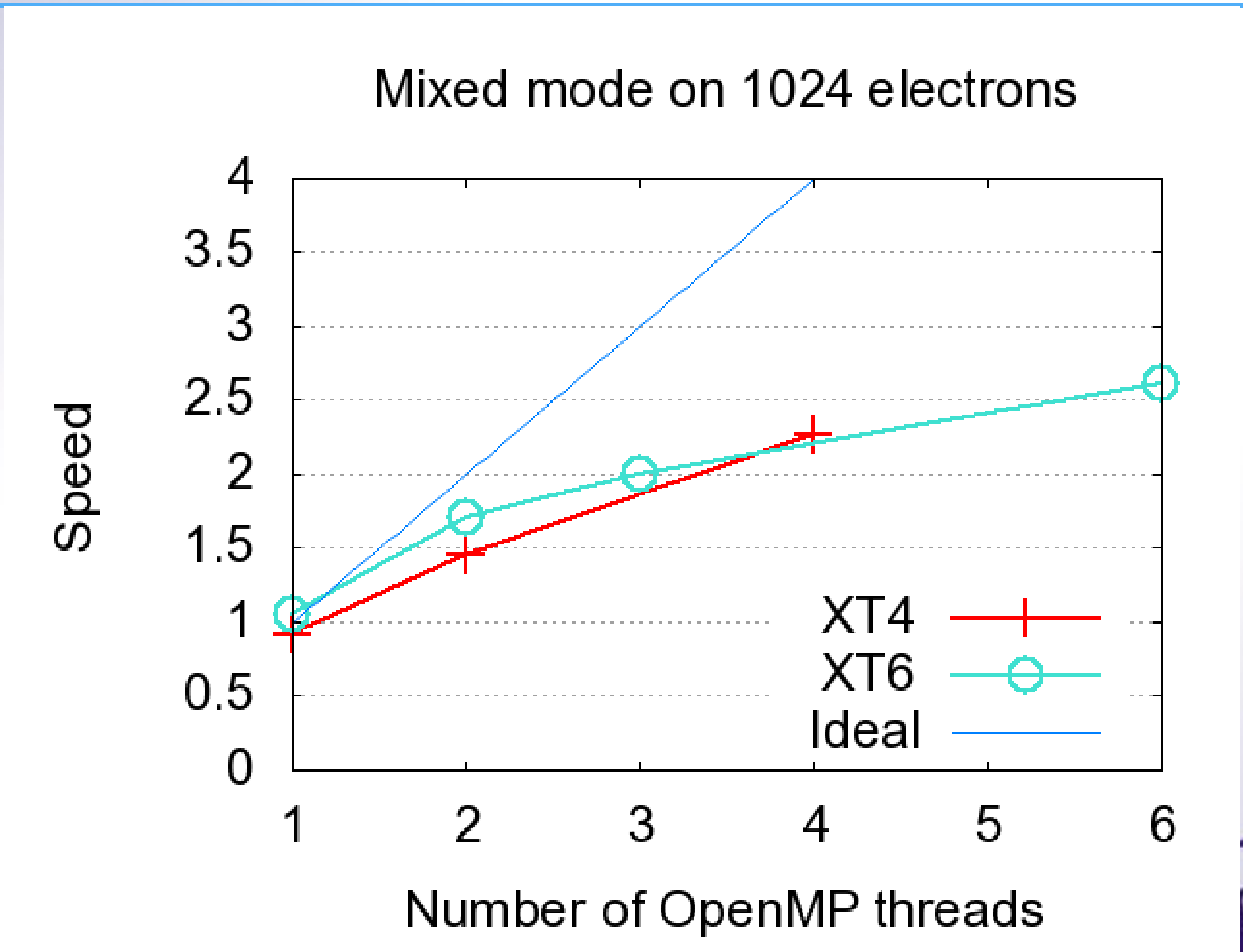
R_{ee} kernel OpenMP scaling for three system sizes



HECToR

RESEARCH COUNCILS UK

CASINO mixed mode performance



GloMAP Mode MPI

Mark Richardson

Background

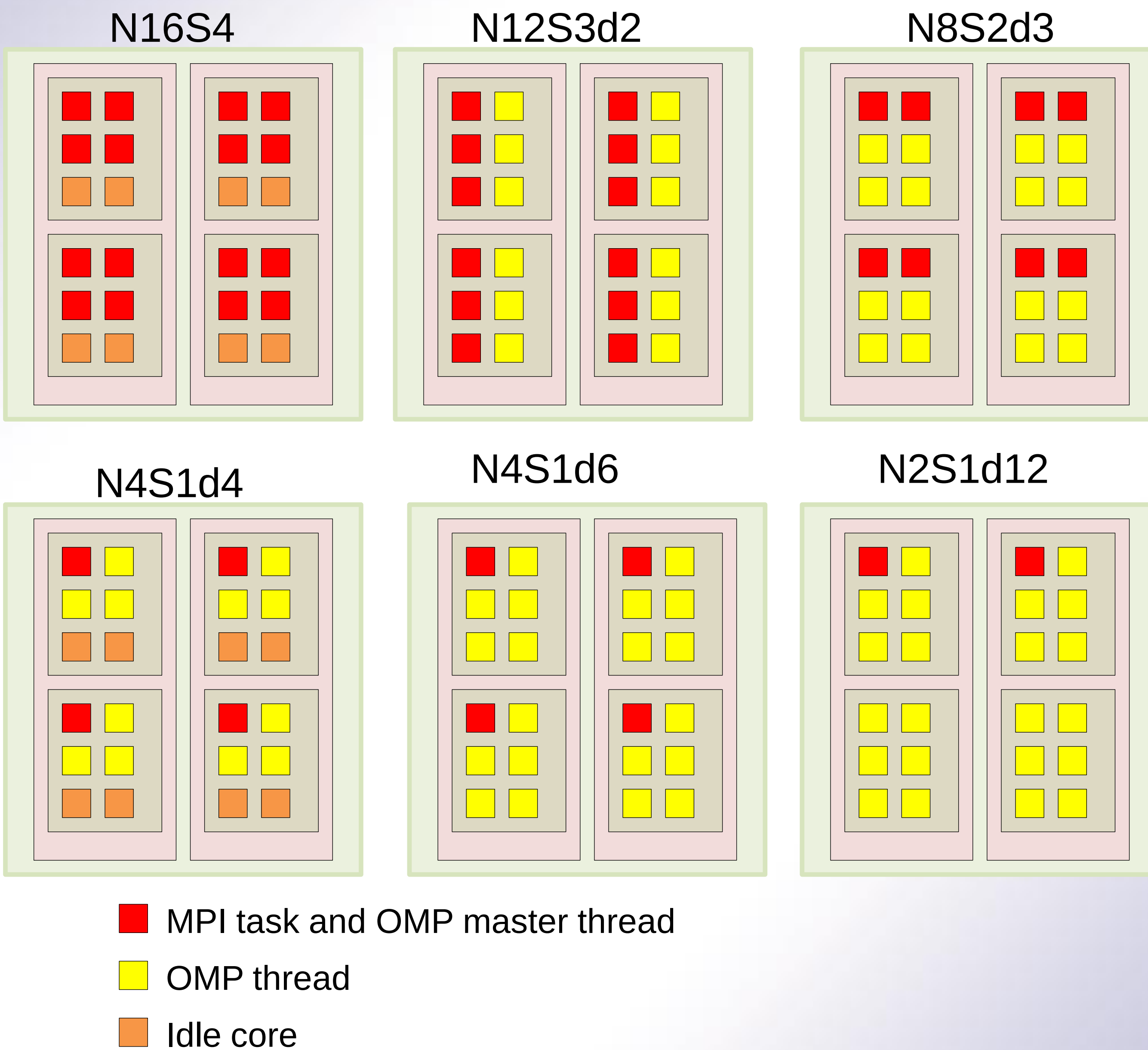
What is GLOMAP

- Aerosol process simulation
- MPI version regular use on HECToR
- Open MP version used on fat nodes (32)
- Hybrid version was subject of DCSE
- Focus of the talk is placement of MPI tasks

XT6: Naive placement is packed (N24)

