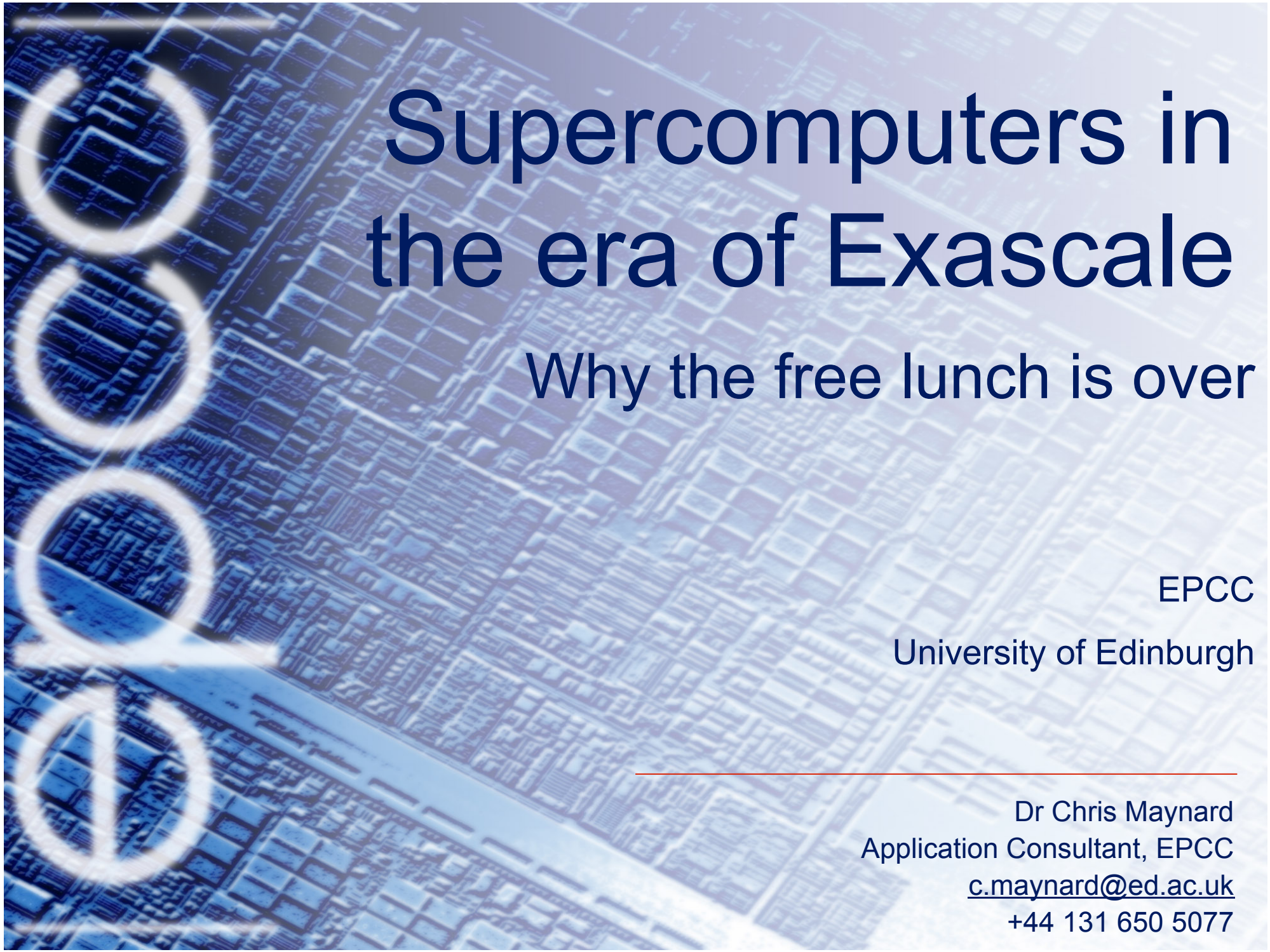


Supercomputers in the era of Exascale

Why the free lunch is over

EPCC

University of Edinburgh

Dr Chris Maynard
Application Consultant, EPCC
c.maynard@ed.ac.uk
+44 131 650 5077

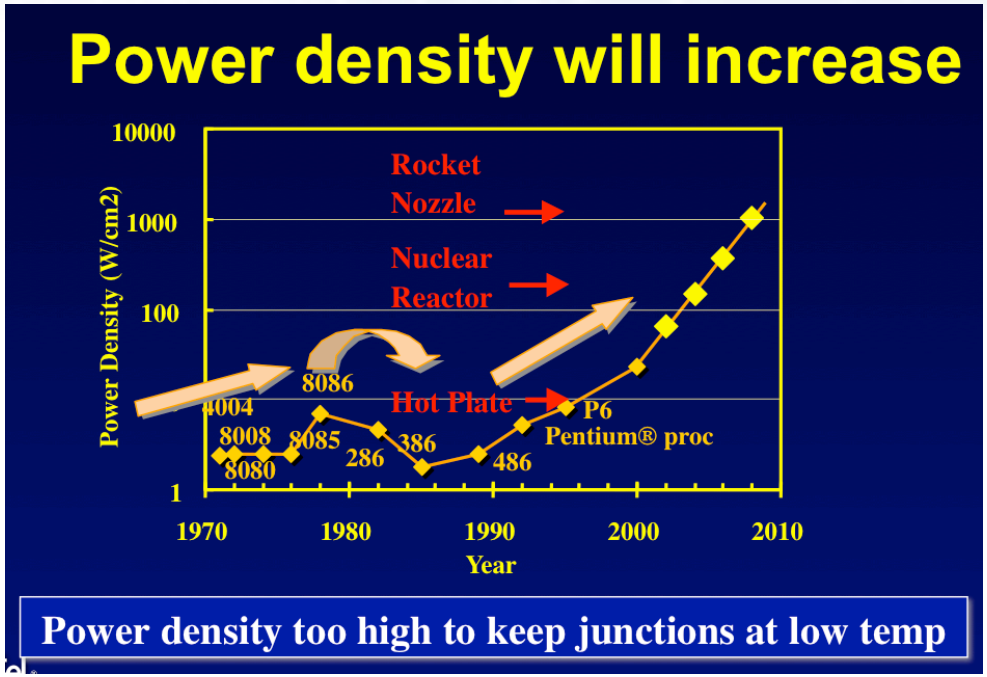
The Future ain't what it used to be

Past performance is not a guide to future performance. The value of the investment and the income deriving from it can go down as well as up and can't be guaranteed. You may get back less than you invested.

Parallel computing is changing

Why concurrency?

Moore's Law: Transistor density doubles every 18 months
→clock speed of processors increases
Power is a problem
→power density at junctions too high to keep temperature low
(Shekhar Borkar 2000)

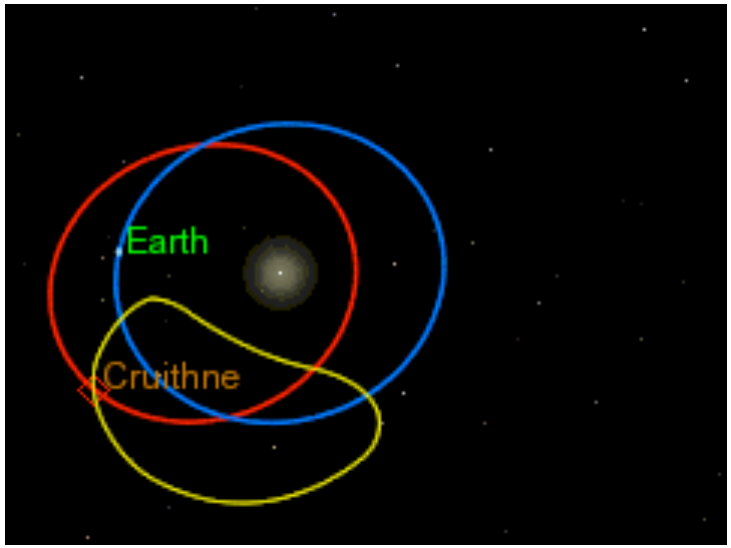


Moore's law reinterpreted
increasing transistor density means increasing *number* (not speed) of cores

