The impact of many-core computer architectures on numerical libraries: past, present and future

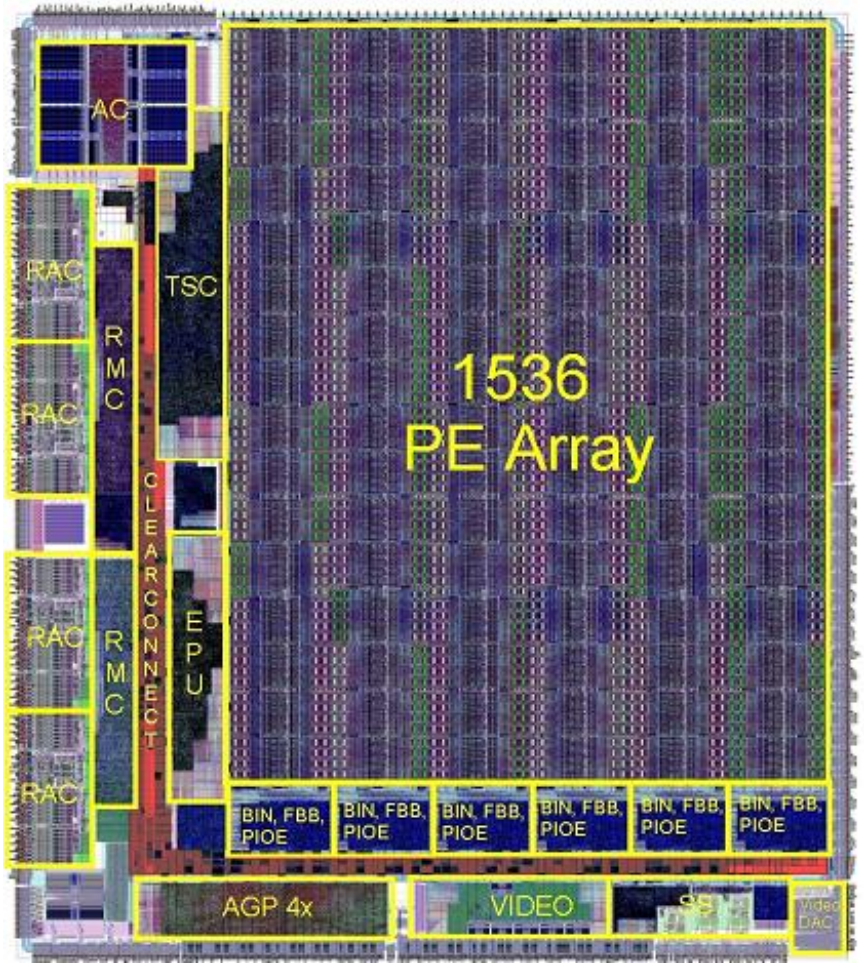
Simon McIntosh-Smith simonm@cs.bris.ac.uk
Head of Microelectronics Research

University of BRISTOL

A brief biography

- Graduated as Valedictorian in Computer Science from Cardiff University in 1991
- Joined Inmos to work for David May as a microprocessor architect
- Moved to Pixelfusion in 1999 – a high-tech start-up designing the first many-core general purpose graphics processor (GPGPU)
- Co-founded ClearSpeed in 2002 as Director of Architecture and Applications
- Joined the CS department at the University of Bristol in April 2009 to focus on High Performance Computing and architectures

Processor CV: Many-core GPUs



Pixelfusion F150: (2000)

- 0.25u embedded DRAM
- 76M transistors
- 3 MBytes eDRAM

Multi Threaded Array Processor

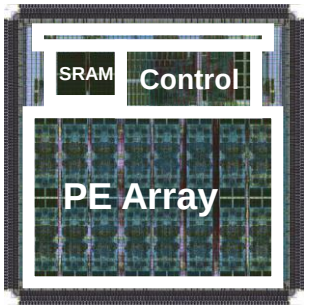
- 1,536 PEs + redundancy
- 4 parallel RAMBUS channels, 6.4 GBytes/s

The first true GPGPU

- Fully programmable

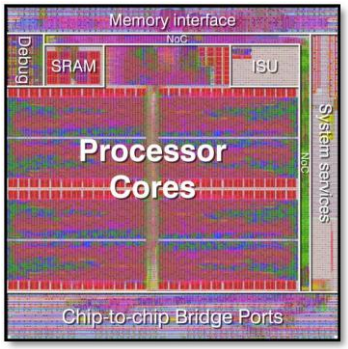
University of BRISTOL

Many-core HPC processors



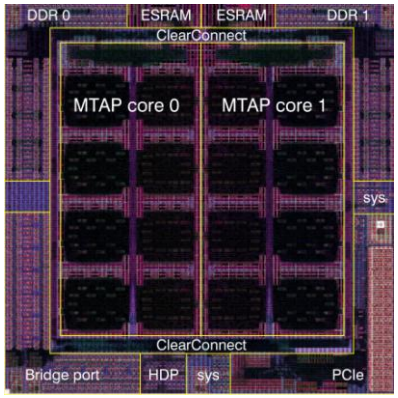
ClearSpeed CS301 (2004)

- **25 GFLOPS** (32-bit), **3W** @ 200MHz
- **64 PEs**, 4 KBytes SRAM each
- IBM 130nm, 41 million transistors



ClearSpeed CSX600 (2006)

- **40 GFLOPS** (64-bit), **12W** @ 210 MHz
- **96 PEs**, 6 KBytes SRAM each
- Integrated DDR2-ECC
- IBM 130nm, 128 million transistors



ClearSpeed CSX700 (2008)

- **96 GFLOPS** (64-bit), **10W** @ 250MHz
- Fully 64-bit architecture
- **192** PEs (2x96)
- 2x ECC DDR2 controllers
- IBM 90nm, 256 million transistors

First principles

What are the issues driving the development of numerical libraries?

