# High Performance Computing - The Future

Dr M. Probert

Autumn Term 2015

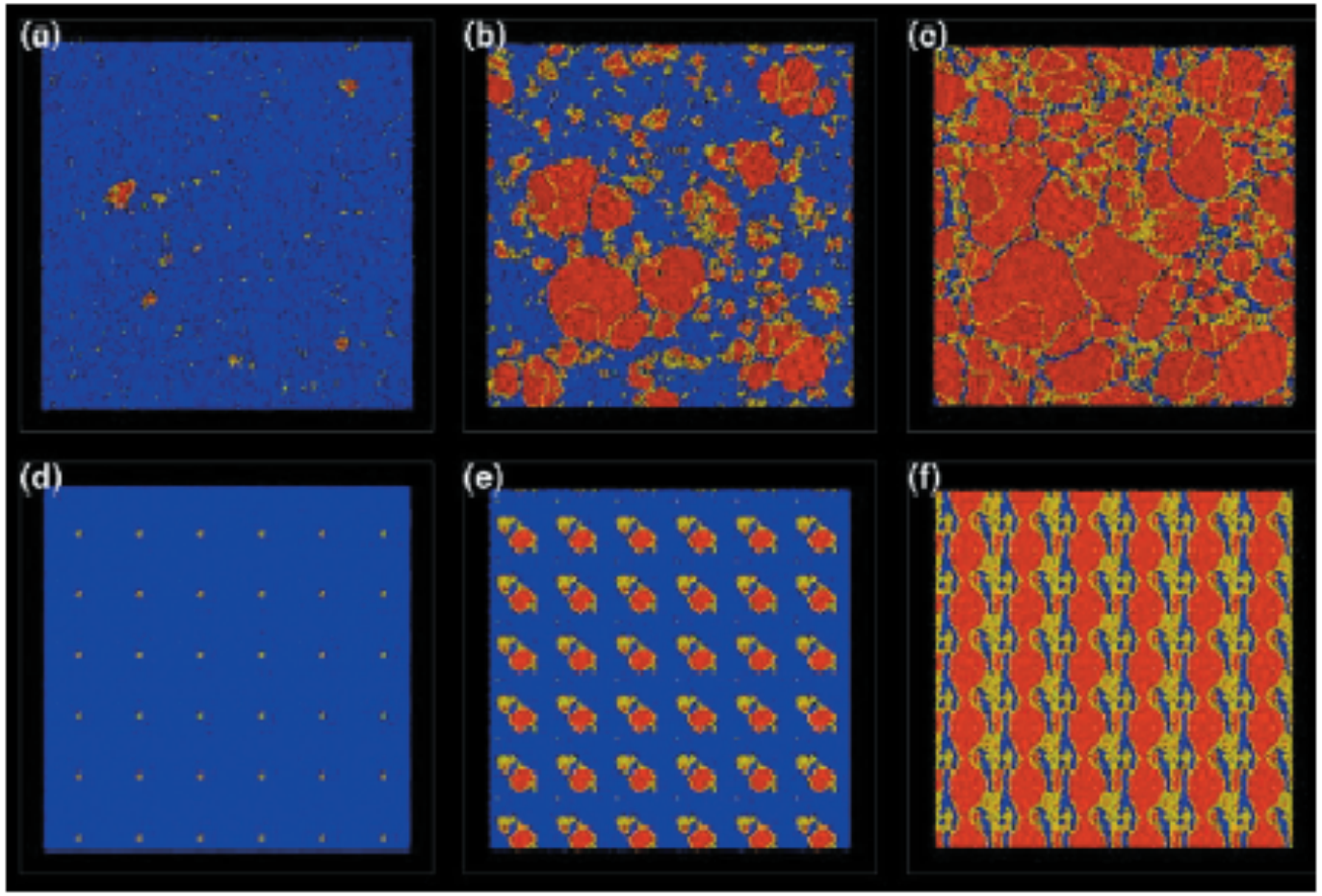
# Overview

- Big Computing / Big Data
- HPC Languages
- GPU programming
- New CPUs

# Big Computing

# Why do we need Big Computing?

- Domain decomposed MD with dynamic non-regular domains




Snapshots from simulations of solidification in tantalum. The top sequence displays nucleation (a) and growth (b) occurring in a 16,372,000-atom simulation, resulting in a realistic distribution of grains and grain boundaries (c). The same process modeled using 64,000 atoms (d–f) produced the artificial final structure shown in (f).