Software needs re-writing
Herb Sutter. 2005. *The free lunch is over*

Real problems are difficult to solve

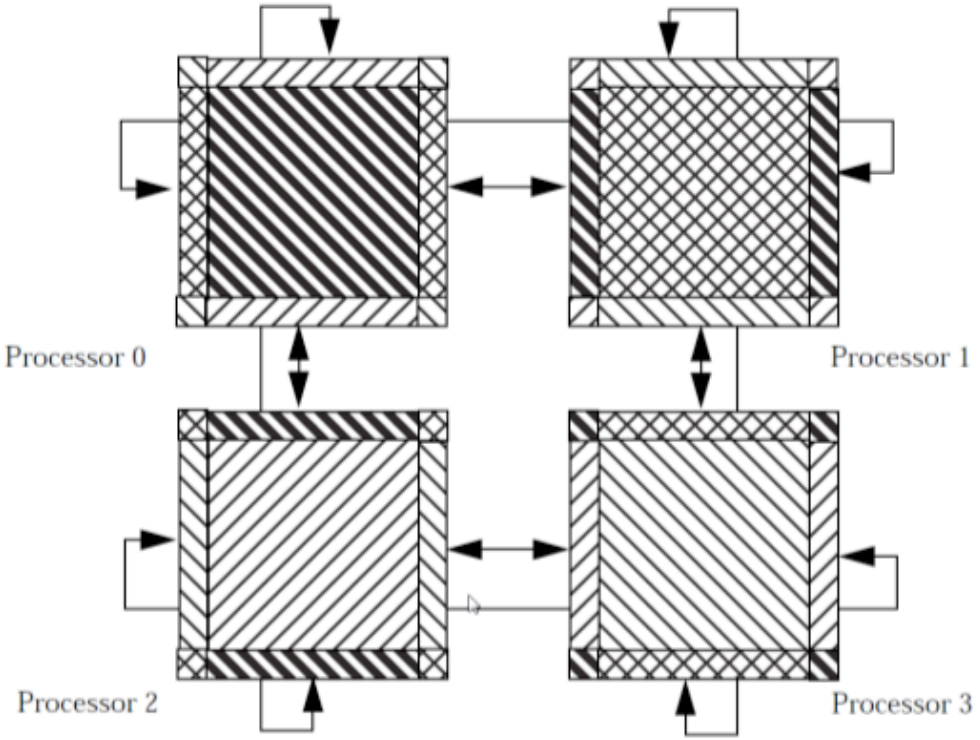
- Problems in **Engineering, Mathematics, Physics, Chemistry, Finance, Biology, Complex systems** represented by differential equations [partial, ordinary, integral etc]
- Example : apple falling - Gravity
  - Newton's universal law of gravitation
  - Newton's second law of motion
  - → All objects fall at the same rate



- Most real problems are too complex for an analytical solution
  - Can only be solved numerically
  - finite difference or finite element
- Three body problem with no symmetry
  - Mass of: Sun (10^{30}Kg) >> Earth (10^{24}Kg) >> Cruithne (10^{14}Kg)

Data parallelism and supercomputers

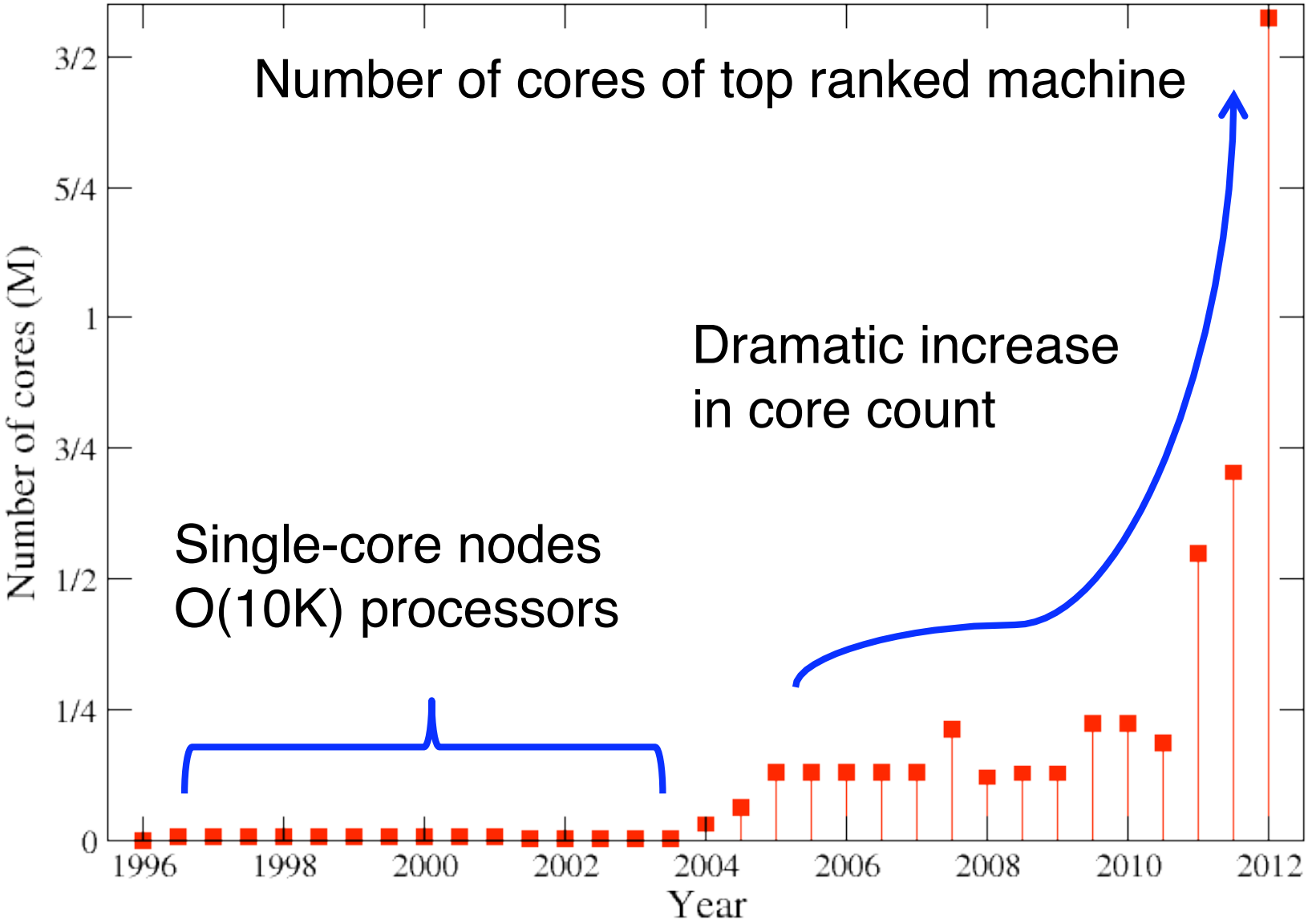
- Many of these problems exhibit *locality*
  - Interaction is between near neighbour elements
- The domain of the problem can be split into subdomains
- Each subdomain can be placed on a processor (with distinct memory)
- Only a small amount of data is remote
- Halo-swap



Anatomy of a supercomputer

- Emphasis over last 15 years on commoditisation
  - HPC machines assembled from PC-like nodes
  - Distinct memory spaces
- network (interconnect) differentiates clusters from supercomputers
  - **`Ethernet → Infiniband → Gemini (Cray)`**
- Each node runs UNIX-like OS
  - explicit inter-node data transfers → message passing
  - majority of programs written in C or FORTRAN
- MPI ubiquitous for distributed data management
  - Some threaded programming
  - mostly OpenMP for shared-memory systems

Trends in Top 500 list



Supercomputers

- ~~No 19~~ No 32. HECToR – UK National supercomputer service. Cray XE6
- 2816 compute nodes
  - AMD 32 core Interlagos processors
  - 90K cores Gemini Interconnect

- ~~No 1~~ No 2. The K Computer, Japan
- 8 core Fujitsu SPARC64 VIIIfx Processors
- 705K cores, Tofu interconnect