Underlying hardware changes

The real Moore's Law

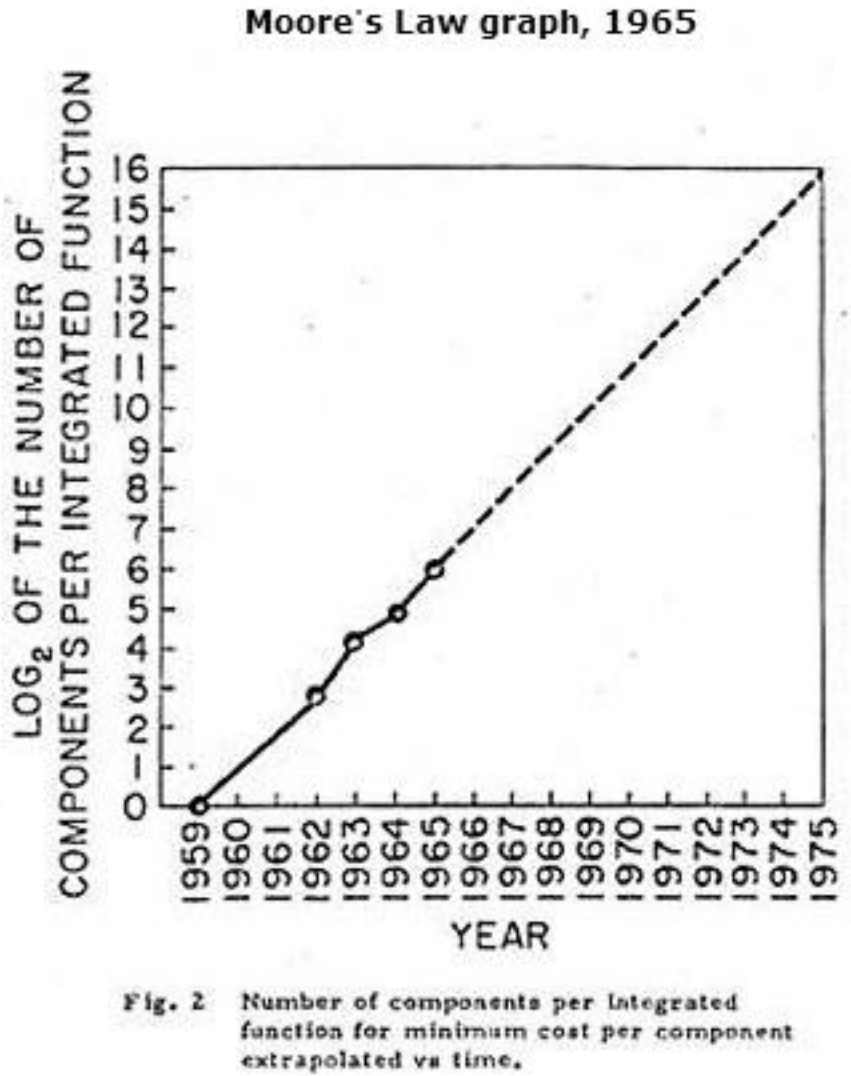
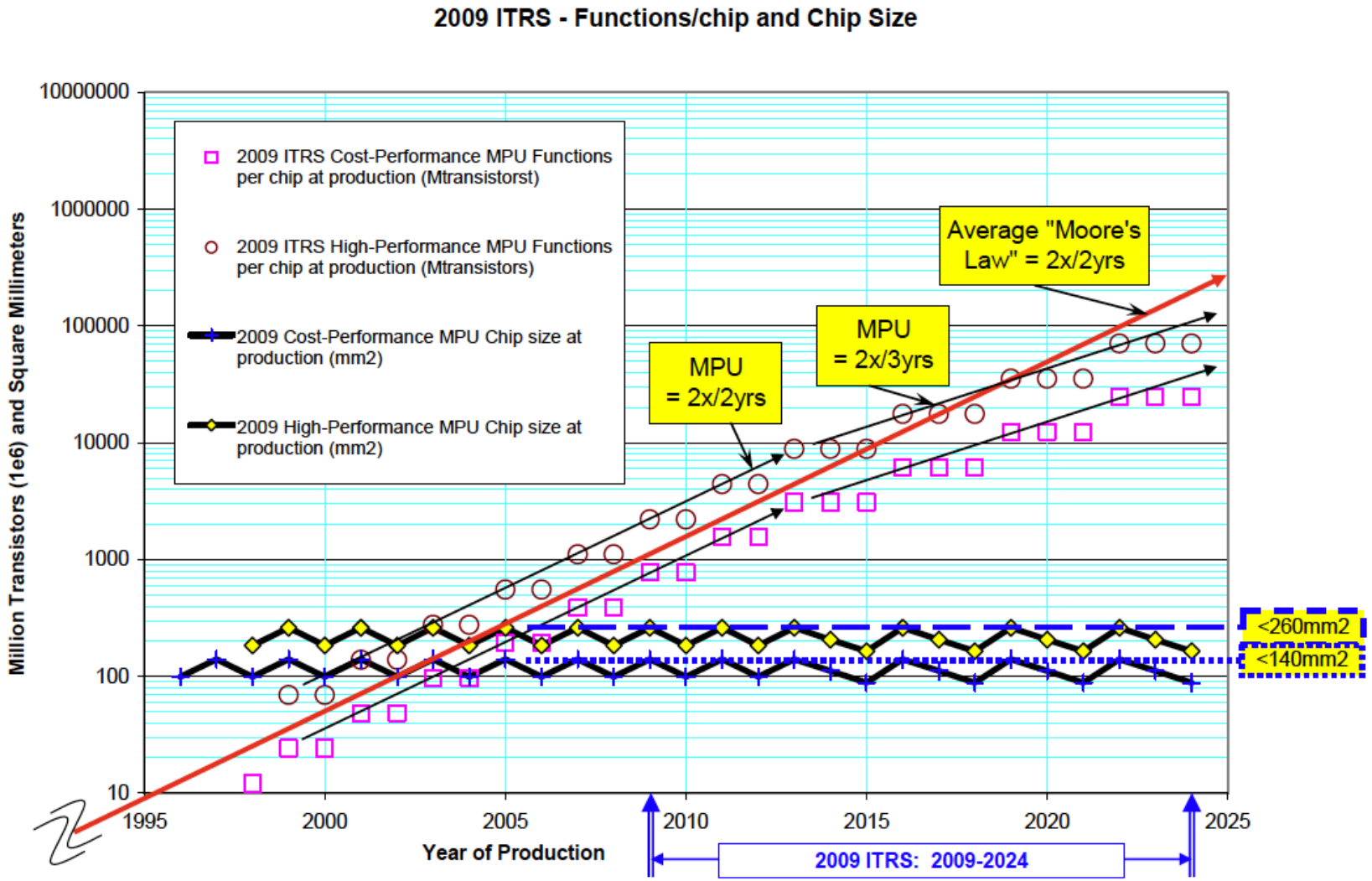


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

45 years ago, Gordon Moore observed that the number of transistors on a single chip was doubling rapidly