# Beowulf

- Beowulf designs are cheap and popular
  - Hardly use this name anymore – become so ubiquitous
  - Rapid growth since mid 1990s – large part of Top500
  - Enabled by powerful and cheap CPUs *and* developments in network technology (Gigabit Ethernet, InfiniBand, etc.)
- Typically a "fast compute, slow interconnect" cluster
  - Challenges to large-scale parallelism
  - Need lots of latency hiding to get good scaling
  - Also a problem in many cluster-based solutions
- Hence interest in slow/low-power CPUs
  - E.g. Intel Atom and IBM Power PC CPUs
  - High packing density, lower running costs,
  - And easier to get codes to scale!

# Tianhe-2 – current #1

- A hybrid architecture = CPU+Accelerator
  - 3,120,000 cores in 16,000 nodes
  - 2 Ivy Bridge + 3 Xeon-Phi per node
  - 54.9 TFLOP peak
- Needs lot of power = 17.8 MW
- A challenging machine to program
  - Need hybrid MPI + OpenACC or OpenMP
- Is this headline grabbing or serious science?

# Titan – current #2

- Another hybrid architecture – CPU+GPU
  - 18,688 nodes each containing 1 AMD Opteron 16-core CPUs + 1 nVdida K20 GPUs
  - plus Cray Gemini interconnect
  - 27.1 TFLOP peak & 8.2 MW power
  - A more challenging machine to program
    - Need hybrid MPI + CUDA
- Many of the Top500 are hybrid machines
  - Not a trend that is going away soon
  - But we desperately need some common open standards to get portability and longevity of codes

# ExaScale Computing

- Currently plans are being drawn up as to how to get to ExaScale
  - $10^{18}$ FLOPs by 2018 according to exponential
  - Power/cooling limitations
  - Programming methods
  - Component reliability - MTBF
  - Parallel scaling challenges
- What science will become capable? How to manage the data generated?

# The “Grid”

- Grid computing has it origins in a 1998 book by Carl Kesselman and Ian Foster called "The Grid: Blueprint for a New Computing Infrastructure.“
- Basic idea – to make the provision of HPC resources as ubiquitous as the electrical grid
  - When you turn on an electrical switch you don’t know and don’t care where the power has come from – so how about HPC?!
  - Hence with so many wasted clock-cycles with modern PCs running screen savers, why not harness that for more useful ends?

# The Grid Idea

- So anyone with spare computing power could donate it (or sell it?) to the Grid
- And anyone needing extra computing power could access (or buy?) it from the Grid
- So the Grid is essentially *middleware* – a broker service between supplier and customer
- Currently fashionable – lot of e-science money is being spent on developing Grid technology
  - But who will use it? Does it work? What are the advantages? What are the risks?

# Grid Advantages

- LHC at CERN generates huge datasets – 35 TB/day – which need to be stored and analysed
  - Hence CERN is at the forefront of implementing Grid technology – cannot store & process such large amounts of data – needs to be able to distribute it around the world to get local storage and analysis
  - Large strain on networks – dedicated 10 Gbit/s fibre optic links to 11 “Tier 1” institutions
  - Called ‘LHC Computing Grid’ – a practical way of managing the volumes of data to be generated
  - But is it really “Grid” as originally envisaged?

# Grid Users / Big Data

- Who else?
  - Large data sets requiring distributed processing
  - Issue with trust – not standard PC types.