New Number 1 Top500

- 'Sequoia' Lawrence Livermore National Lab (USA) IBM BlueGene/Q
  - 96 cabinet, 1,572,864 cores
  - Bespoke interconnect
  - 16 cores per node

- 'DiRAC' 6 rack BG/Q system in Edinburgh
- Number 20 on list
- Dr Peter Boyle, School of Physics, Edinburgh involved in design project

Challenges of Exascale

- Extrapolating current trends predicts an Exascale machine in 2018-2020 AD
  - Currently at 10 Petaflops
  - n.b. *Exascale* rather than *Exaflops*
  - All the extra performance will come from concurrency
  - Currently at O(10^6) cores, consider O(10^9) or (2^{30}) for exascale

Peta ~ 10^{15}
Exa ~ 10^{18}

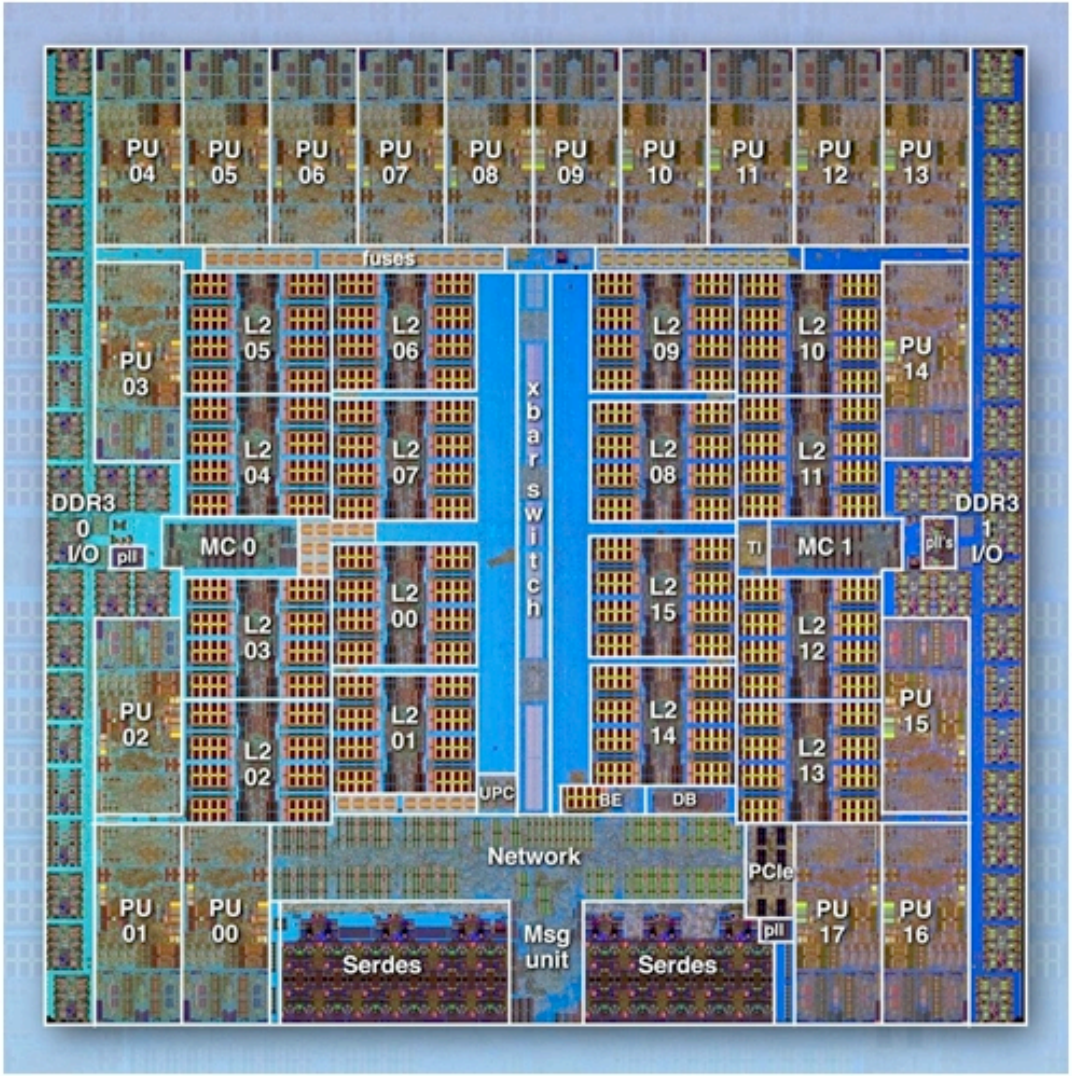
- Total power consumption
  - Most power efficient processors today ~2Gflop/W IBM BG/Q
  - Sequoia ~16.3Pflops, 7.89 MW
  - → 0.5 MW per Petaflop → 0.5 GW for Exascale
    - Same again for cooling, and more for storage!
  - K computer 10.5 Pflops, 12.7 MW ~ 1.21 Mflop/W
  - Cray XE6 ~ 0.3 Mflop/W

Resilient hardware

- Consider failure rates of current hardware
- → MTTI of minutes for Exascale machine
  - This is bad
  - Some work has started on Fault Tolerant algorithms programming environments
- How realistic is this?
  - Depends who you talk to
- Personal opinion
  - its is unlikely that failure rates will be quite so bad
  - it is unlikely fault recover will be devolved to the application developer

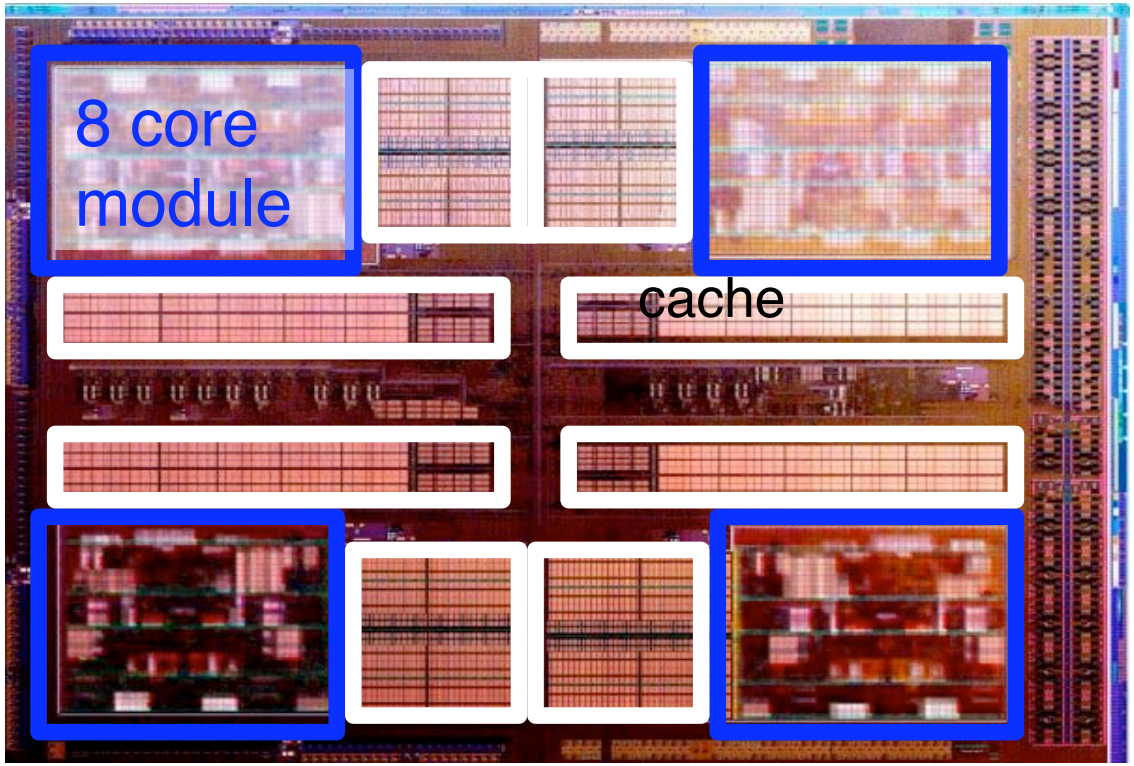
Dark Silicon – IBM BG/Q

45 nm technology
runs at 1.6 GHz
18 cores on die
16 compute cores
1 core IO, communication
1 spare, shut down at Fab
keep power consumption low (55W)
Large area of silicon devoted to cache
hide memory latency