http://www.intel.com/technology/mooreslaw/

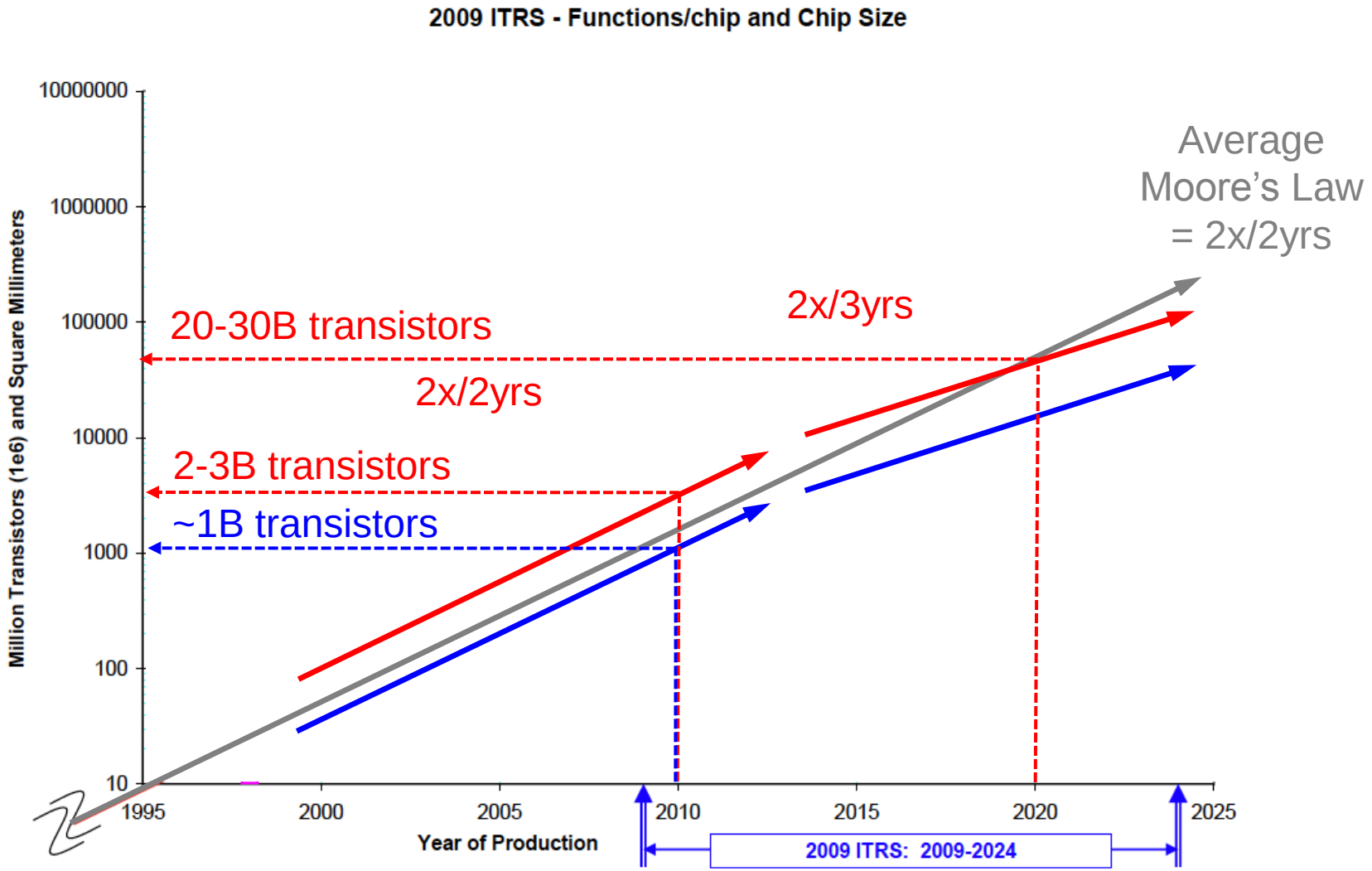
Moore's Law today



University of BRISTOL

http://www.itrs.net/Links/2009ITRS/2009Chapters_2009Tables/2009_ExecSum.pdf

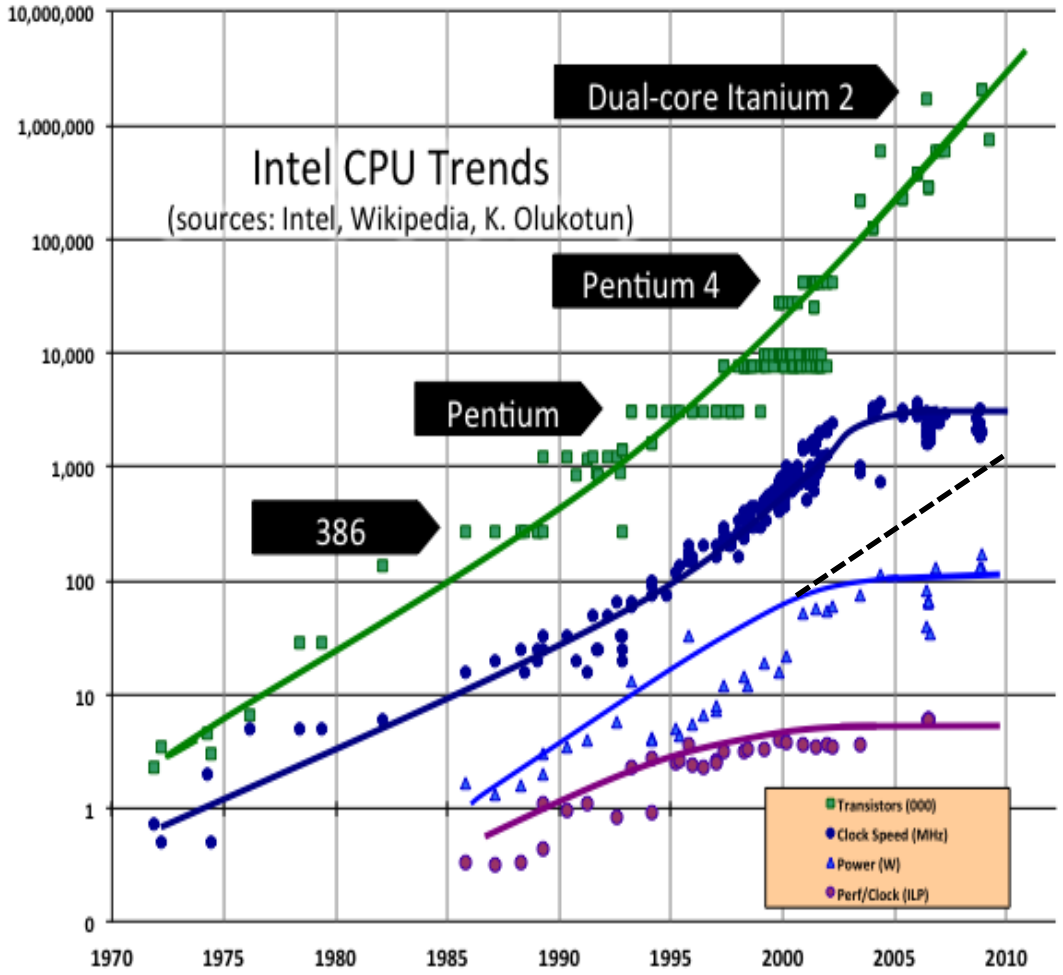
Moore's Law today



University of BRISTOL

http://www.itrs.net/Links/2009ITRS/2009Chapters_2009Tables/2009_ExecSum.pdf

Important technology trends



The real Moore's Law

The clock speed plateau

The power ceiling

Instruction level parallelism limit

University of BRISTOL

Herb Sutter, "The free lunch is over", Dr. Dobb's Journal, 30(3), March 2005. On-line version, August 2009.
http://www.gotw.ca/publications/concurrency-ddj.htm