- Will look at 32, 64 and 128 MPI task configurations
  - In pure MPI mode can look at “balanced” placements
- With hybrid version can look at spreading out
  - n32N4S1d4 n64N12S3d2 (i.e. 128 cores)
- Note the final node is not balanced

Diagrams of nodal arrangements



HECToR

RESEARCH COUNCILS UK

PBS script to run hybrid on phase2b

```
#!/bin/bash --login
#PBS -N n32N4S1d6
# this is 8 nodes x 24 cores
#PBS -l mppwidth=192
# we say we will use 24 per node to get 8 nodes
#PBS -l mppnppn=24
#PBS -l walltime=0:20:00
#PBS -A z03
#PBS -m abe

cd $PBS_O_WORKDIR
export GMM=${HOME}/Projects/GMH_build
export MPICH_PTL_UNEX_EVENTS=80000
export MPICH_MAX_SHORT_MSG_SIZE=8000
export MPICH_UNEX_BUFFER_SIZE=400M
date > StartedJob.${PBS_JOBNAME}
export OMP_STACKSIZE=1000M
export OMP_NUM_THREADS=6
export PSC_OMP_AFFINITY=FALSE

# this is what we want to do on the nodes
aprun  -n 32 -N 4 -S 1 -d ${OMP_NUM_THREADS} ${GMM}/src_32/xtgmm.exe

# end of script
```

Utilisation considerations

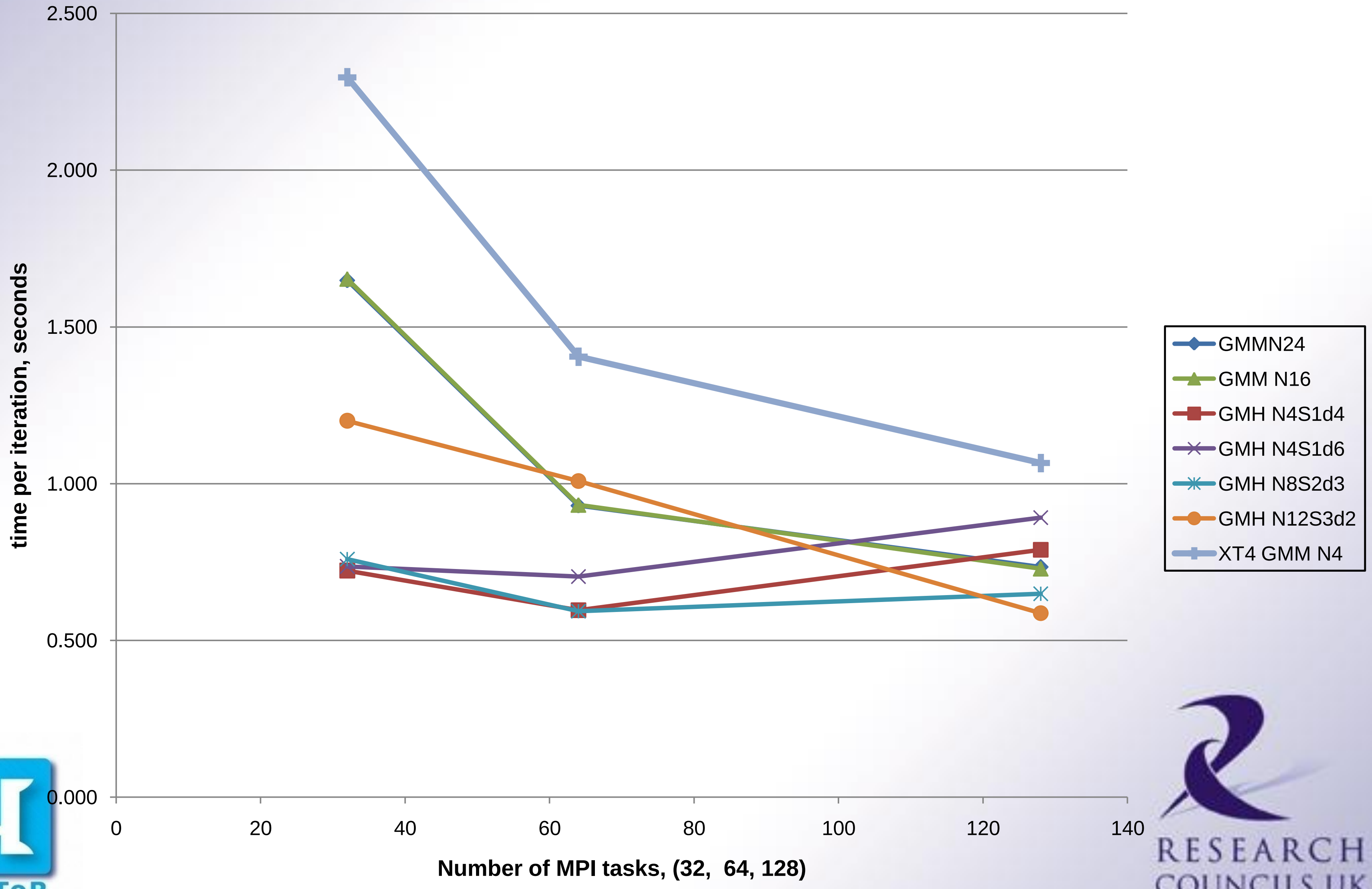
OMP costs skewed by modulus 24

Table shows number of nodes in use and utilisation

This an average due to end node under-utilisation

MPI	32		64		128	
N24	2, 0.67	32/48	3, 0.89	64/72	6, 0.89	128/144
N4d6	8, 1.00	192	16, 1.00	384	32, 1.00	768
N4d4	8, 0.67	128	16, 0.67	256	32, 0.67	512
N8d3	4, 1.00	96	8, 1.00	192	16, 1.00	384
N12d2	3, 0.89	64/72	6, 0.89	128/144	11, 0.97	256/264