Dark Silicon II – AMD Interlagos

- 32 nm technology
- 32 cores on die
- 2.1 – 2.6 GHz
  - boost to 3.6 GHz
- power 115 W
- Large area for cache



- Active power management
  - Switch of some cores, boost frequency of others
  - Which cores? ideally use multiple modules, maximise silicon use – not possible

Power and resiliency

- If you solve the power problem …
- likely you will solve the resiliency
  - Fundamental changes to circuit design philosophy
  - Resiliency through redundancy - Dark silicon
- More silicon is available than power
  - this is already true
- If hardware doesn't respond, power up some more
  - state information is required however
  - this is system level software

Memory

- Memory access costs time and power
- number of cores increasing faster than memory bandwidth
- most scientific applications require 2 memory words per flop
  - most scientific codes are memory bound
- Use cache to hide this latency
  - Costs power to keep chip memory consistent
- Increasing SIMD/vectorisation
  - SSE, AVX, etc
- These support units are shared by two or more cores
  - using more cores reduces the availability of these units per core
- NUMA - distinct memory hierarchy

AMD Is 0.17 B/f
IBM BG/Q 0.15 B/f
Fujitsu SPARCVIII 0.5 B/f
Nvidia GPU ~ 0.1 B/f
This will decrease in the future

Accelerators

A non-bootable device which is used to compute some aspect of the calculation.
Cell Processor, FPGAs, GPUs, MIC
None are simple to use

Modern compilers allow high level abstraction in programming models for CPUs and still give good performance

Effective utilisation of accelerators requires hardware aware code

Very bad at branches, data transfer host <--> device

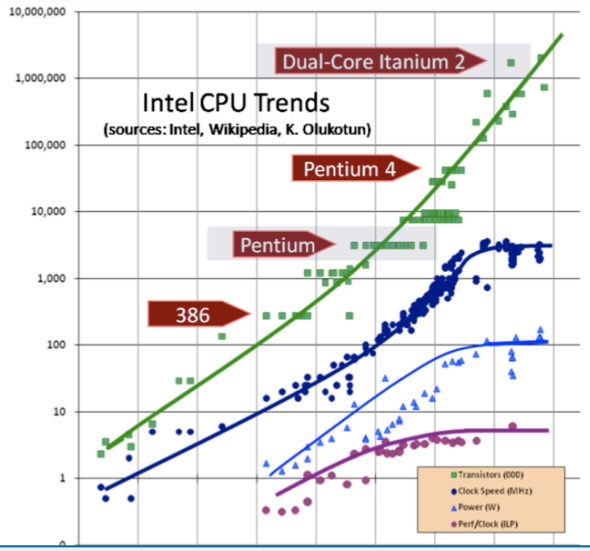
GPUs good at data parallel, high memory bandwidth, extremely high floating point performance, hide memory latency by high concurrency

Programming model for GPUs

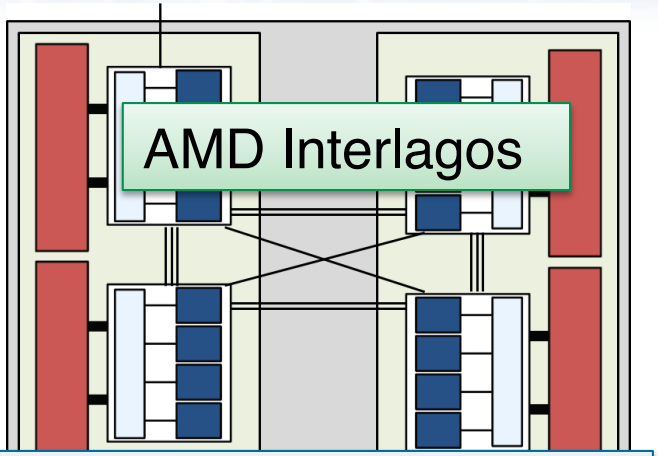
- Most GPUs are NVidia
- Most programs are written in CUDA
  - gives decent performance requires multiple versions of the code
- Open CL
  - Performance is OK in principle can maintain a single code stream
  - in practice this is not always true
- Great is your problem fits into single GPU

- Recent development openACC
  - Directives based programming similar to OpenMP
  - Cray, PGI and CAPS have implementations + Nvidia
- Mixed-mode programming MPI + directives

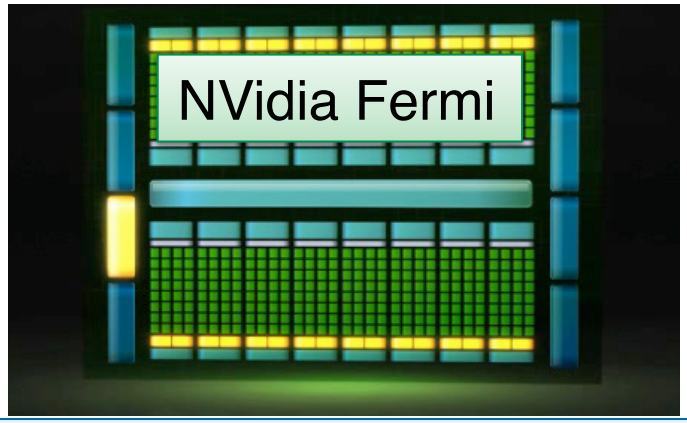
Parallel Programming just got harder!



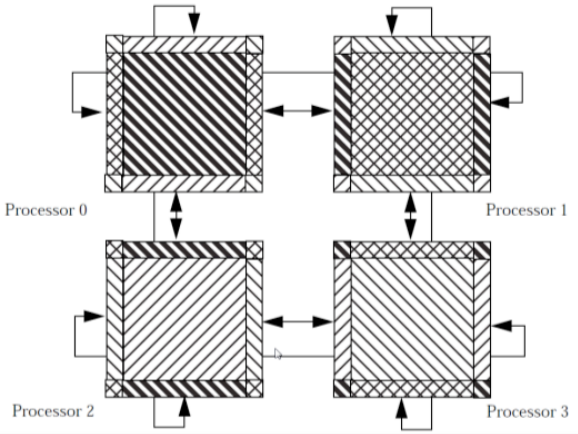
Moore's Law: More not faster



Some cores are more equal than others. NUMA



Heterogeneous Architectures: Accelerators



Data parallel: cores → MPI task

scale 2^{30} heterogeneous cores?

Parallel programming model

- Data parallelism expressed over MPI is the standard model
- For exascale, find 100-1000 more parallelism
  - Scaling existing codes is a challenge
- Is MPI the right model for 2^{30} cores
  - Synchronous two-way communication model
- Should on-node and off-node models be the same?
  - MPI between internode, OpenMP/OpenACC intranode
  - in principle this works, hard to program
- Consider other programming models
  - Partitioned Global Address Space (PGAS)