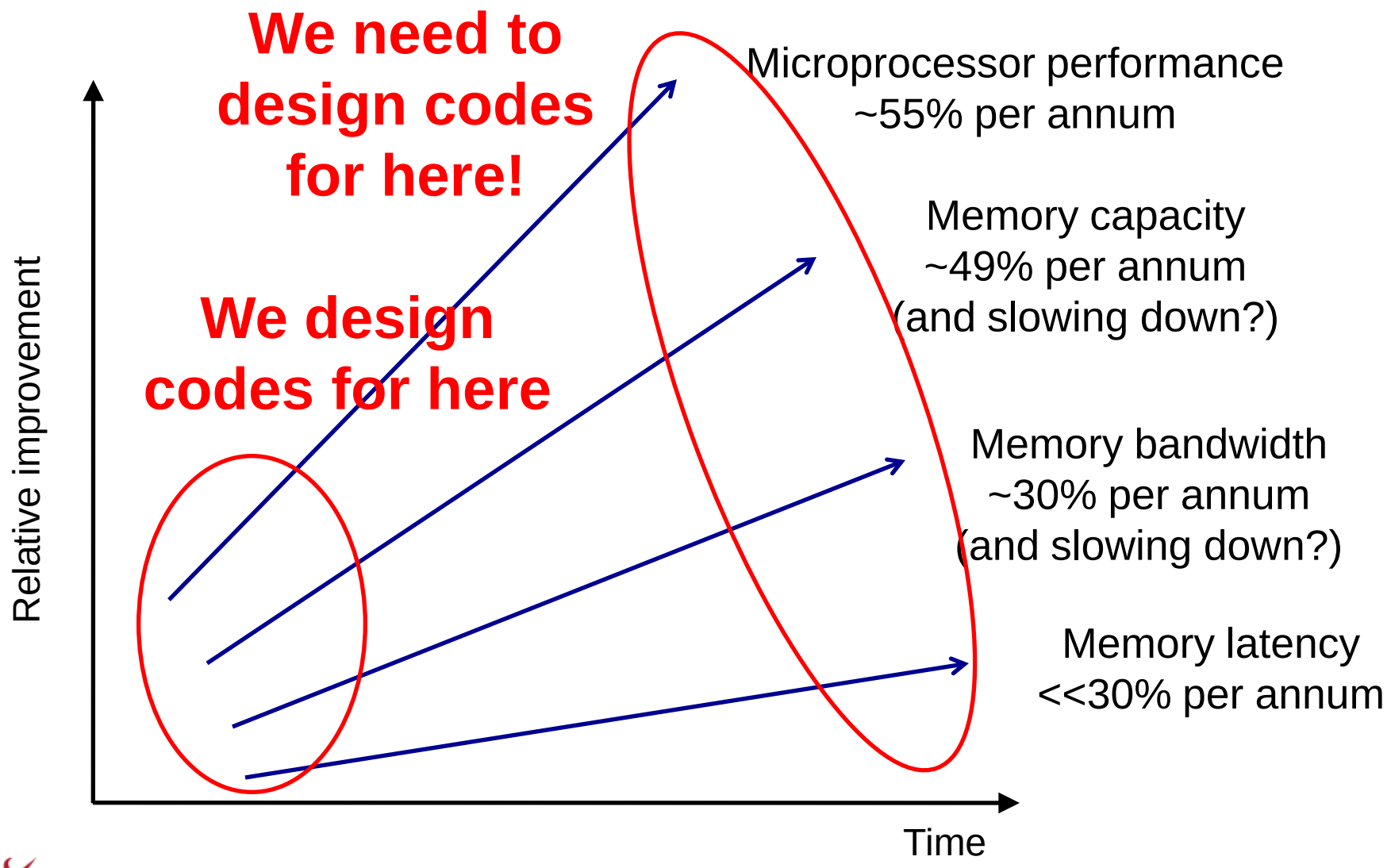
How best to use billions of transistors?

- Lots more cores on-chip (***doubling every 2 years***)
  - Core designs will stay roughly the same

- Power consumption must be held in check
  - Chip voltages can't be dialled down any more
  - ➢ Clock speeds may ***decrease***
  - ➢ Memory bandwidth per core likely to ***decrease***
  - ➢ Memory per core likely to ***decrease***

- Different types of core
  - ***Heterogeneous computing***
  - E.g. a few heavyweight (x86) cores together with many more lightweight (GPU) cores

Relative hardware trends



University of BRISTOL

Heterogeneous computing is not new

- Most systems are ***already*** heterogeneous
  - PCs have CPU, GPU, network processor, I/O processor, …
  - Has been a common approach in embedded systems since the early `90s

- But now heterogeneous systems are starting to include several ***different*** types of ***general-purpose, programmable*** processors
  - Users have to programme more than one type of processor to get the most out of a system

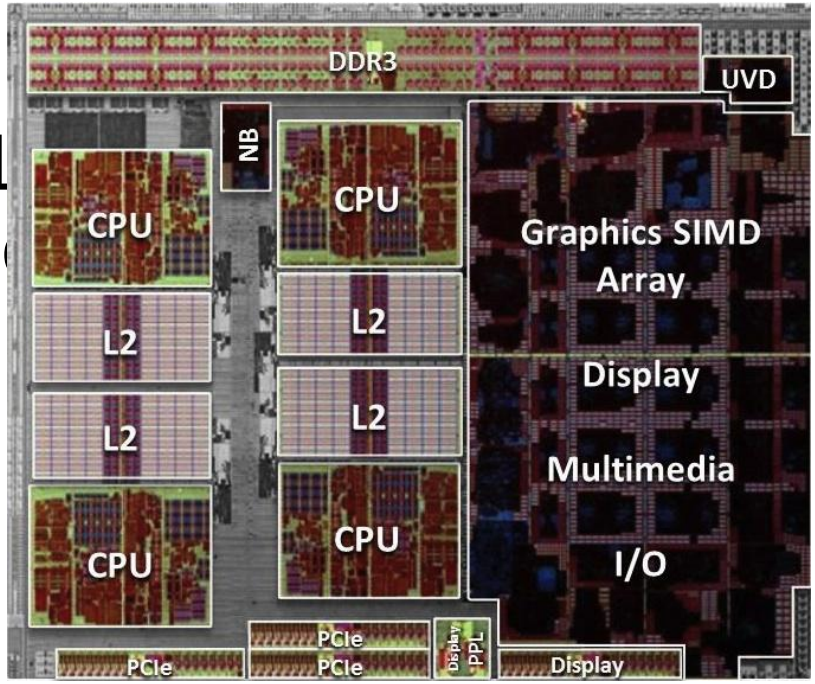
5 core tablet at CES last week

University of BRISTOL

Trends in processors

AMD's first "Fusion" chip, shipping since late 2011

- Integrates a quad core x86 CPU with an OpenCL programmable GPU in the same chip
- Also Intel (Ivy Bridge), Nvidia (Tegra, Denver), IBM (Cell), …



Emerging standards

- OpenCL, OpenACC, DirectCompute, C++ AMP, …

Heterogeneous systems in the Top500

- Tokyo Tech's TSUBAME was first in 2006
  - Started with ClearSpeed, now using GPUs
- Now several systems in existence, more on their way:
  - #2 Tianhe-1A (China), 2.57 PFLOPS, Intel and NVIDIA
  - #4 Dawning (China), 1.27 PFLOPS, Intel and NVIDIA
  - #5 Tsubame 2 (Japan), 1.19 PFLOPS, Intel x86 and NVIDIA
  - #10 RoadRunner (USA), 1.04 PFLOPS, IBM Cell, AMD x86
  - Around 35 GPU-based systems in Top500 in Nov 2011
- Most of the >10 PFLOP systems using many-core processors (GPUs or Intel's MIC) – Titan (ORNL), Stampede (TACC), Blue Waters (UIUC/NCSA), …

http://www.top500.org

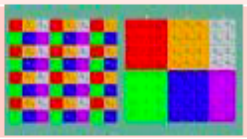
Parallel numerical libraries:

Past, present and future

University of BRISTOL

A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels

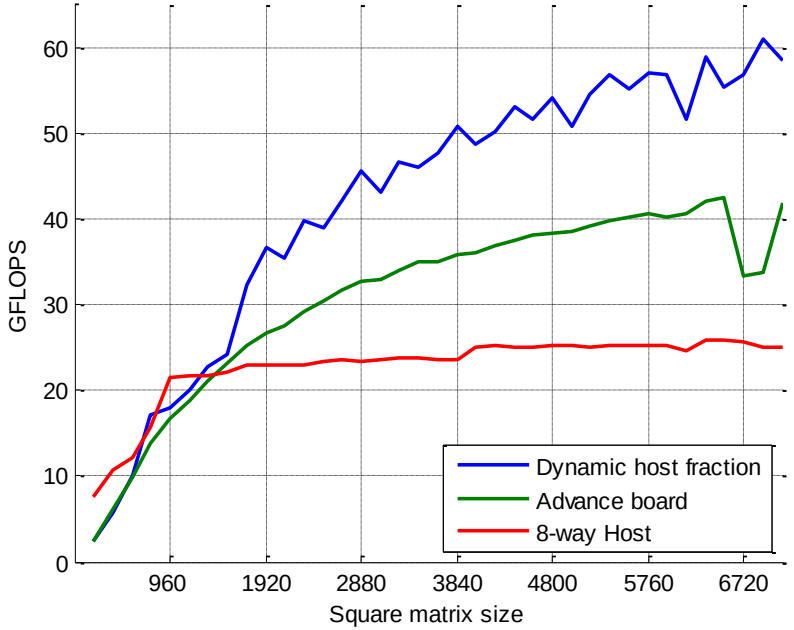
Those new algorithms

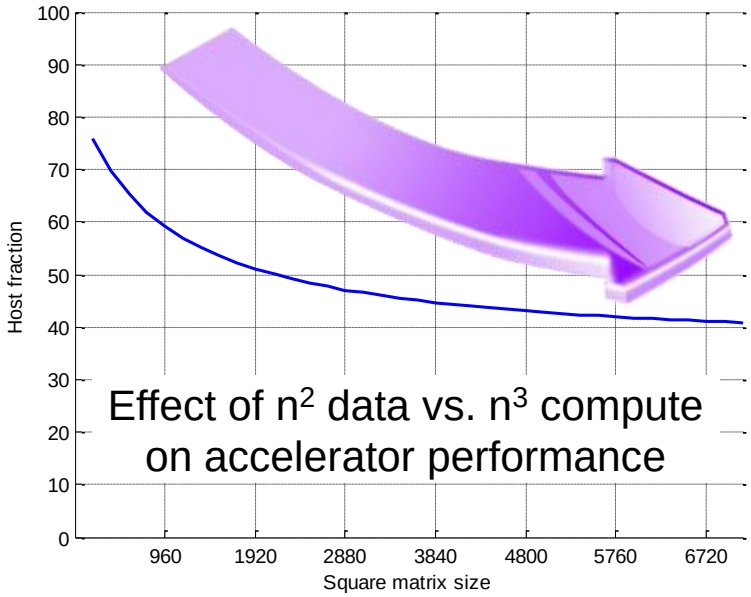
- have a very low granularity, they scale very well (multicore, petascale computing, …)
- removes a lots of dependencies among the tasks, (multicore, distributed computing)
- avoid latency (distributed computing, out-of-core)
- rely on fast kernels

Those new algorithms need new kernels and rely on efficient scheduling algorithms.

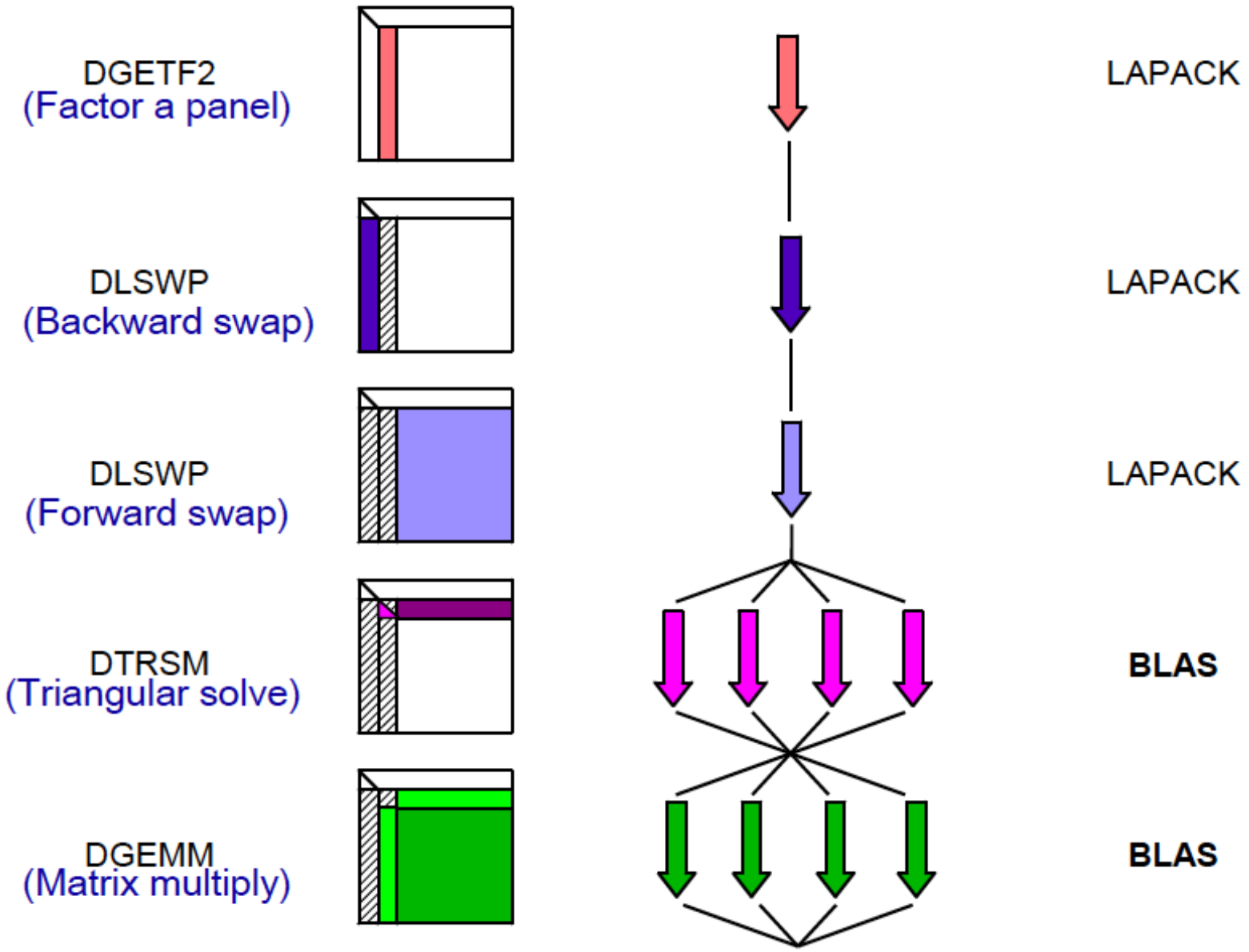
ClearSpeed's CSXL BLAS/LAPACK

- CSXL was a BLAS/LAPACK library that used run-time heuristics to load balance across heterogeneous compute resources
- Transparently harnessed multiple host CPU cores and multiple accelerators ***simultaneously***
- Could also handle datasets larger than the memories of the accelerators
- S. McIntosh-Smith, J. Irwin, "Delivering aggregated performance for high-performance math libraries in accelerated systems", International SuperComputing, Dresden, June 2007