Scaling including Open MP



Effect of Open MP

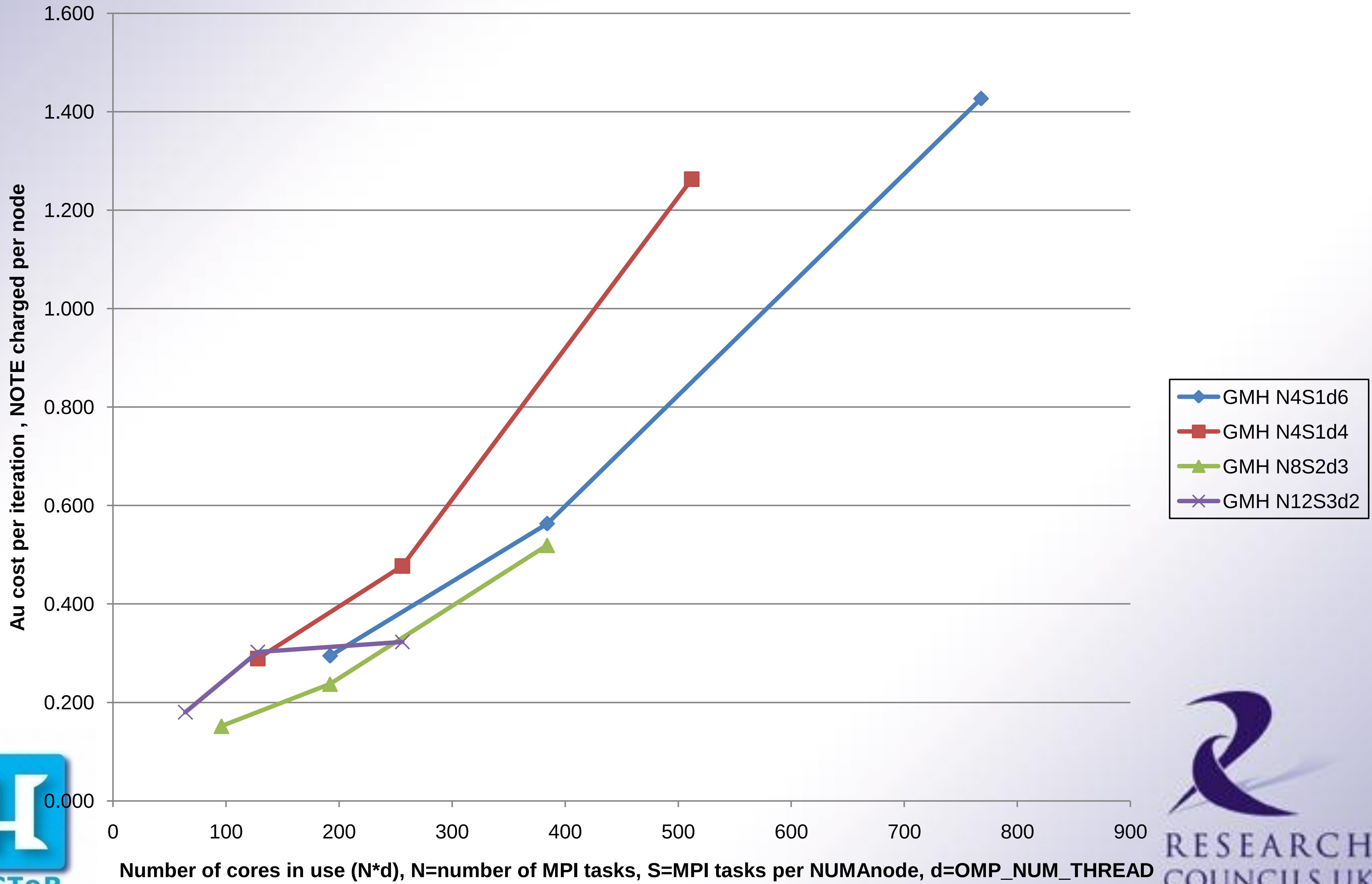


Table 1: phase2a Cray PAT results for fully populated nodes GLOMAP mode purely MPI version and three configurations

Phase 2a is one quad core per node, will use the fully packed pure MPI version for comparisons (i.e. Production version)

	M32 % of whole sim	M32 % of GMM only	M64 % of whole sim	M64 % of GMM only	M128 % of whole sim	M128 % of GMM only
ADVX2	4.2	5.7	2.8	5.5	1.3	5.7
ADVY2	11	14.9	7.1	13.9	2.2	9.7
ADVZ2	4.9	6.6	3.2	6.3	1.4	6.2
CONSOM	5.4	7.3	3.6	7.1	1.1	4.8
CHIMIE	40.9	55.3	27.4	53.7	12.4	54.6
MAIN	7.7	10.4	7	13.7	4.4	19.4
TOTAL FOR GMM	74	100	51	100	22.7	100
MPI	13.3	-	28.3	-	47.4	-
MPI_SYNC	12.7	-	20.7	-	29.9	-

Summary and Conclusions

Glomap mode has been converted to hybrid and tested on XT6

The performance of the pure MPI production code on phase2b has improved 30% over phase2a timings

Activating the OpenMP directives allows the code to use more cores per MPI task

- Reduced number of MPI tasks keeps the code in the “good” scaling region (64 MPI tasks)

Filling nodes in a “balanced” manner has little effect on performance

- Except when considering the AU usage

Open MP is limited

- can only be as effective as the loop count

CABARET and Incompact 3D

Phil Ridley

What is CABARET?

- High resolution scheme for CFD problems suited to
  - Shock capturing
  - Linear wave propagation
- Compact Accurately Boundary Adjusting High REsolution Technique (CABARET)
- General purpose Implicit Large Eddy Simulation (ILES)
  - Removes all scales smaller than the grid scale from the solution
  - No effect on the large eddies that are directly simulated

What is different about CABARET

- Extra evolutionary variable
- Preserves small phase and amplitude error
- Non linear flux correction
- Removes under-resolved fine scales from solution
- Balance between numerical dissipation and dispersion error

What is Incompact3D?

- Direct Numerical Simulation of turbulent CFD applications
  - Suitable for flows passing through fractal geometries
  - Turbulence resolved over entire spatial grid scale
  - Billions of grid points are required to resolve fine scales
  - Implicit finite difference scheme
  - Spectral method used to solve the pressure equation
- FFTs require use of multi-dimensional data decomposition

Improving Scalability of Incompact3D

- Ongoing HECToR dCSE project
  - Turbulence, Mixing and Flow Control group at Imperial
  - Opportunities identified to develop reusable software components for a wider range of applications
- Parallel library development
  - A general-purpose 2D decomposition library
  - For applications based on 3D Cartesian data structures
  - Result - a distributed 3-dimensional FFT library
  - Very useful for distributed spectral-based Poisson solvers

HECToR

RESEARCH COUNCILS UK

Summary of CABARET and Incompact3D codes

- Fortran 90 / MPI
- CABARET unstructured grid
- Incompact3D structured grid
- Structured multi-dimensional data decompositions
- Preprocessing for grid decomposition
- Postprocessing for output
- At least 10^6 grid points for CABARET
- At least 10^9 grid points for Incompact3D

HECToR

RESEARCH COUNCILS UK

Main loops

CABARET

```
DO I=1,NSIDE
NCF=GEMSIDECELL(I,1)
IF(NCF/=0)THEN
CALL TAKESTENCIL1F(I)
IF (ABS(CHAR3B)<DEPS) CHAR3B=0
IF (ABS(CHAR3F)<DEPS) CHAR3F=0
IF(CHAR3B+CHAR3F.LE.0)THEN... ENDIF
ENDIF
IF(NCF<0) THEN ... ENDIF
END DO
```

Not surprisingly these won't vectorise with any compiler!

BUT main loops in Incompact3D arise from a regular cartesian grid structure and vectorise extremely well!

CABARET XT4/XT6 Performance

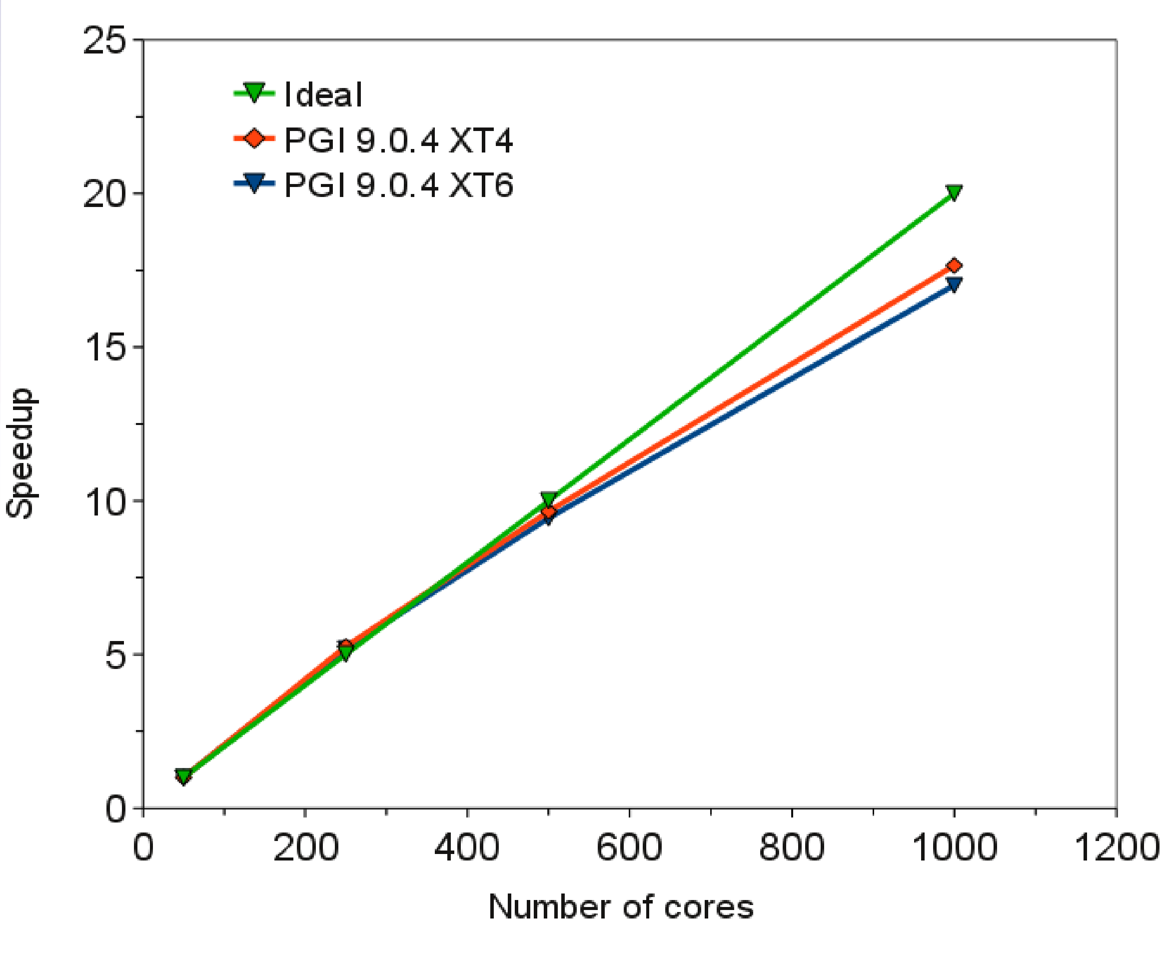


Figure 1 : Performance for 276 CABARET timesteps using 6.4M grid points (no I/O and fully populated nodes)