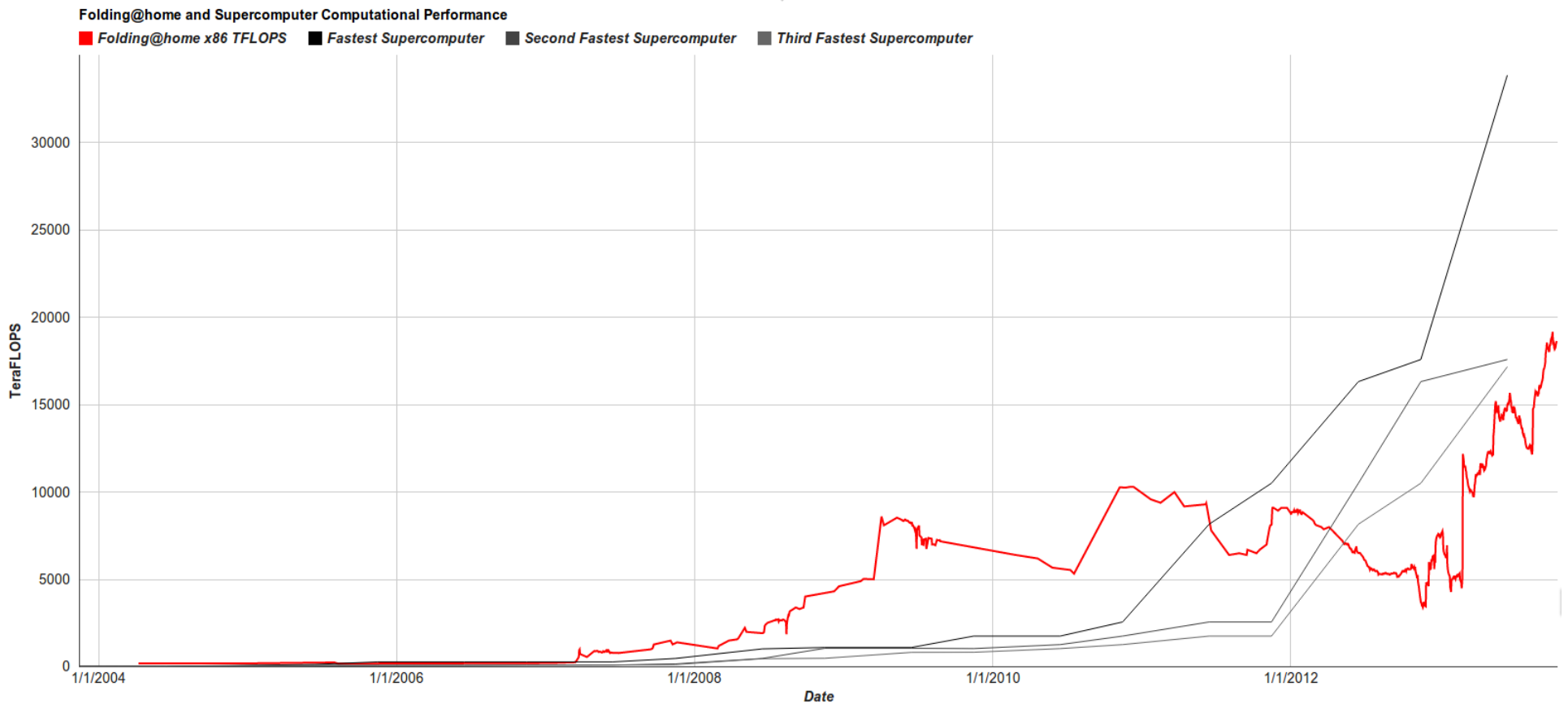
- Square Kilometer Array (SKA)
  - Building due to start in 2018 in South Africa & Australia with total x50 sensitivity and 10,000x faster than anything else
  - On-site computer power of $10^8$ PCs
  - Generate 10x global internet traffic
    - Output 10 GB/s = 36 TB/hour …

- National Grid Service
  - Has standard suite of software available for all users

# Distributed Computing Projects

- Distributed computing projects, such as SETI@home, etc. are already "Grid-like".
- Folding@Home
  - 1st computing project ever to sustain 1 PFLOP (Sept07)
  - Now running at 40 PFLOPs with 108,000 active CPUs + 64,000 active GPUs!
  - Was also PS3 until Sony 2012 stopped support
  - Data generated has produced over 115 papers so far
  - MPI parallel since 2006, threads and OpenCL in 2010
- Lots of other "@home" projects …

# Power of Distributed Computing



# HPC Languages

# Dedicated Language?

- Alternative to using a common language (e.g. C/ C++ or Fortran) + libraries or directives
- Tried in early days but lost out
- DARPA started High Productivity Computing Systems program in 2004 to build peta-scale
  - IBM Roadrunner 2008 at Los Alamos USA
  - Develop hardware + languages + o/s + file system
- Languages included
  - Fortress (Sun), Chapel (Cray), X10 (IBM)
  - All examples of PGAS (partitioned global address space) languages
- Or improve traditional languages?

# Fortress (Sun)

- In mid-2000s there was a lot of effort to rebrand Java as a HPC language (Java Grande) but:
  - No IEEE 754 support + few intrinsic math functions
  - Not in MPI or OpenMP standards
  - Slow unless converted to native code
  - Java Grande Forum died – brief revival when Java went GPL (end 06) but then nothing …
- SUN created a new HPC language – "Fortress"
  - Designed to be a secure Fortran that was intrinsically parallel and type-safe with pseudocode syntax
  - Started in 2005, open source in 2007, but thenp roblems with JVM licensing meant Oracle decided to drop the project in 2012 so now looks dead …

# Chapel (Cray)

- Designed to separate algorithms from data representation
- Multi-threaded parallelism for data + task + nested parallelism
  - Based upon HPF ideas
- With support for OOP
- Started in 2009, open source, latest version 1.11 in April 2015

# X10 (IBM)

- Focus on concurrency and distribution with OOP like Java or C#
- Asynchronous PGAS
- Uses parent + child to handle locks/ race conditions
  - Parent can wait for child but not v.v.
- Can use JVM or compile to native code
- Started in 2004, open source, latest version 2.5.3 in June 2015

# UPC (Unified Parallel C)

- Based upon C99 with SPMD model
- Can handle either shared or distributed memory machines
  - An explicitly parallel execution model
  - Appears as shared address space to programmer
    - any variable can be r/w from any processor but physically associated with a single processor
  - Synchronization primitives and a memory consistency model
  - Memory management primitives