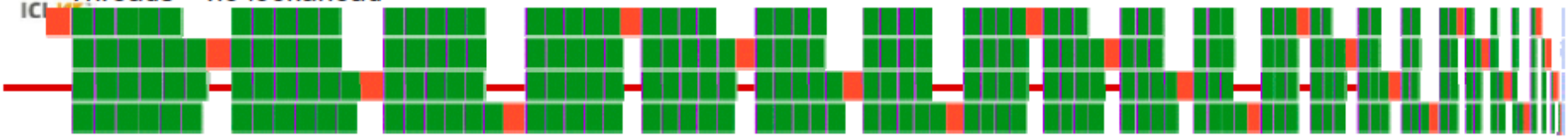


Steps in the LAPACK LU

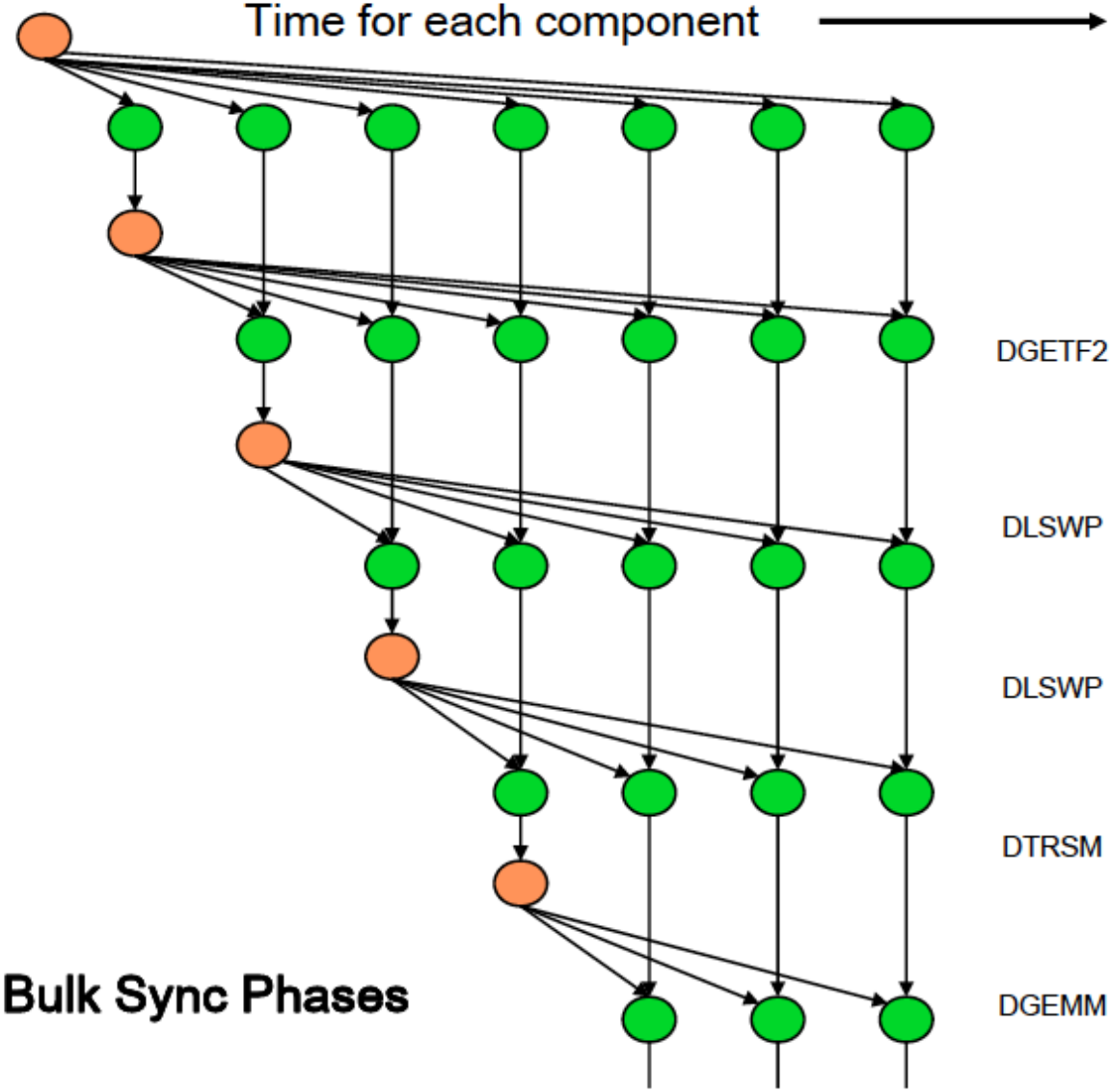


LU Timing Profile (4 processor system)

Threads – no lookahead



Time for each component



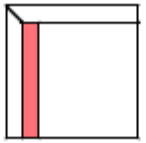
DGETF2

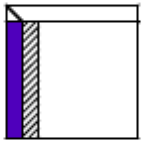
DLASWP(L)

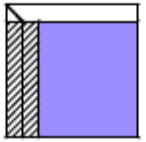
DLASWP(R)