CABARET Simulation

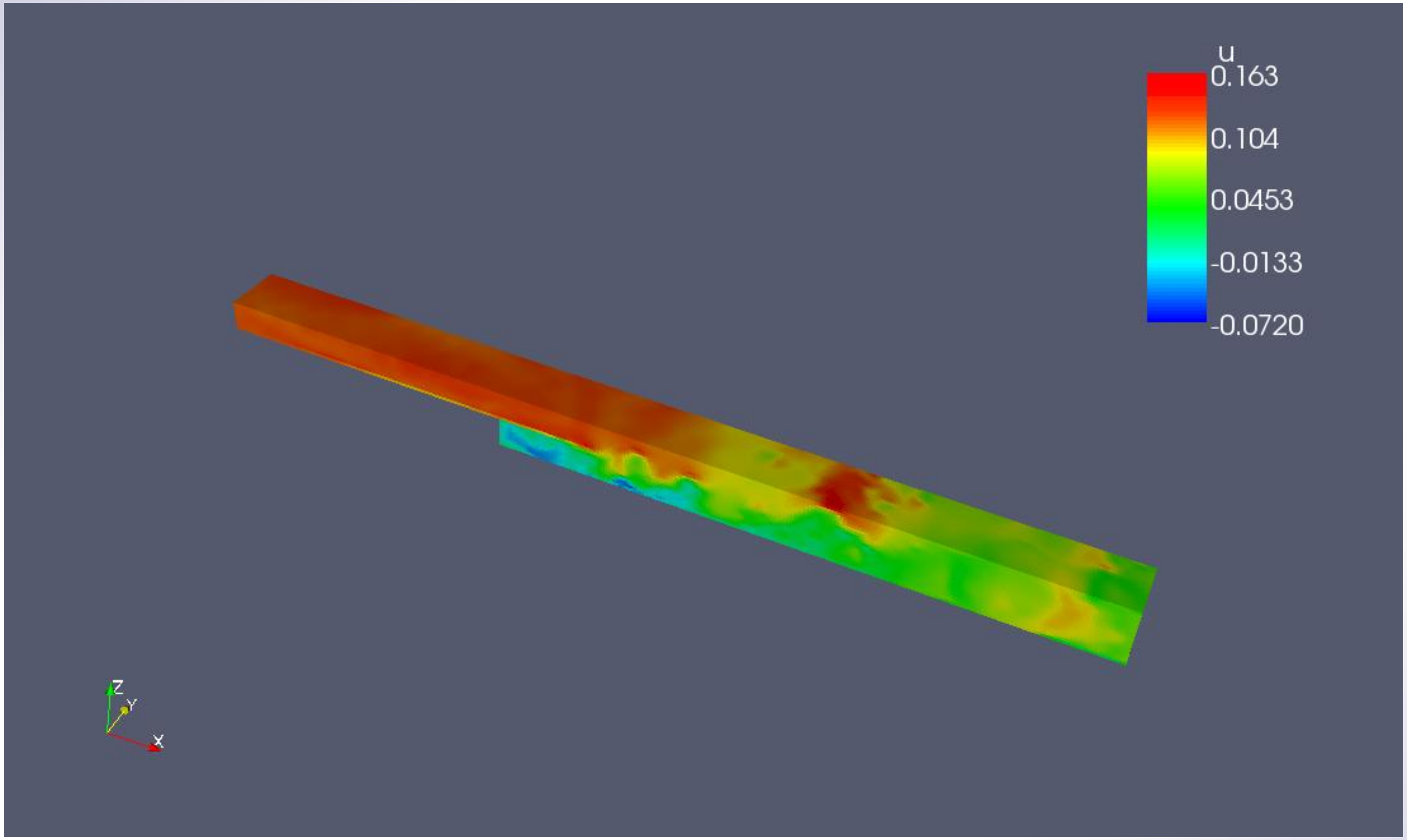


Figure 2 : Backward facing step, Re=5000, M=0.1, laminar flow bcs,10000 iterations