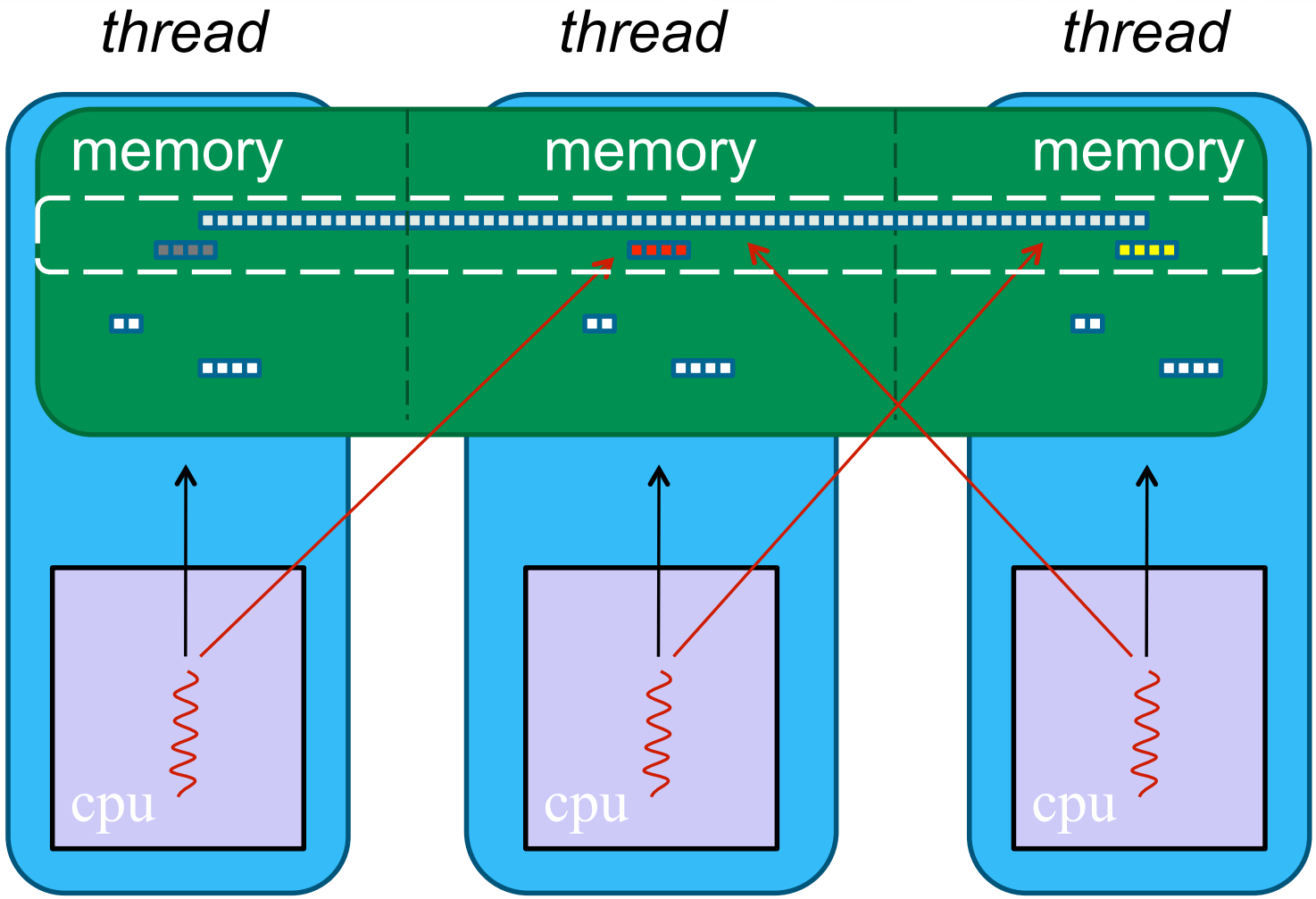
Partition Global Address Space (PGAS)

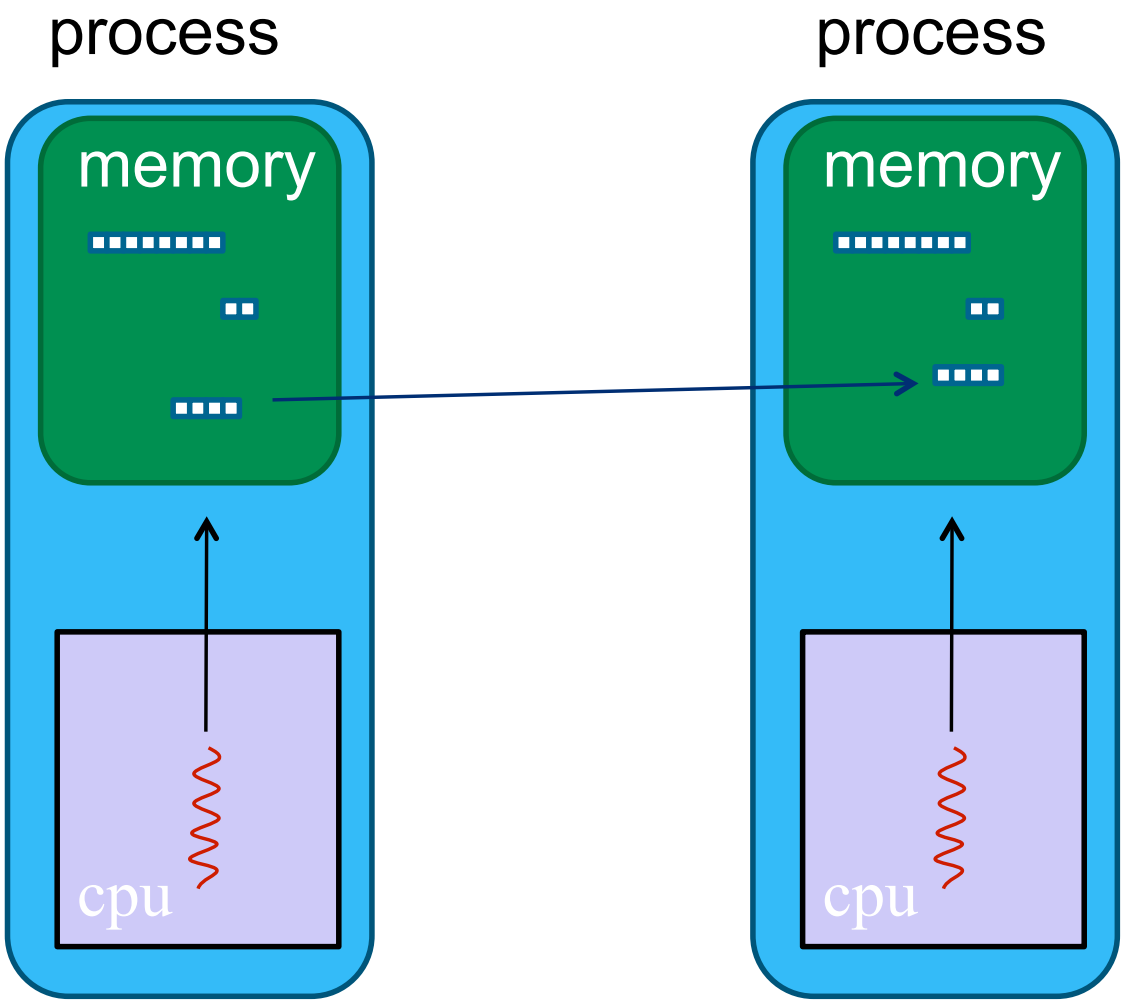
- Distributed memory is globally addressable - GAS
  - Partitioned into shared and local – P
  - Direct read/write access to remote memory
- Asynchronous put/get
  - remote node is not interrupted whilst memory access occurs
  - no explicit buffering
- May map directly onto hardware / Direct compiler support
  - Cray XE6 hdw and Cray compiler
- Language extensions
  - Unified Parallel C (UPC)
  - Co-Array Fortran (CAF)
- Libraries
  - SHMEM (Cray SHMEM)
  - One-sided MPI

How Super is PGAS?

UPC

- Parallel extension to ISO C 99
  - multiple threads with shared and private memory
- Direct addressing of shared memory
- Synchronisation blocking and non-blocking `upc_barrier;`
- work sharing `upc_forall(exp;exp;exp;affinity);`
- private and shared pointers to private and shared objects
- Data distribution of shared arrays
- Cray compiler on XE6 supports UPC
- Portable Berkely UPC compiler
  - built with GCC


thread
thread
thread
memory
memory
memory
cpu
cpu
cpu




```
MPI_Send(a,...,1,…)   MPI_Recv(a,...,0,…)
```