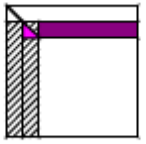
DTRSM

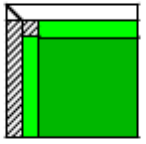
DGEMM

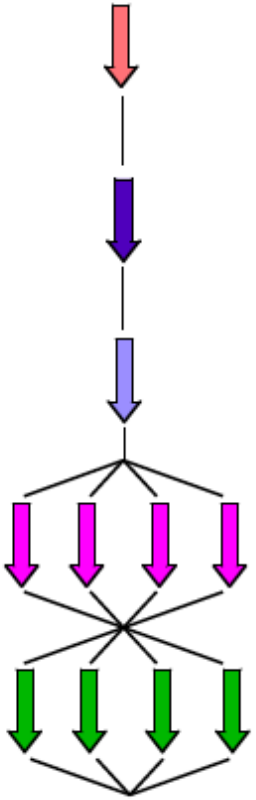






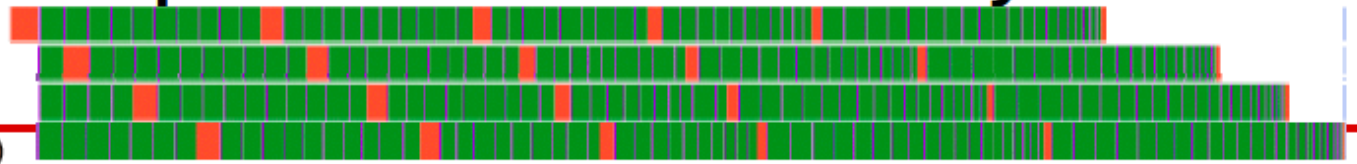


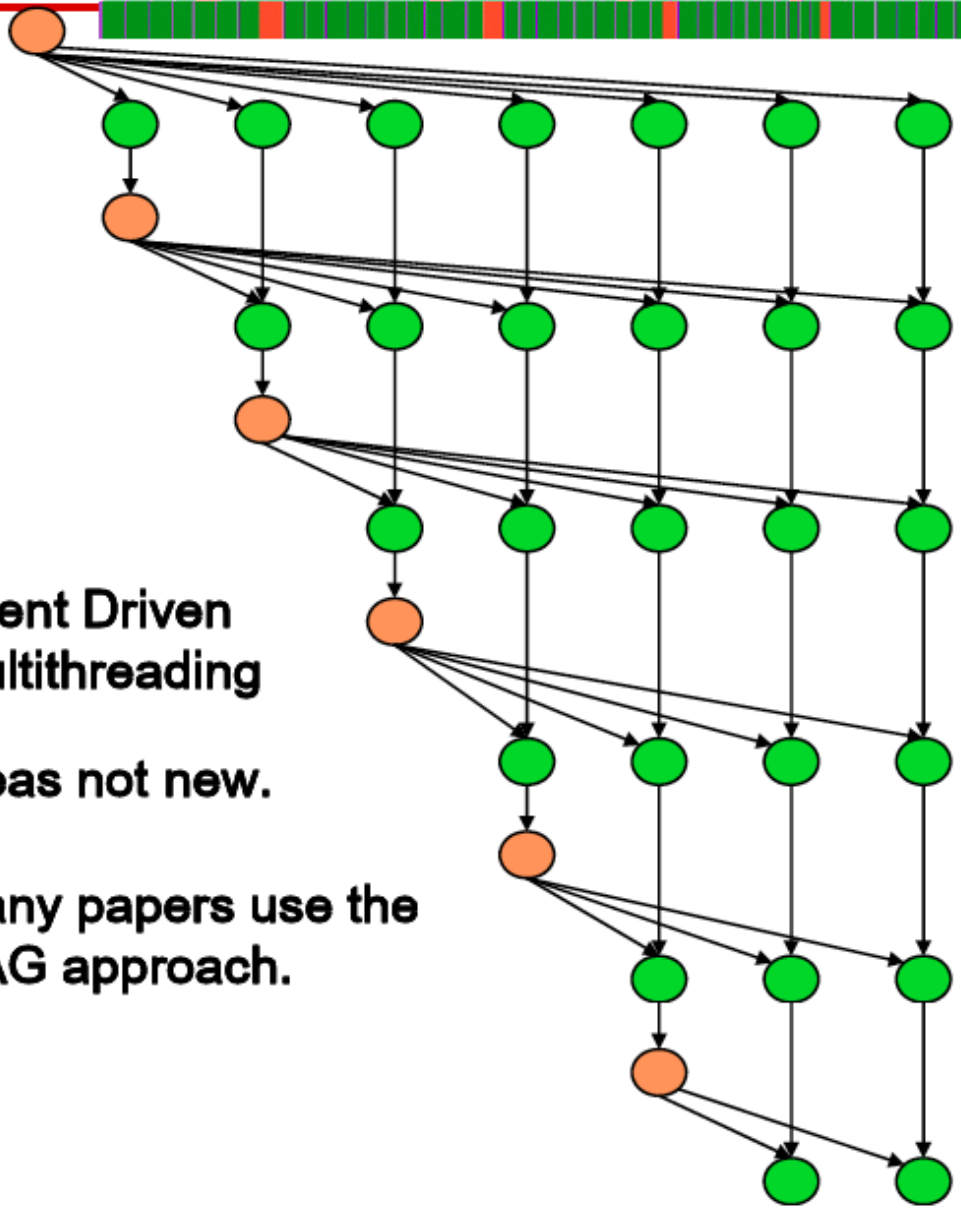




Bulk Sync Phases

Adaptive Lookahead - Dynamic





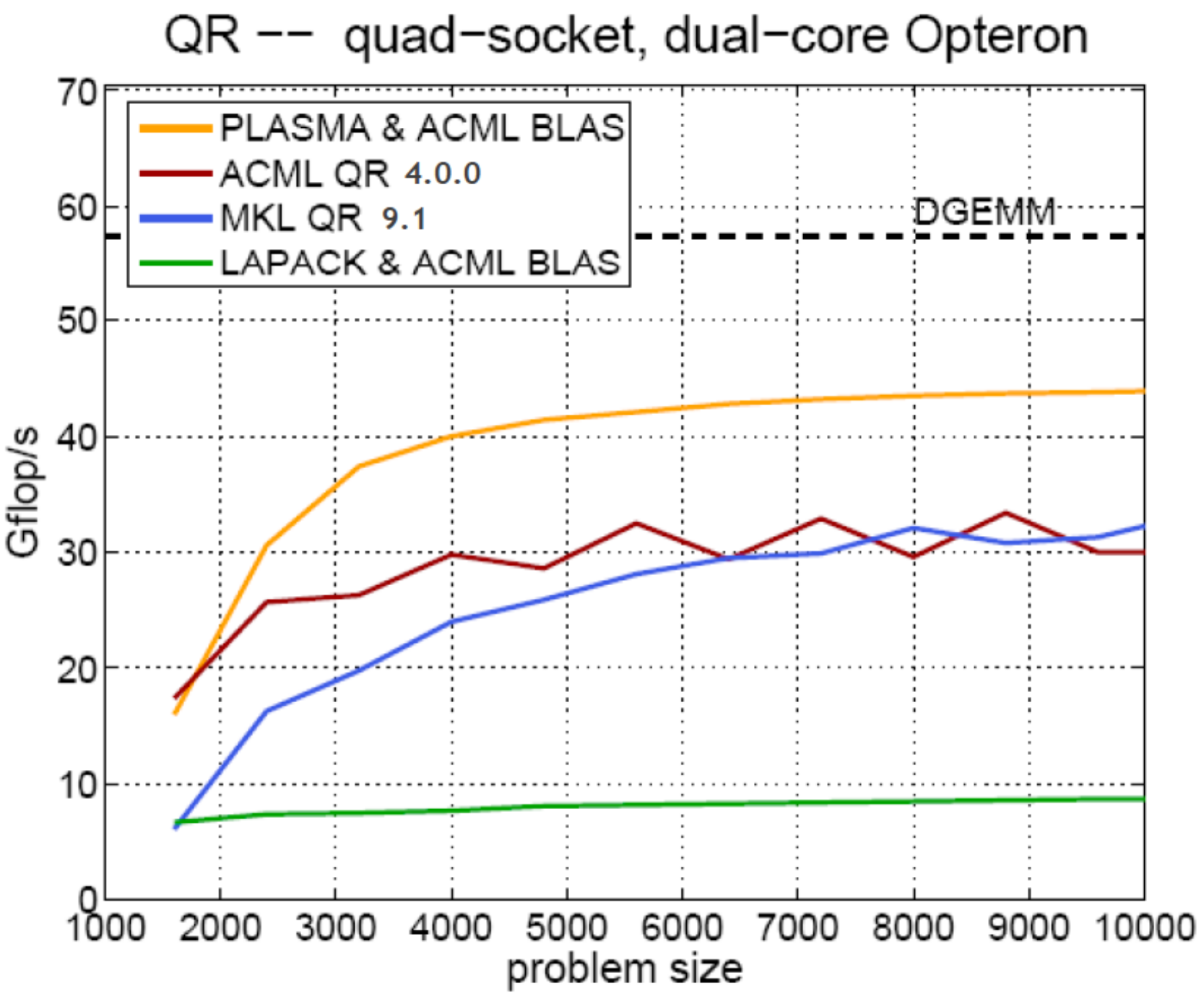
Event Driven Multithreading

Ideas not new.