Incompact3D Strong Scaling on XT4

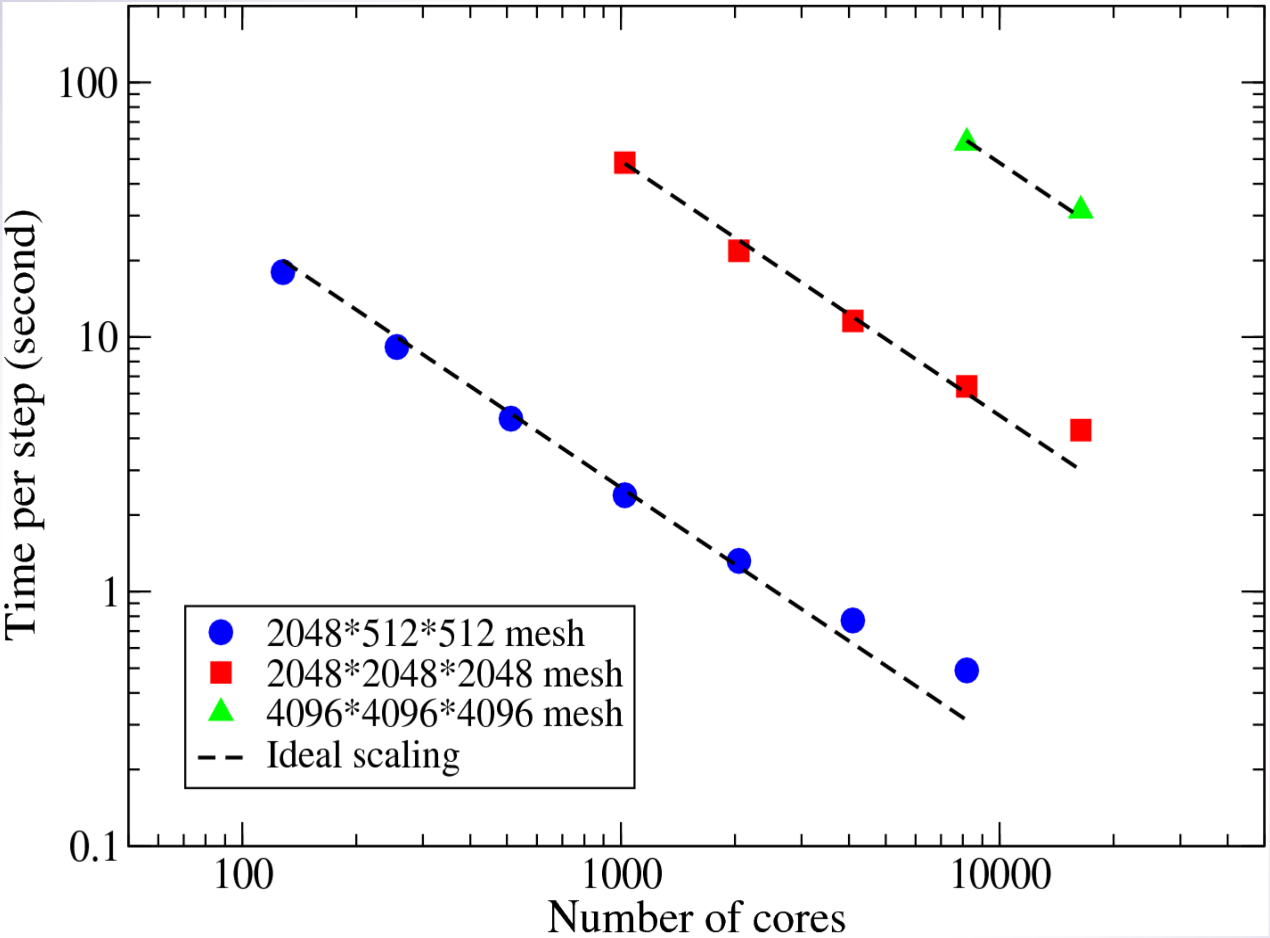


Figure 3 : Comparison of scaling for 3 Incompact3D test cases on the XT4

Incompact3D XT4/XT6 Results

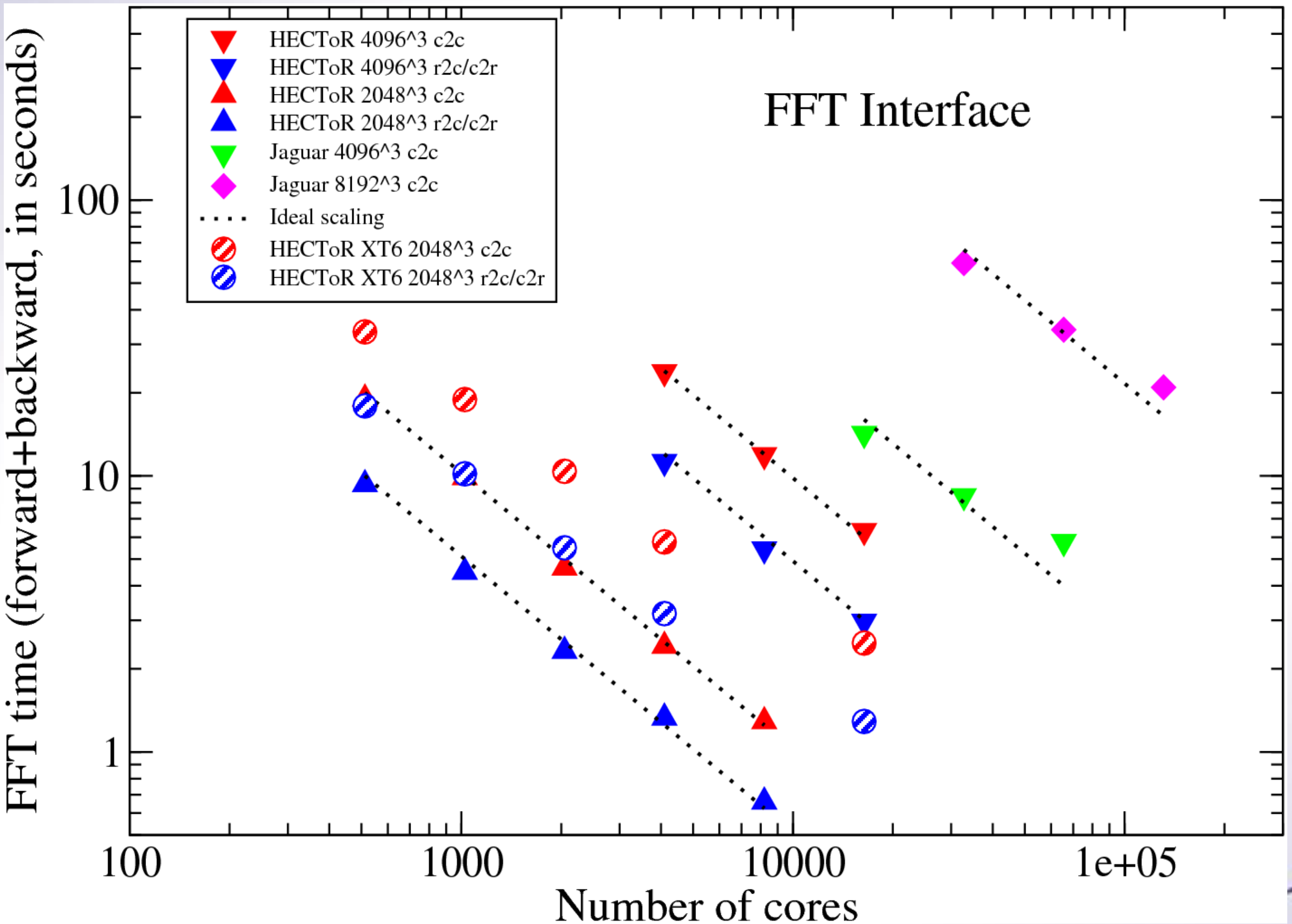


Figure 4 : Comparison of scaling for Incompact3D test cases on the XT4/XT6 and Jaguar

Summary

- CABARET
  - General purpose Implicit Large Eddy Simulation (ILES)
  - Non vectorisable loops mean that computation and communication has to be ordered optimally
- Incompact3D
  - Direct Numerical Simulation of fractal generated geometries
  - Highly scalable, user-friendly 2D decomposition library and distributed FFT library

HECToR

RESEARCH COUNCILS UK

Useful Information

- HECToR distributed CSE
  - The applications codes discussed today have benefited from software development in order to help improve their performance under this scheme

    http://www.hector.ac.uk/cse/distributedcse/

- The data decomposition Library developed within the Incompact3D project is an excellent framework for scalability for similar application based algorithms
  - Source code available for all HECToR users

ning.li@nag.co.uk or phil.ridley@nag.co.uk

DL_POLY

Valene Pellissier

Agenda

- DL_POLY
- Running Jobs on XT6
  - Intro
  - Time
  - Cost
- Summary

DL_POLY

- DL_POLY :
  - Molecular dynamics simulations of macromolecules, polymers, ionic systems and solutions on a distributed memory parallel computer
  - Developed at Daresbury Laboratory by W.Smith and I.T. Todorov
  - DL_POLY_3 : based on Domain Decomposition, suits to large computer systems, 2^N procs, 10^3 to 10^9 atoms
  - DD : division of simulated systems into equi-geometrical spatial blocks or domains

- Test case 8 :
  - 16 gramicidin A molecules in aqueous solution
  - 792,960 atoms