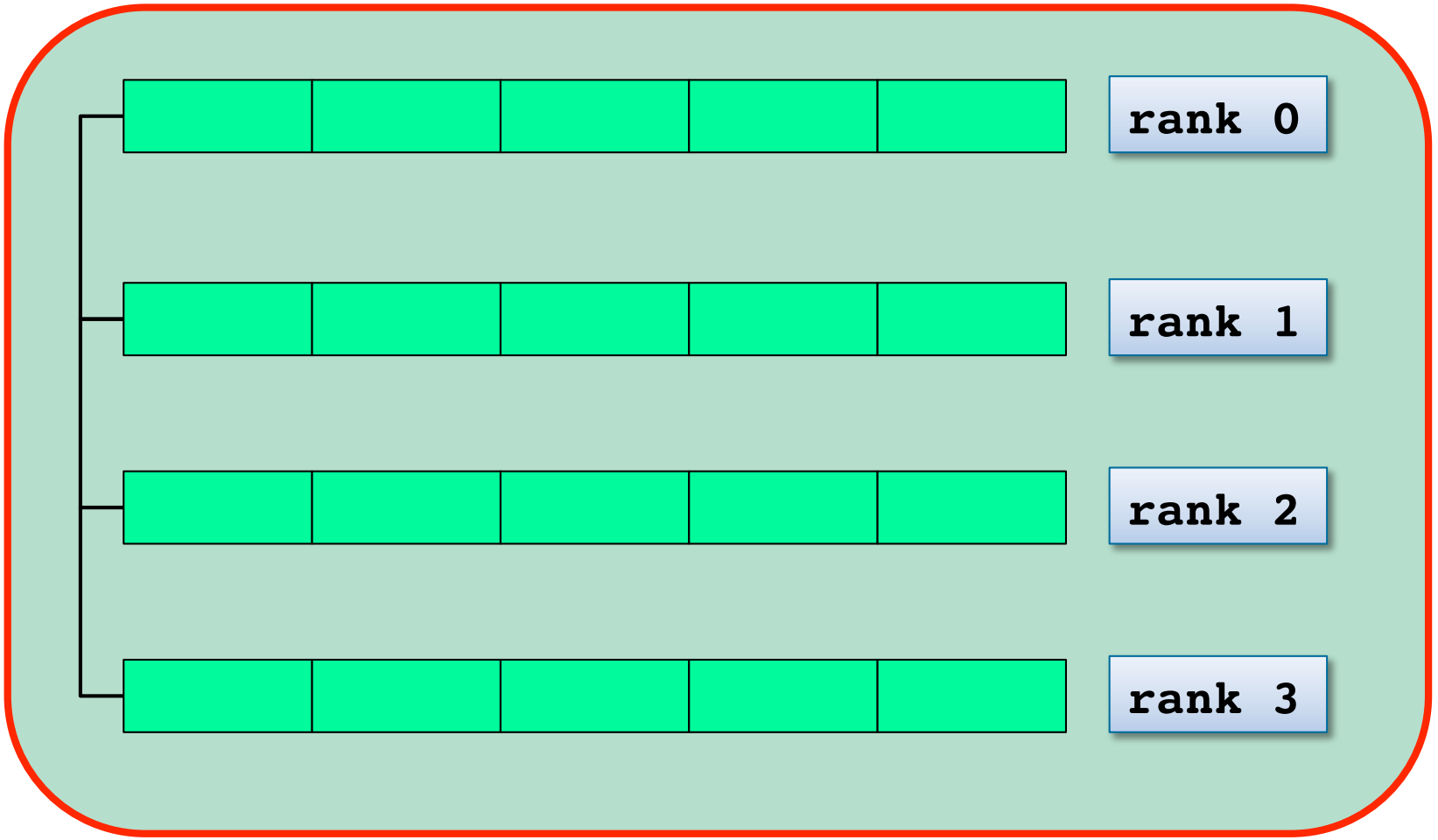
MPI

- Mature, widely used and very portable
- Implemented as calls to an external library
  - Linked **`send`** and **`receive`** messages (both known to the other)
  - Collective calls, Broadcasts, gather/scatter, reductions, synchronisation
- One sided MPI calls in MPI 2 standard
  - Remote memory access (RMA)
  - **`puts`**, **`gets`** and synchronisation
  - Not widely used Data is accessed by MPI RMA calls to the data structure in the window

```
MPI_Get(&localHash,1,MPI_INT,destRank,destPos,1,MPI_INT,win);
```

origin_addr, count and type, where, count and type, Win

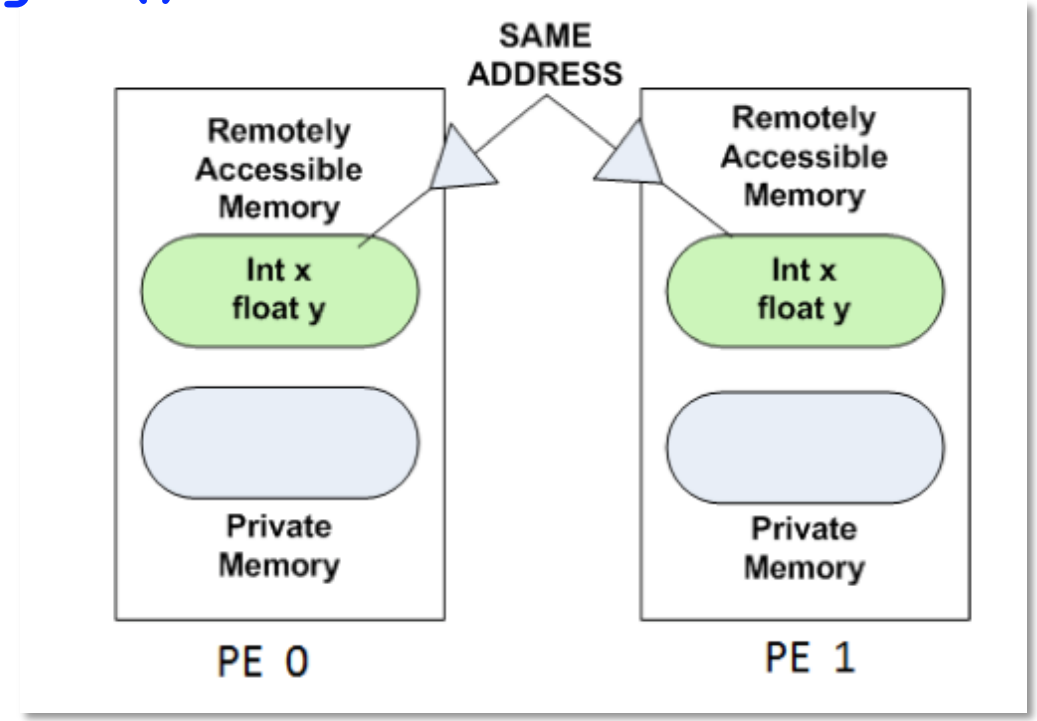
MPI data structure



Window 0

SHMEM

- Symmetric variables for RMA
- Same size, type and relative address on all PEs
- Non-stack variables, global or local static
- Dynamic allocation with **`shmalloc()`**
- **`shmem_put()`** and **`shmem_get()`** routines for RMA
- Cray SHMEM
- OpenSHMEM
- Global synchronisation
  - **`shmem_barrier_all()`**