# Fortran 2003

- IEEE exception handling
- Allocatables in derived types
- Interoperability with C
- More OOP:
  - procedure pointers and structure components, structure finalization, type extension and inheritance, polymorphism
- Access to environment (similar to argc etc)
- Asynchronous I/O
- Almost all features now available in gfortran (only 2 bits missing v4.9)

# Fortran 2008 – Co-Arrays

- Allows SPMD within Fortran
  - easier to use than MPI
  - designed for data decomposition
- Example

```
REAL, DIMENSION(N)[*] :: X,Y
X(:) = Y(:)[Q]
```

  - Additional [] shows that this item is a co-array and is distributed
  - Second line shows how to copy values from 1 "memory image" to another (c.f. MPI_Send/Recv)
- Available in gfortran since v4.6
- Most of F2008 now in gfortran 4.9 (except 7 bits)

# Future Fortran?

- Fortran 2015 draft standard published
  - Now in committee stage until 2018
- Two major additions:
  - TS29113 (Further Interoperability with C), was approved in 2012 and available in Intel ifc v16
  - TS18508 (Additional Parallel Features in Fortran), extends coarrays. Including DO CONCURRENT and SELECT RANK.
- Plus minor improvements
  - to environment variables, STOP commands, etc.

# GPU Programming

# GPU Programming - CUDA

- GPU has many inherently parallel features
- nVidia has released CUDA
  - a standard API for high level languages with support for Windows, Mac and Linux
  - SDK supports PathScale Open64 C compiler + third-party wrappers available for Python, .Net and Java, etc.
  - v4 (May 2012) *aka* Kepler – with 3x performance/Watt of Fermi (v3, Jan 2010)
  - Supports GDR5 with ECC for proper science
- Also OpenCL, OpenACC and OpenMP v4 for non-vendor specific approaches!

# OpenCL

- A language for data and task parallel computing using CPUs and GPUs
  - Created by Apple and based on c99
  - released as open standard in June 2008
  - Built into MacOS since v10.6 (“Snow Leopard”)
  - Works on NVidia, AMD, IBM, S3, etc
  - Platform independent cf. OpenGL
- Low level – even more so than CUDA – but device independent and an open standard ...
- Microsoft has released DirectCompute as set of DirectX APIs to enable GPU usage in Windows

# OpenACC

- CUDA and OpenCL have steep learning curve
  - Need to know about device memory etc.
- OpenACC is a compiler directive based approach
  - Hence much higher level, more like OpenMP
  - Minimal change to existing codes
  - Supports Fortran, C/C++, etc
  - Backend generates CUDA or OpenCL as required!
  - Originally proprietary – CAPS – but open standard
  - Similar ideas in Portland Group ‘accelerator model’...