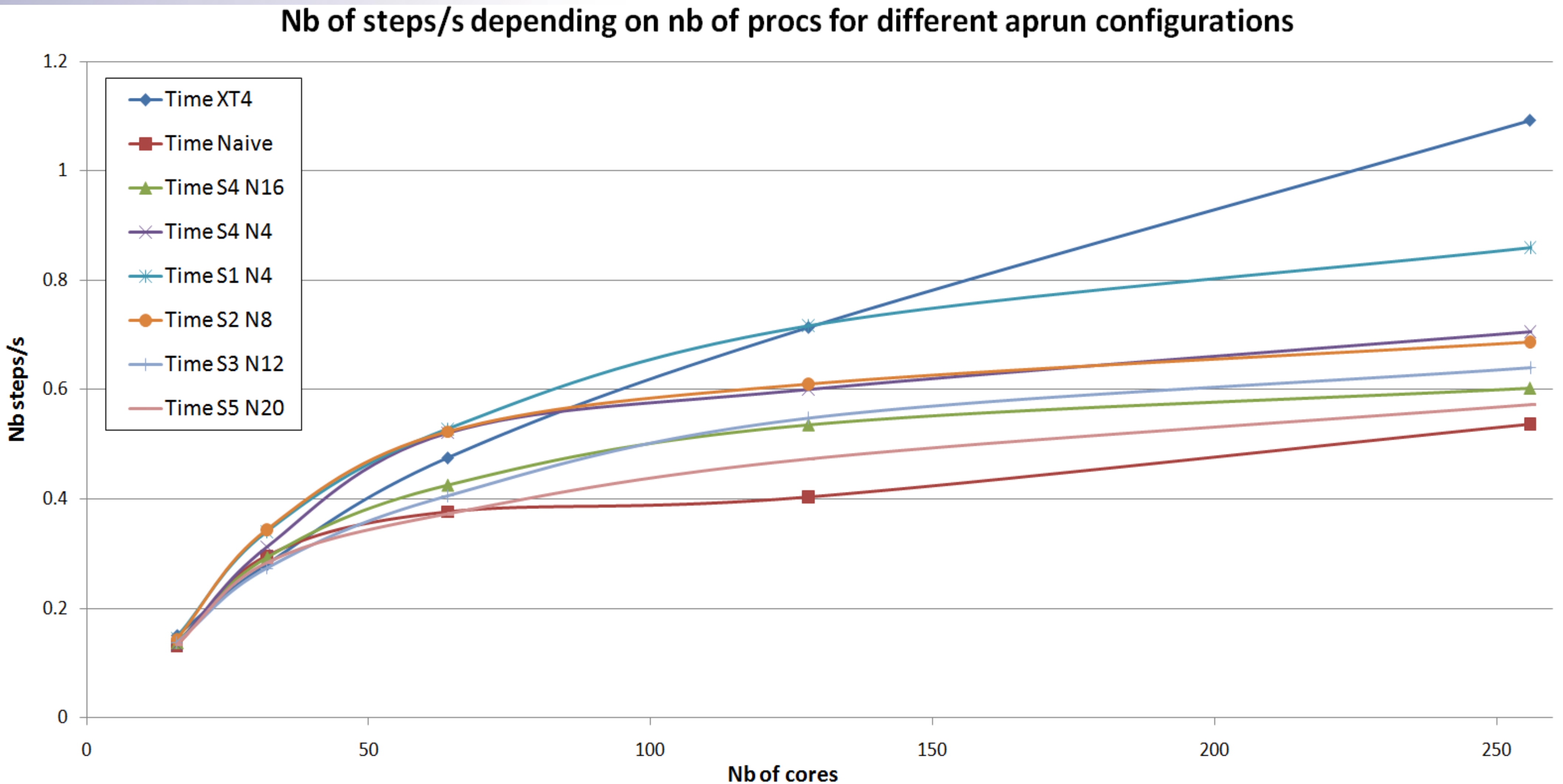
Running Jobs on XT6 - Intro

- Aprun parameters :
  - -n : total number of processes
  - -N : number of processes per node
  - -S : number of processes per hex-core die

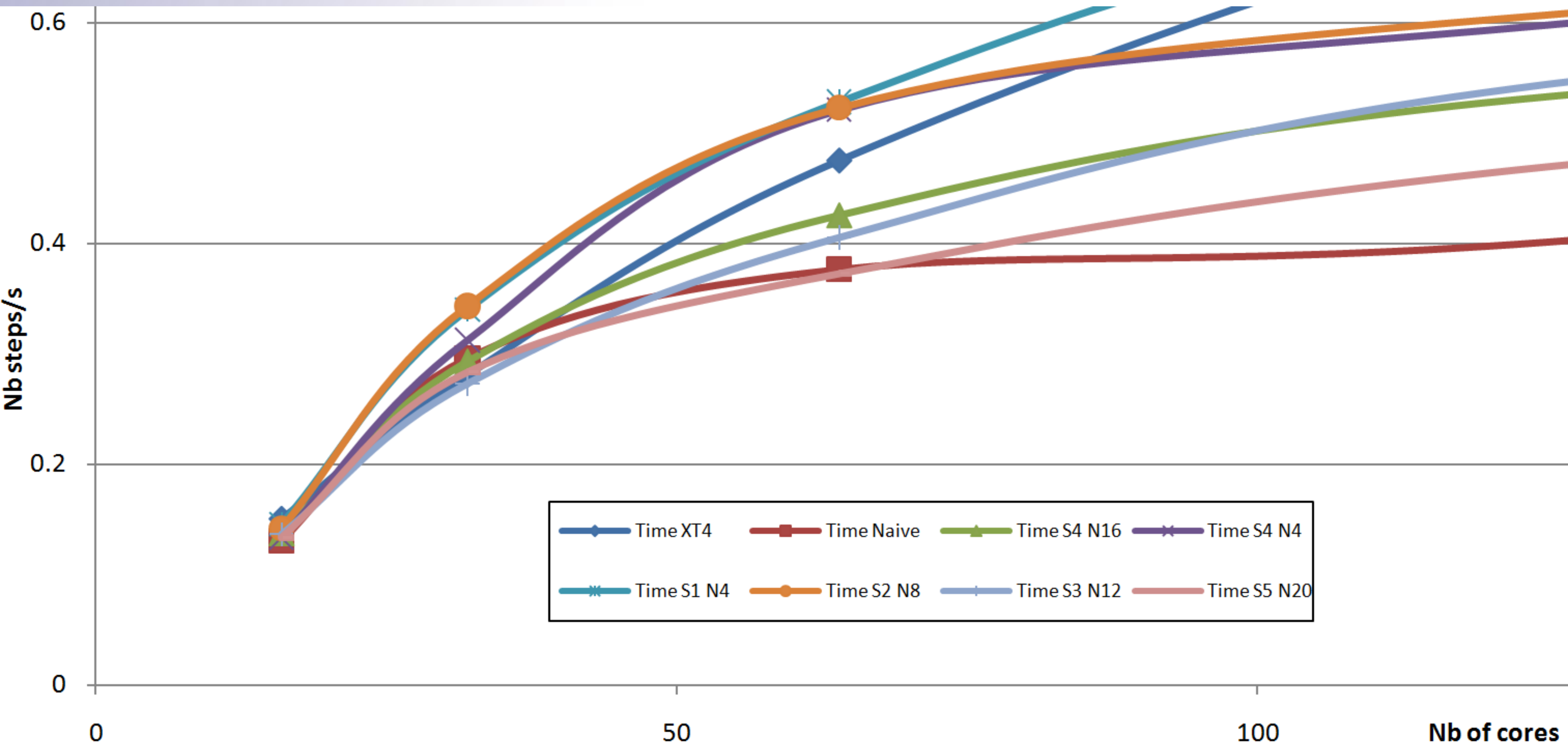
- Number of nodes involved for the different aprun configurations :

Procs	Naive	S1 N4	S2 N8	S3 N12	S4 N4	S4 N16	S5 N20
16	1	4	2	2	4	1	1
32	2	8	4	3	8	2	2
64	3	16	8	6	16	4	4
128	8	32	16	11	32	8	7
256	11	64	32	22	64	16	13
512	22	128	64	43	128	32	26

Running Jobs on XT6 – Time (1/3)



Running Jobs on XT6 – Time (2/3)



HECToR

RESEARCH COUNCILS UK

Running Jobs on XT6 – Time (3/3)

Procs	XT4	Naive	S4N16	S4N4	S1N4	S2N8	S3N12	S5N20
16	199.15	228.67	217.72	223.91	205.59	210.89	219.27	223.86
32	106.79	101.34	102.25	95.99	88.22	87.33	109.80	105.73
64	63.13	79.69	70.47	57.59	56.80	57.37	74.01	80.48
128	42.04	74.32	55.98	49.93	41.84	49.19	54.77	63.45
256	27.42	55.86	49.75	42.47	34.88	43.68	46.87	52.45
512	22.14	57.62	47.99	41.39	31.29	38.31	47.31	51.00
Diff XT4 (%)	X	62.73	41.33	25.65	7.27	21.27	40.79	51.86
Diff Naive (%)	-30.69	X	-11.29	-20.01	-29.79	-23.12	-10.53	-4.83