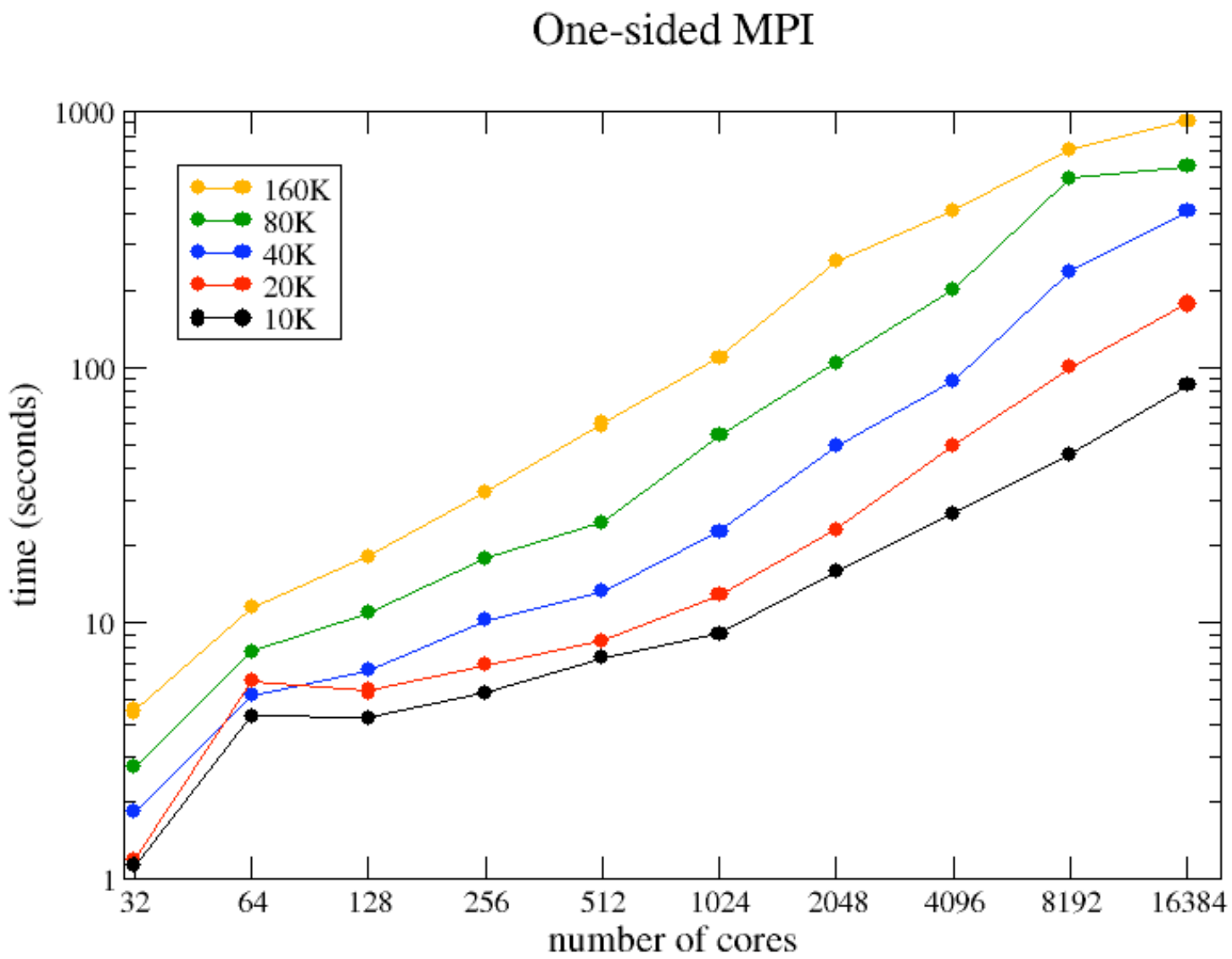
Benchmark

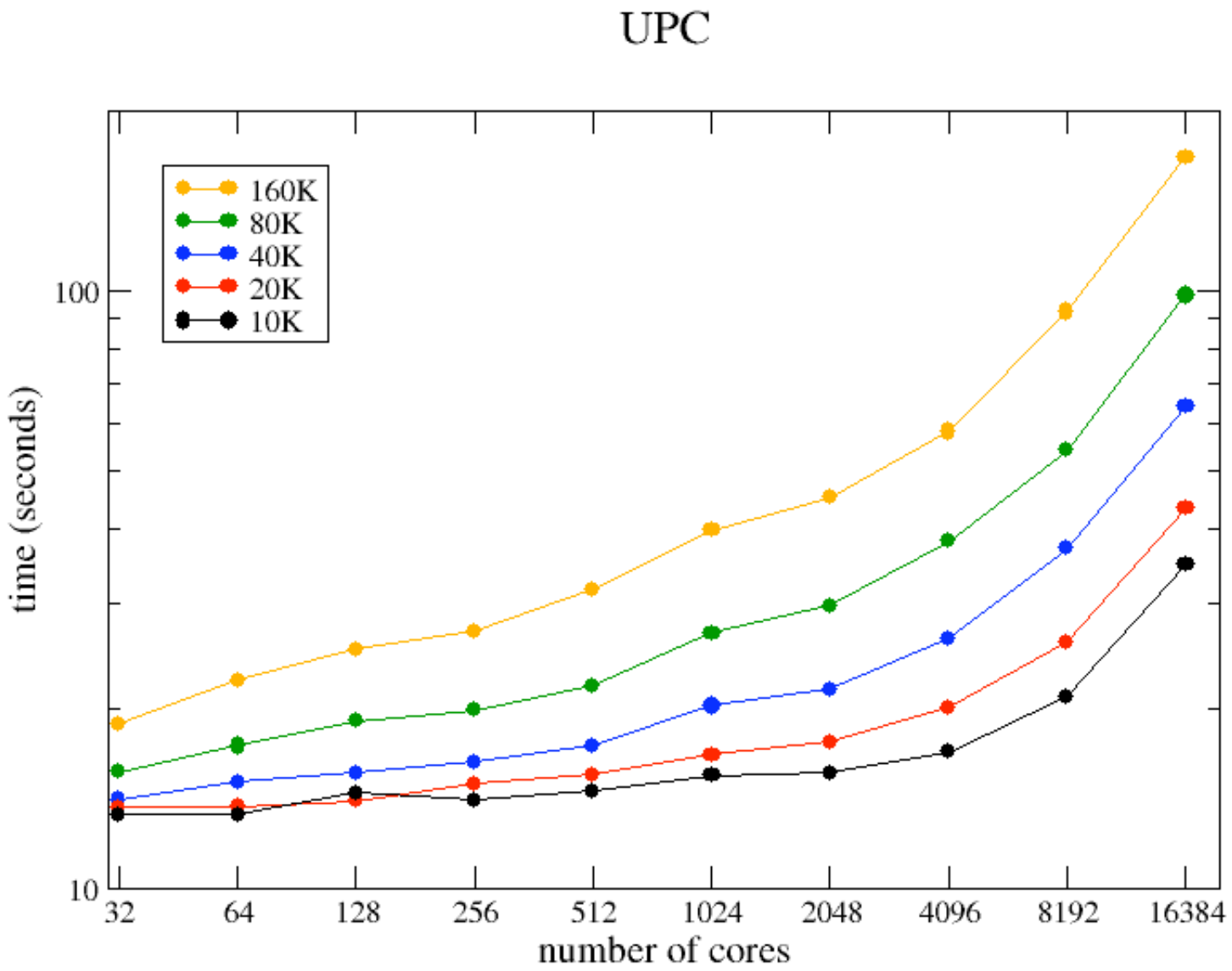
- Distributed hash table – C test code
  - Design of HPCGAP project, symbolic algebra code
  - Integer like objects, computes hash **`func`**, inserts object in table
- Different **`threads`**/**`tasks`**/**`Pes`** read/write access same data object
  - Race condition
  - Use locks to protect data elements
  - Different semantics/properties for **`UPC`**/**`MPI`**/**`SHMEM`**
    - basically same, lock entire data block with affinity to a particular thread