# New CPUs

# Low Power HPC

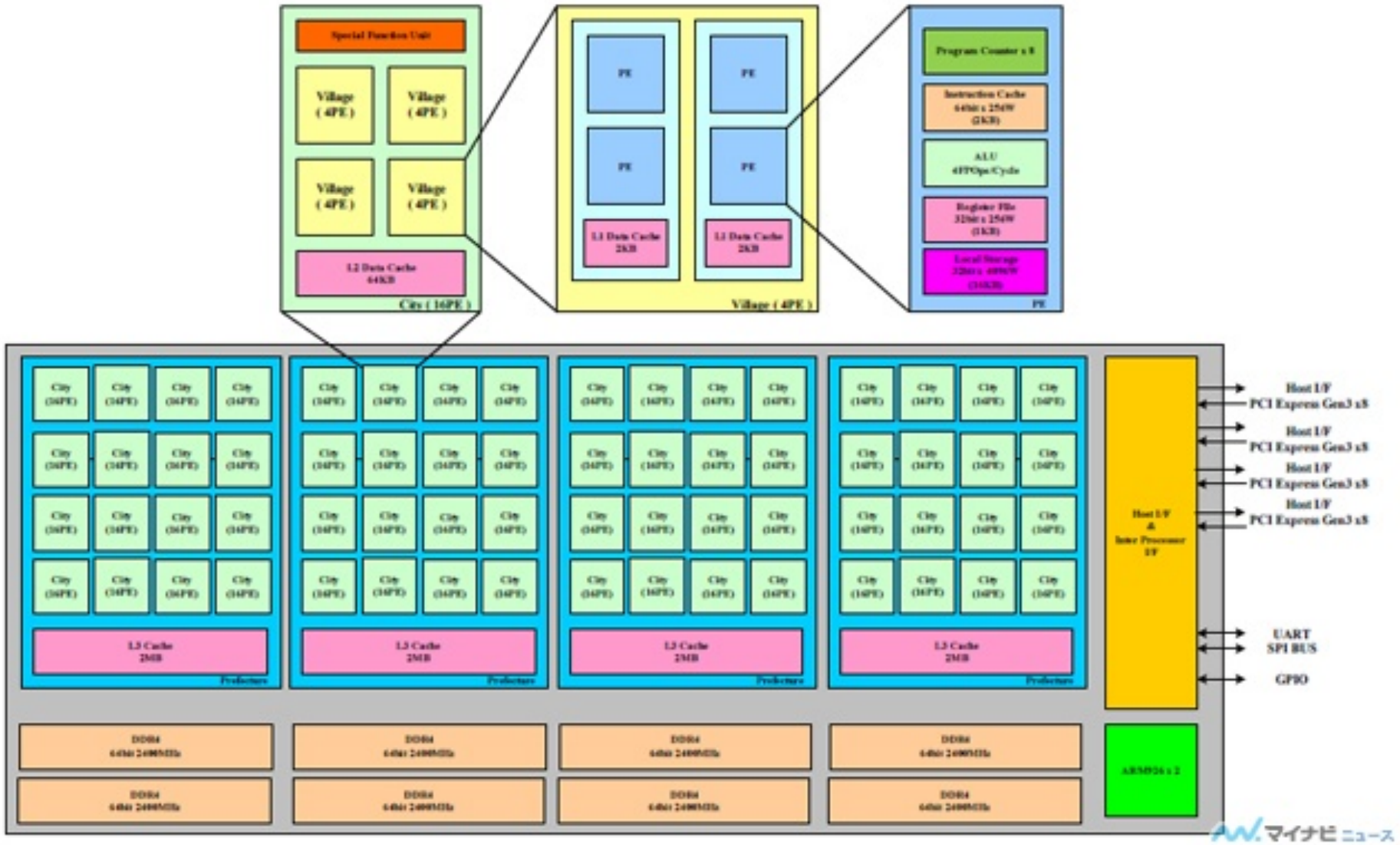
- Green500 focuses on power-per-Watt (latest Nov 2015)
  - Top machine (at RIKEN) has 7.0 GFLOP/W
  - Top places dominated by hybrid designs with novel combinations of hardware e.g.
  - Xeon + PEZY-SC (top 1+2+3) or AMD Fire Pro or nVidia K40 accelerators
- Exascale?
  - If could scale #1 machine to Exascale it would take 143 MW to run
  - When Green500 was launched in 2007 it was projected to take 3000 MW => 21x better!
- Also look at SWaP (space, wattage and performance) = performance/(space*power)

# PEZY-SC

- 2nd gen SC=Super Computer
  - Launched Sept 2014
  - 1024 cores, 65 W and 1.5 TFLOP with double precision – twice the power efficiency of K40
  - Programmed using OpenCL

- Japanese company launched in 2010
  - Peta, Exa, Zeta, Yotta initials!
  - PEZY-SC2  with 4096 cores in developement

# PEZY-SC



# PEZY-SC in Shoubu at RIKEN

# AMD

- FirePro S9170
  - Launched in July 2015
  - Peak performance 2.6 TFLOP and 32 GB GDDR5
  - Designed to beat nVidia – claim OpenCL has 40% better performance with DGEMM than nVidia K80 dual GPU card
  - 275 W and $3100 (vs K80 $4200 = 24 GB)

- APU line (Accelerated Processing Unit)
  - Current (3rd gen) best is ‘Kaveri’ (July 2014)
  - Fusion of 4 CPU + 8 GPU cores in a single package for 95 W and ~£100 - flat memory model
  - Used in PS4 & Xbox One – almost ‘system on a chip’