Time depending on number of processes for different aprun configurations

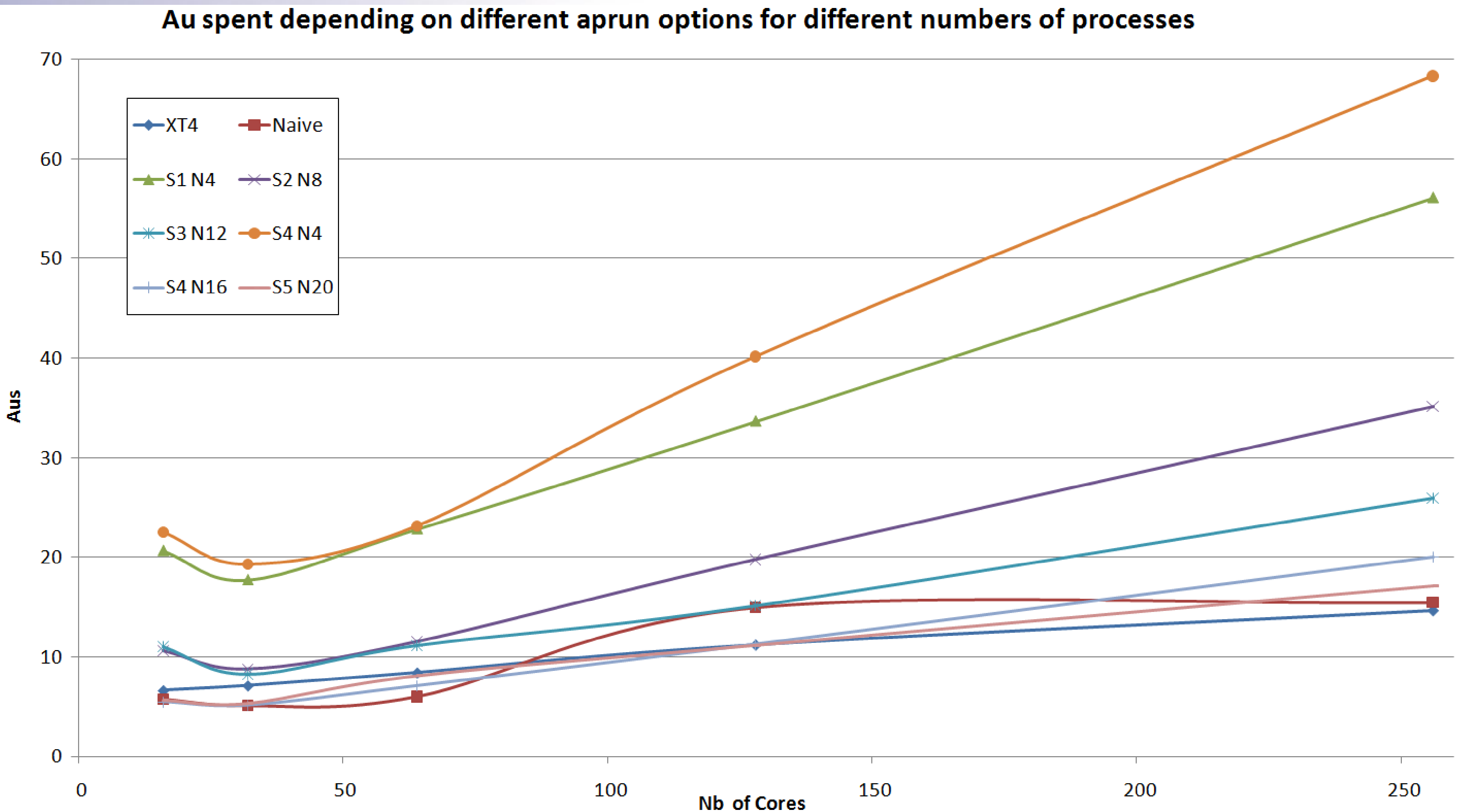
Best

Best on XT6

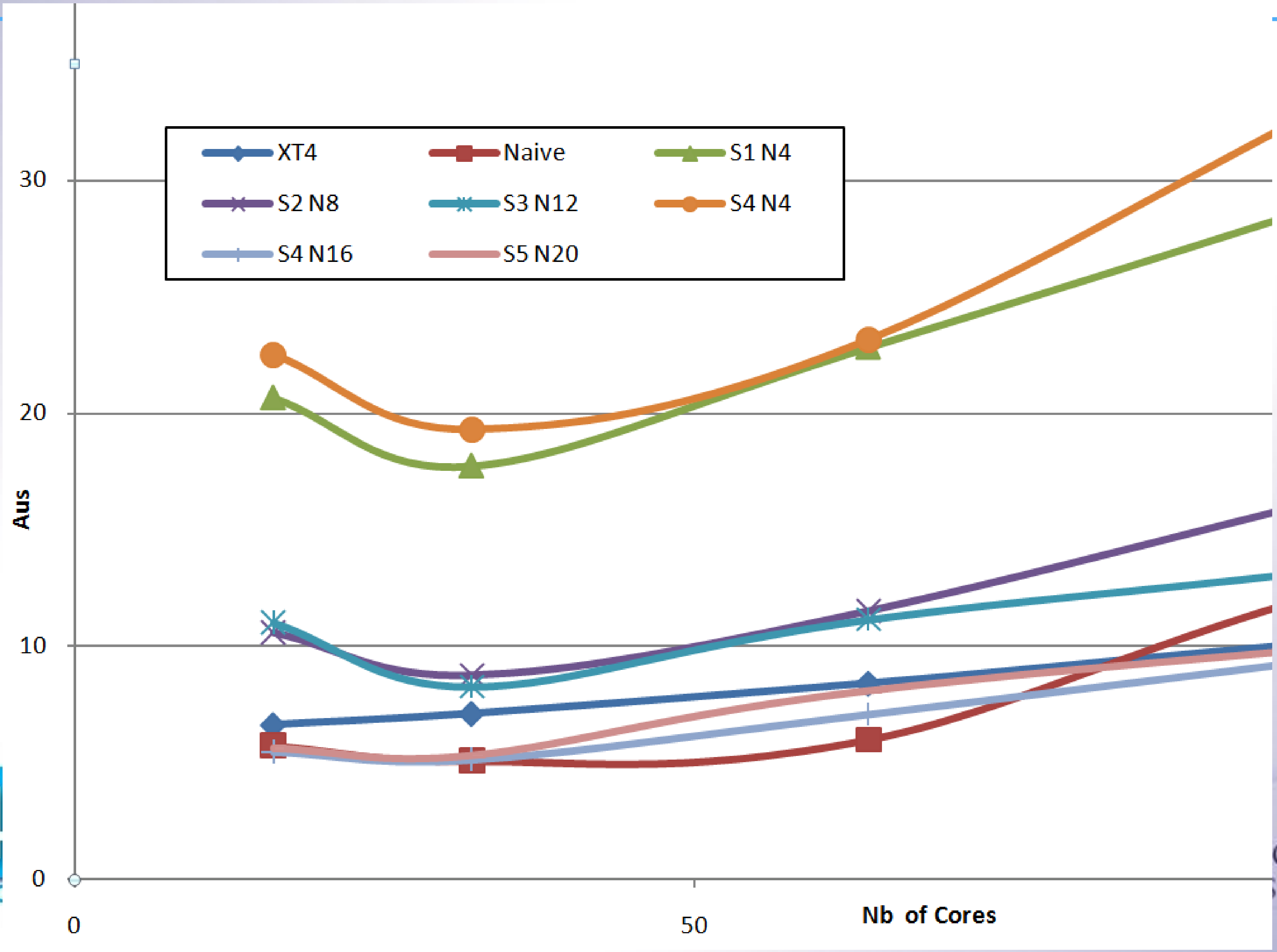
HECToR

RESEARCH COUNCILS UK

Running Jobs on XT6 – Cost (1/3)



Running Jobs on XT6 – Cost (2/3)



Running Jobs on XT6 – Cost (3/3)

Procs	XT4	Naive	S4N16	S4N4	S1N4	S2N8	S3N12	S5N20
16	6.64	5.75	5.47	22.51	20.67	10.60	11.02	5.63
32	7.12	5.09	5.14	19.30	17.74	8.78	8.28	5.31
64	8.42	6.01	7.08	23.16	22.84	11.16	11.16	8.09
128	11.21	14.94	11.26	40.16	33.65	15.14	15.14	11.16
256	14.65	15.44	20.01	68.32	56.10	25.92	25.92	17.14
512	23.61	31.86	38.60	133.17	100.65	51.13	51.13	33.33
Diff XT4 (%)	X	0.52	6.54	278.98	223.54	82.87	57.24	2.20
Diff Naive (%)	6.56	X	6.68	280.86	232.07	83.68	61.62	4.53

AUs depending on number of processes for different aprun configurations

Best

Best on XT6

HECToR

RESEARCH COUNCILS UK

Summary (1/2) - Results

- On average :
  - S1 N4 is the fastest and the most expensive
    - 30% faster than Naive (up to 45%)
    - 3 times more Aus than Naive
  - S4 N16 /S5 N20 : best compromise