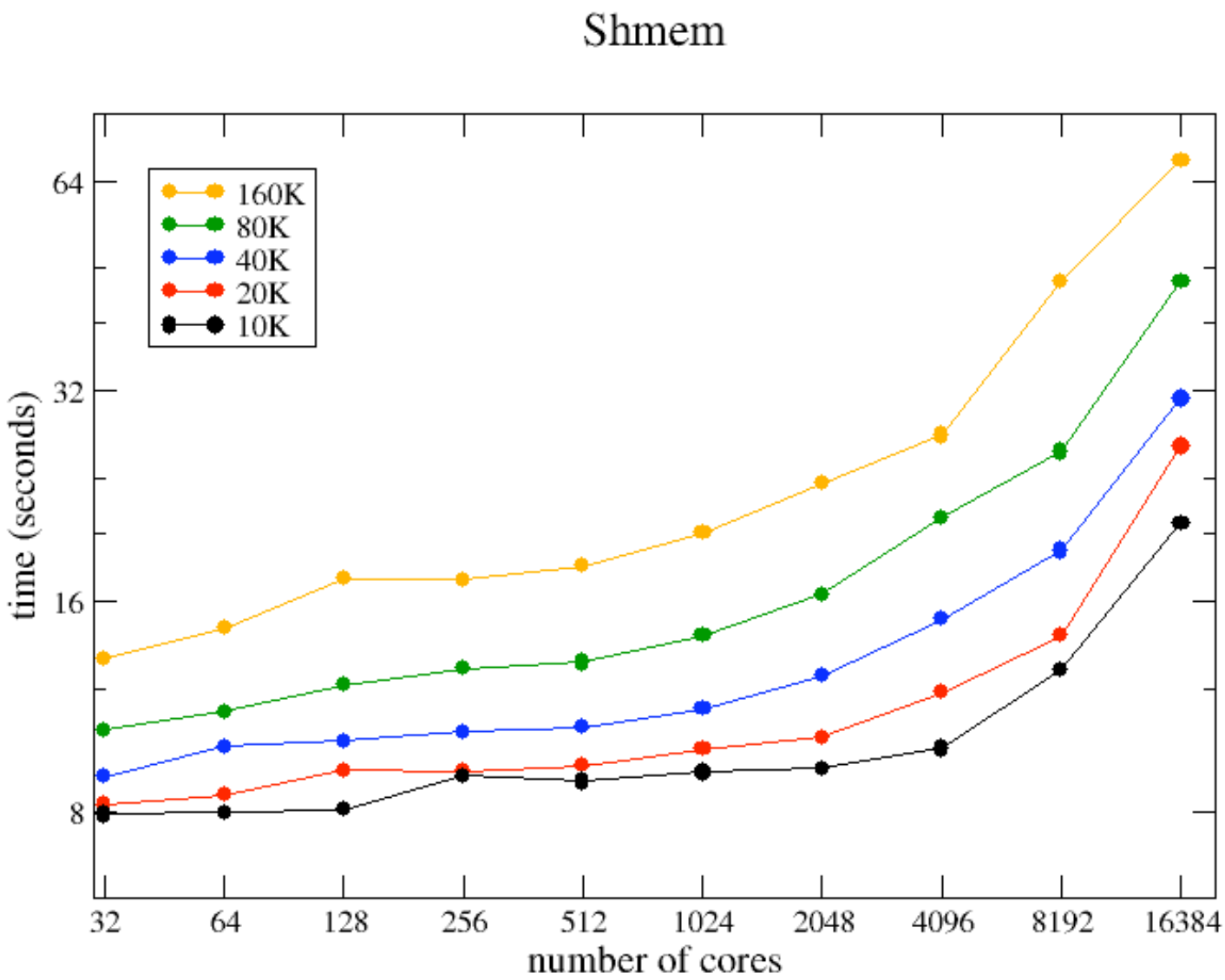
Hardware and Environment

- HECToR – UK National supercomputer service. Cray XE6
- All codes compiled with Cray C compiler (and Cray SHMEM)

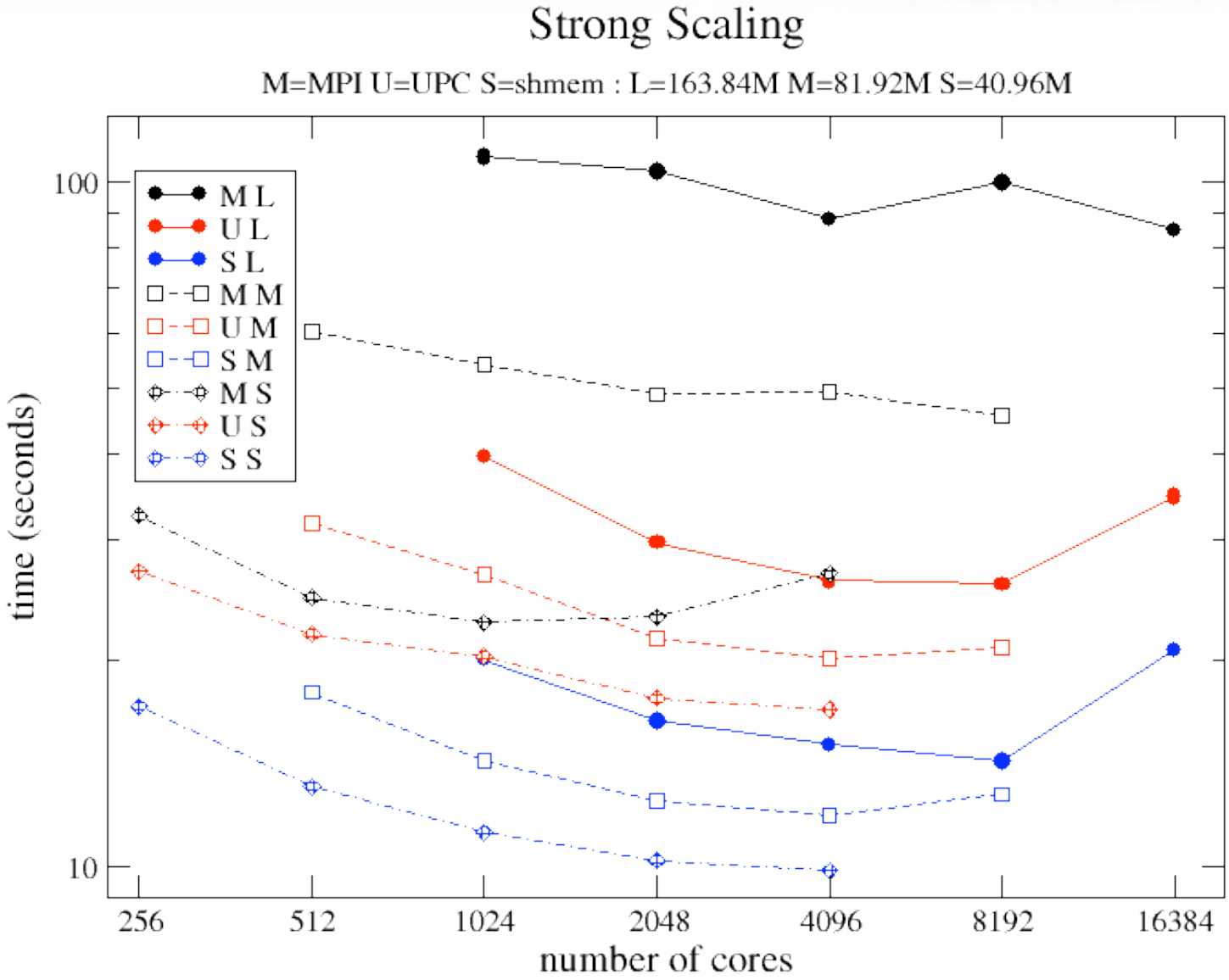
- benchmarks run on XE6 Gemini interconnect
  - Phase 3 AMD 32 core interlagos
  - Phase 2 AMD 24 core Magny-cours
- Integer object of 8 bytes with two passes of the hash table
- Weak scaling results show
  - wall clock time versus number of cores
  - fixed size of local hash table
  - different sizes are shown as different curves on the plots


One-sided MPI
1000
100
10
1
time (seconds)
160K
80K
40K
20K
10K
32
64
128
256
512
1024
2048
4096
8192
16384
number of cores


UPC
160K
80K
40K
20K
10K
100
10
time (seconds)
32
64
128
256
512
1024
2048
4096
8192
16384
number of cores


Shmem
160K
80K
40K
20K
10K
64
32
16
8
time (seconds)
32
64
128
256
512
1024
2048
4096
8192
16384
number of cores

Strong scaling



One-sided MPI performance

- Why is it so bad?

- It really sucks
- Not an implementation issue!

- Other PGAS RMA models have memory restrictions
  - SHMEM “symmetric” variables
  - UPC static or global variables
  - CAF non-stack memory
- One-sided MPI widow can be constructed from any memory
  - This is hard to implement efficiently (apparently in MPICH2)
  - Likely cause poor performance

Conclusions

- Parallel computing is becoming much more complex
  - Free lunch is over, no more performance gains via clock speed
- Power is the problem (micro and macro) – hardware
  - Resiliency, but it is correlated – hardware
- Assuming the power problem is solved …
- Software challenge
  - find much more parallelism in our application
  - complex memory hierarchies
  - heterogeneous architecture
  - Programming model, on-node and off-node,
  - Two stage programming model? MPI/PGAS

Final thoughts …

- Free lunch is over for performance but …
- All applications
  - supercomputer/desktop/mobile device)
  - require re-writing for increased parallelism
  - Lots of different activity out there
  - Who is going to do this?
- This is your *lunch ticket* (Job for life!)