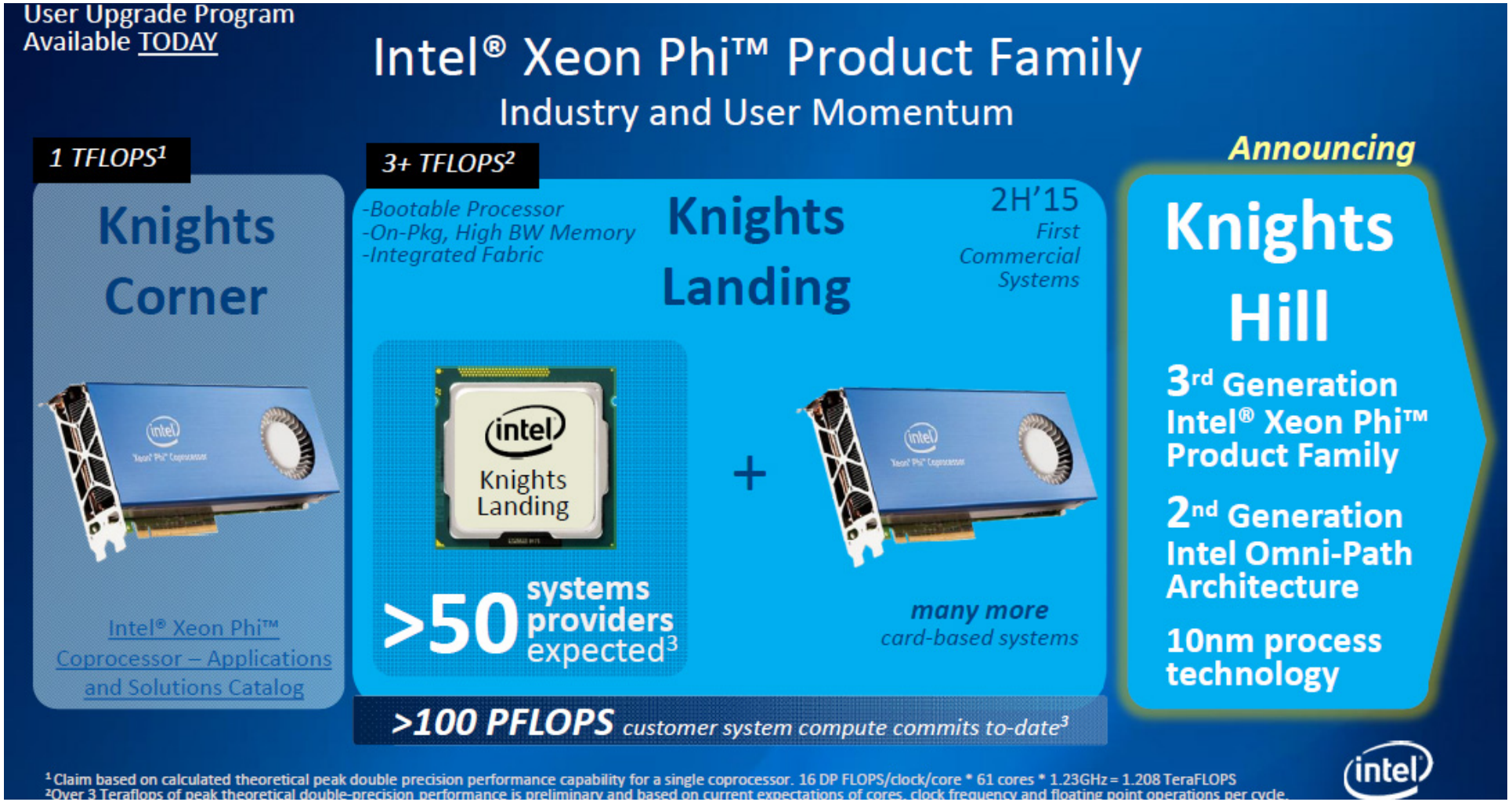
# Intel

- More multi-core processors
  - Experimental 48-core “Single Chip Cloud Computer” released mid-2010 – not on roadmap
  - Design “can be scaled to 100 cores” claim
    - Limited by on-chip network + issues with cache coherency
    - 6x4 array of Pentium tiles
- Larabee
  - CPU+GPU fusion but with x86 instruction set
    - Full cache coherency
    - Much more flexible and easier to programme
    - v1 was due early 2010 but not released as uncompetitive with GPUs – project used as basis for …

# Intel Xeon Phi (Knight's Corner)

- "Many Integrated Core Architecture"
  - Built upon Larabee + SCC + 80-core Tera-scale
  - 60 core accelerator in 2013 – 1 TFLOP dp @300W
  - c.f. Intel ASCI Red = first TeraFLOP supercomputer in 1997 cost $55 million with 10,000 Pentiums!
- Easy programming model as x86 architecture
- 2nd Gen = Knight's Landing mass release 2016
  - 72 Atom cores with 4 threads/core so 2.8 TFLOP dp@200 W target and integrated memory
  - 2 versions – accelerator and host CPU