Rel.Diff. Comp to Naive (%)	S4N16	S5N20
Time	-11.29	-4.83
AUs	6.68	4.53

    - Comp. to XT4 for 16,32, 64 procs :
      - -20% Aus for 5% more time
- Selectively :
  - Quickest : S1N4 for 512 procs :
    - 45% faster than Naive but 3 times more AUs
  - Cheapest : S4N16 for 128 procs :
    - 25% less Aus and 25% faster than Naive

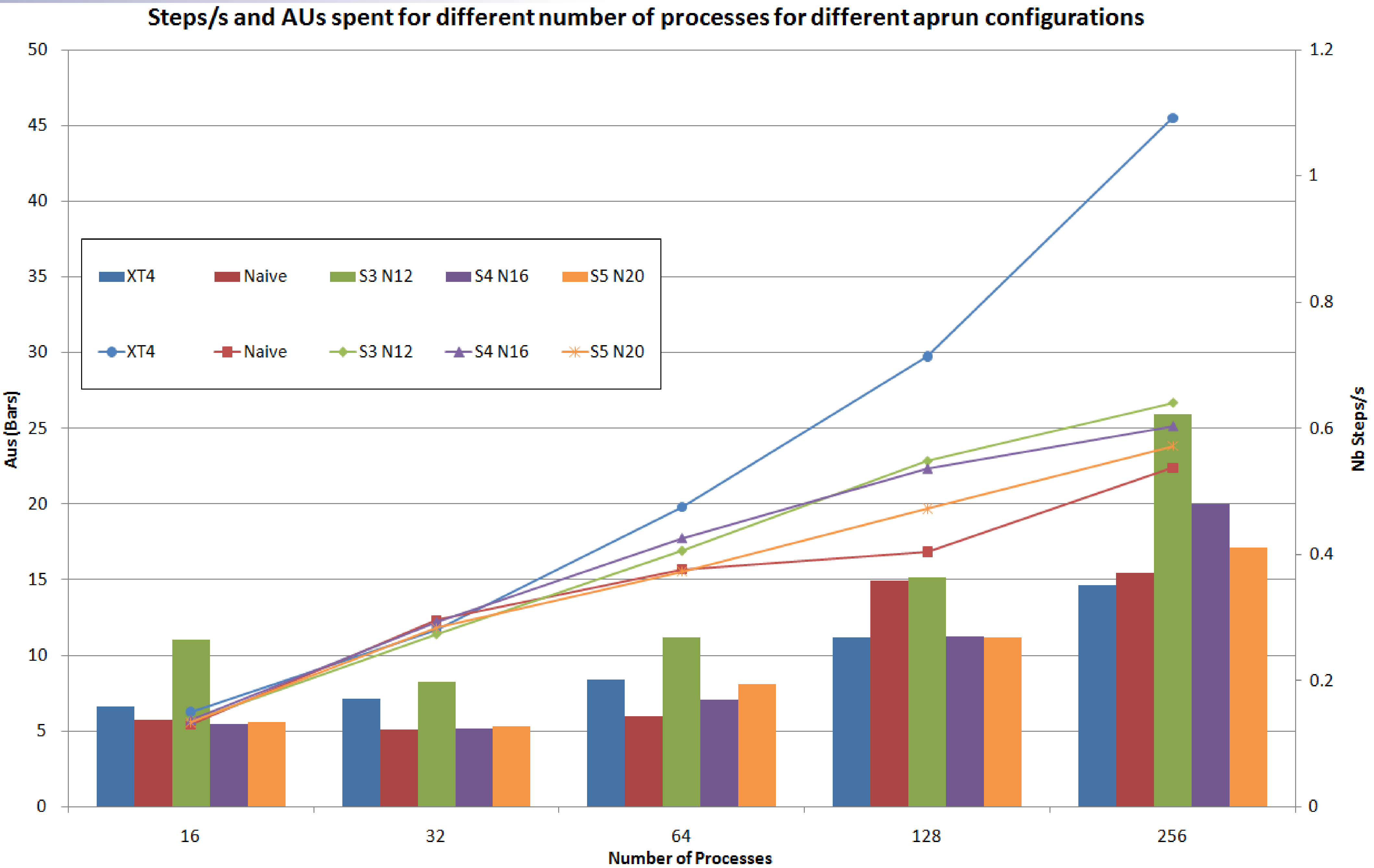
Summary (2/2) – What to do ?

- Actual XT6 is faster when populating the nodes more sparsely, but can be more expensive (due to the Seastar interconnect, same as in XT4, more cores per interconnect)

- To make the best of XT6 :
  - Run your code with different aprun configurations and different nb of processes for a small nb of steps

- New Gemini interconnect should enable faster interconnection

Summary Graph



CASTEP

Chris Armstrong

CASTEP

- Important HECToR code:
  - Lots of HECToR users; used in official benchmarking.
- Simulates materials and molecules at the atomic level.
- 3D FFT transpose, involving MPI_Alltoall, is a major bottleneck.
- Using 144 MPI processes (24*6), al3x3 benchmark.
- Benchmarked 2 versions of the code:
  1. Vanilla, main branch.
  2. SHM-MPI_Alltoall optimisation: one call per node.

CASTEP: XT6 performance, different configurations

- XT4, Vanilla = 1351s (4 procs/node)
- XT4, SHM = **1235s**

Nodes more sparsely populated

ID	XT6 Config.	Vanilla	SHM
A	-n 144 -N 24 –S 6	2559	1815
B	-n 144 -N 12 –S 3	2095	1357
C	-n 144 -N 8 –S 2	1636	1134
D	-n 144 -N 4 –S 1	1236	1099
E	-n 144 -N 4 –S 4	1283	**1081**

- **A-C**: Packing procs into a node causes performance degradation
  - More contention on memory and off-node link.
- **D**: Best “vanilla” performance (than XT4): least contention on memory. Even better performance with SHM.
- **E**: Closest to 4procs/node XT4 config: 4 procs sharing a die, better performance due to increased memory bandwidth.
- **E**: SHM: The best performance: always working from die 0 memory.

HECToR

RESEARCH COUNCILS UK

CASTEP users: "what's the cost?"

- Users have to spend a lot more AUs to match the same kind of performance...
  - Under-populating nodes => using more nodes.
  - Users are charged for whole compute nodes, even if not all cores are actually used.

ID	XT6 Config.	Vanilla	XT6/XT4 AUs*	SHM	XT6/XT4 AUs*
A	-n 144 -N 24 –S 6	2559	0.95	1815	0.74
B	-n 144 -N 12 –S 3	2095	1.56	**1357**	**1.10**
C	-n 144 -N 8 –S 2	1636	1.83	1134	1.38
D	-n 144 -N 4 –S 1	1236	2.76	1099	2.68
E	-n 144 -N 4 –S 4	1283	2.86	1081	2.64

*Based on current figures of 7.5 AU/core/hour XT4, 3.77 AU/core/hour XT6

Probably the most attractive, but we're still getting inferior performance at increased cost.

HECToR

RESEARCH COUNCILS UK

CASTEP: Threaded BLAS/LAPACK

- Can we make the idle cores do some work?
  - Threaded BLAS.
  - XT4 Target: **1235s**, utilising 144 cores (6 XT6 nodes).

XT6 Config.	SHM	XT6/XT4 AUs
-n 24 -N 4 –S 1 –d 6	2086	0.85
-n 48 –N 8 –S 2 –d 3	1767	0.72
-n 36 –N 4 –S 1 –d 6	**1221**	**0.75**

- **Using threaded BLAS allows users to improve cost AND performance.**

HECToR

RESEARCH COUNCILS UK

Summary

- AMD Magny-Cours processor is a “many core” CPU.
  - Will become more common away from supercomputing.
- There are performance benefits:
  - More cores available.
  - More scope for shared-memory & mixed-mode parallelism.
  - Increased memory bandwidth.
  - Greater L3 cache.
- But these are only attainable if users understand:
  - The NUMA architecture.
  - The correct configuration/placement for a job.
  - That multithreading may be the only cost-effective way to run jobs – codes may need to invest more in OpenMP and/or make use of threaded libraries.

HECToR

RESEARCH COUNCILS UK