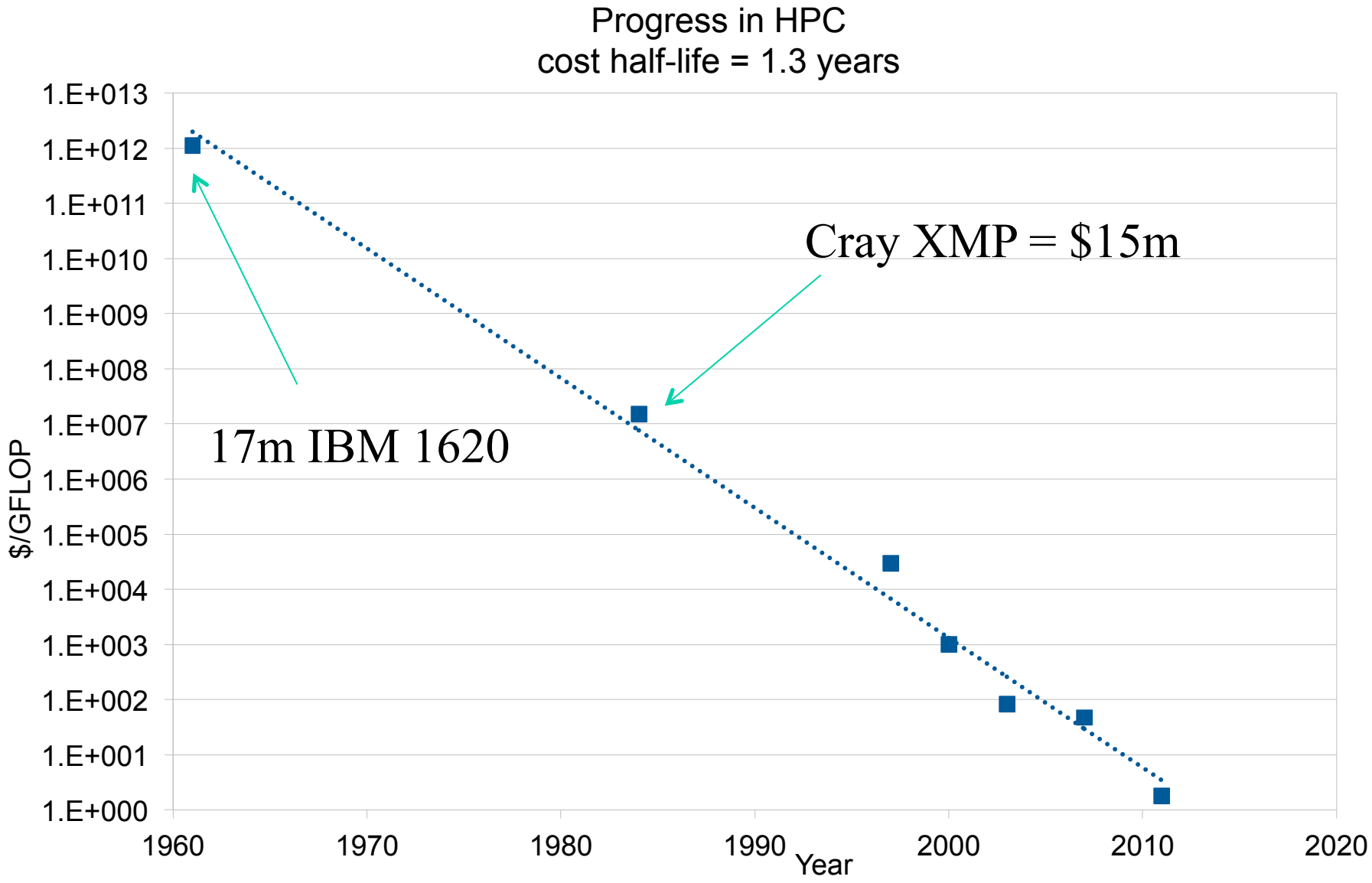
# Intel Xeon Phi Roadmap



# Affordable Supercomputing?

- Cost/FLOP has been going down for many years
- Recent developments in supercomputing include Beowulf / GPU / MIC etc
- New startups include Adapteva (developer of Epiphany chip and the Parallella "supercomputer")
  - Used KickStarter to raise $898k in 1 month in 2012
  - Epiphany has 16 cores = 32 GFLOP @ 2W
  - Parallela is complete credit-card size computer with 2 A9 CPU + Epiphany + RAM etc for $99
  - FPGA based ….

# Moore's Law for HPC Cost



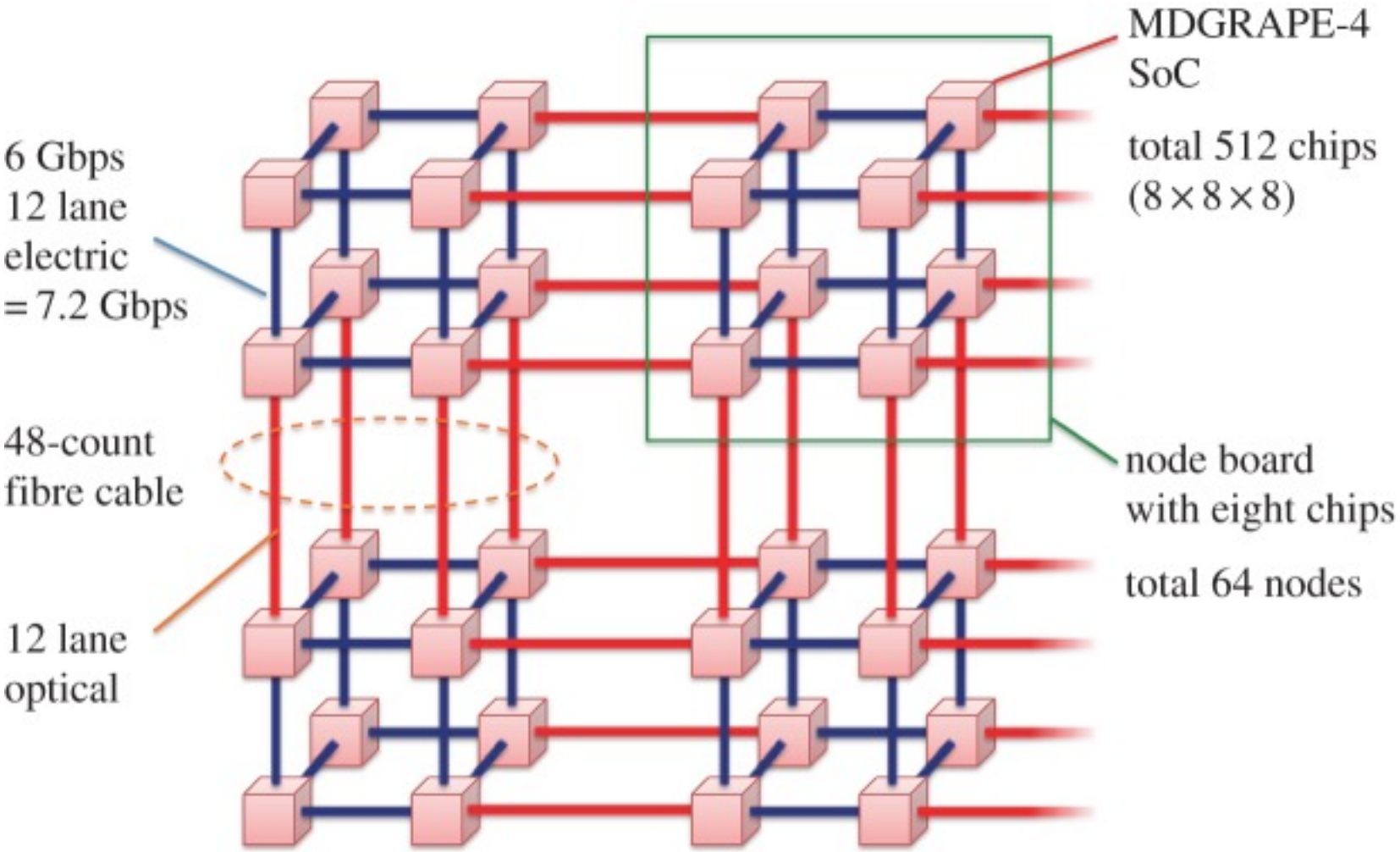
# FPGA?

- Field-Programmable Gate Array
- A chip where the interconnects between logic blocks can be decided by the user ‘in the field’ using HDL
- NOT a general purpose computer but can implement key functions in hardware
- Traditionally much slower and more expensive and lower volumes than ASIC

# MD-GRAPE

- Special purpose computer for protein MD calcs
- Dedicated hardware for particular classes of force calculation etc
- System-on-Chip design: combines dp and sp cores + memory + 3D torus interconnect
- V3 had 1 PFLOP with 4080 CPUs in 2006 and cost $9m c.f. BlueGene/L at same time had 131,072 cores for 0.28 PFLOP and $250m
- V4 being built (now overdue …)

# MDGRAPE-4


MDGRAPE-4
SoC
total 512 chips
(8×8×8)
6 Gbps
12 lane
electric
= 7.2 Gbps
48-count
fibre cable
12 lane
optical
node board
with eight chips
total 64 nodes

# Further Reading

- Grid at http://www.epcc.ed.ac.uk/services/grid-computing/
- National Grid Service at http://www.ngs.ac.uk
- Folding@Home at http://folding.stanford.edu/

- Fortress at http://projectfortress.java.net
- Chapel at http://chapel.cray.com
- X10 at http://x10-lang.org
- Fortran2003 at http://www-users.york.ac.uk/~mijp1/COL/fortran_2003.pdf
- UPC at http://upc.gwu.edu/

- Green500 at http://www.green500.org
- AMD FirePro at http://www.amd.com/Documents/firepro-s9150-datasheet.pdf
- PEZY-SC at http://pezy.co.jp/en/products/pezy-sc.html
- Xeon Phi at http://en.wikipedia.org/wiki/Xeon_Phi
- Parallela at http://www.parallella.org
- MDGRAPE at http://www.ncbi.nlm.nih.gov/pubmed/24982255