Many papers use the DAG approach.

```
while(1)
    fetch_task();
    switch(task.type) {
        case PANEL:
            dgetf2();
            update_progress();
        case COLUMN:
            dlaswp();
            dtrsm();
            dgemm();
            update_progress();
        case END:
            for()
                dlaswp();
            return;
    }
}
```

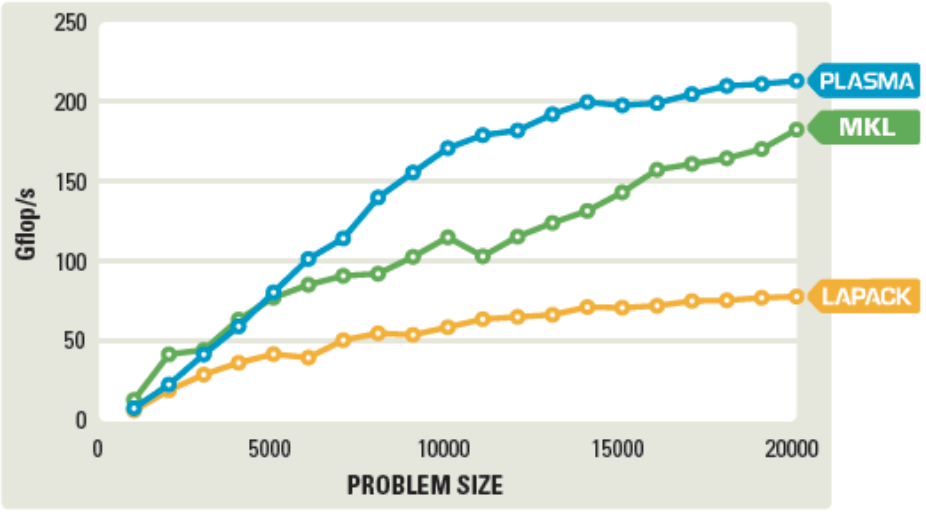
Reorganizing algorithms to use this approach


QR -- quad-socket, dual-core Opteron
PLASMA & ACML BLAS
ACML QR 4.0.0
MKL QR 9.1
LAPACK & ACML BLAS
DGEMM
Gflop/s
problem size
0
10
20
30
40
50
60
70
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

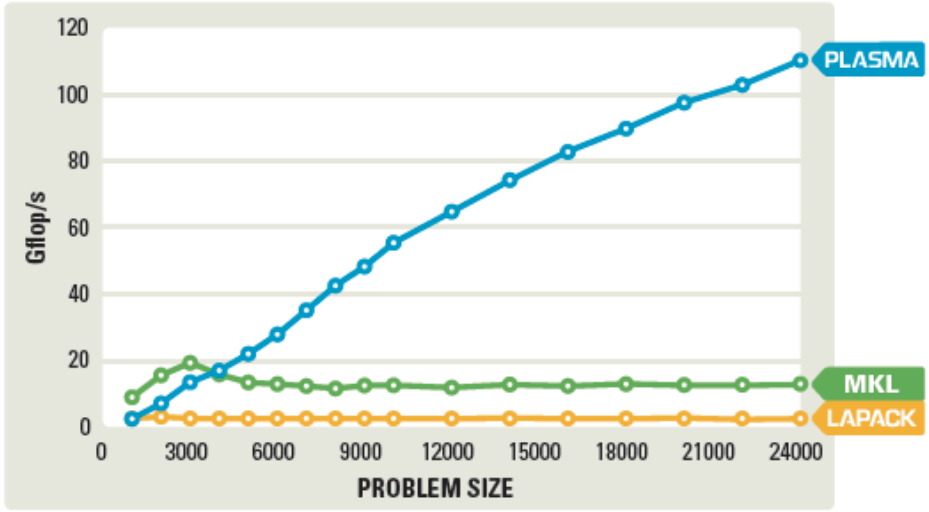
PLASMA coverage

FUNCTIONALITY	COVERAGE
Linear Systems of Equations	Cholesky, LDLT, LU with partial pivoting
Matrix Inversion	Cholesky, LU with partial pivoting
Least Squares	QR and LQ
Mixed Precision Iterative Refinement	linear systems using Cholesky or LU, least squares using QR or LQ
Symmetric Eigenvalue Problem	eigenvalues only
Singular Value Problem	singular values only
Level 3 Tile BLAS	GEMM, HEMM, HER2K, HERK, SYMM, SYR2K, SYRK, TRMM, TRSM
In-Place Layout Translation	CM, RM, CCRB, CRRB, RCRB, RRRB

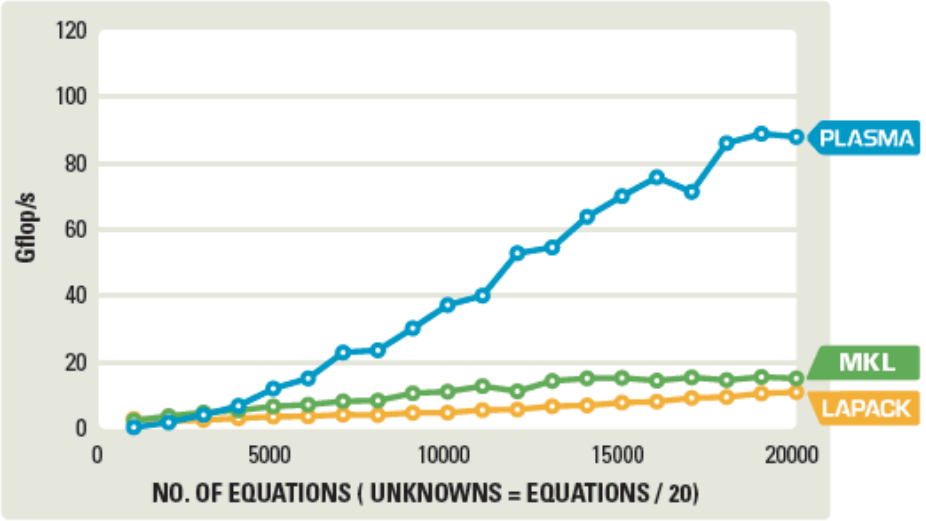
Solving Linear System (DGESV)
48-core, 2.1 GHz AMD Magny-Cours System



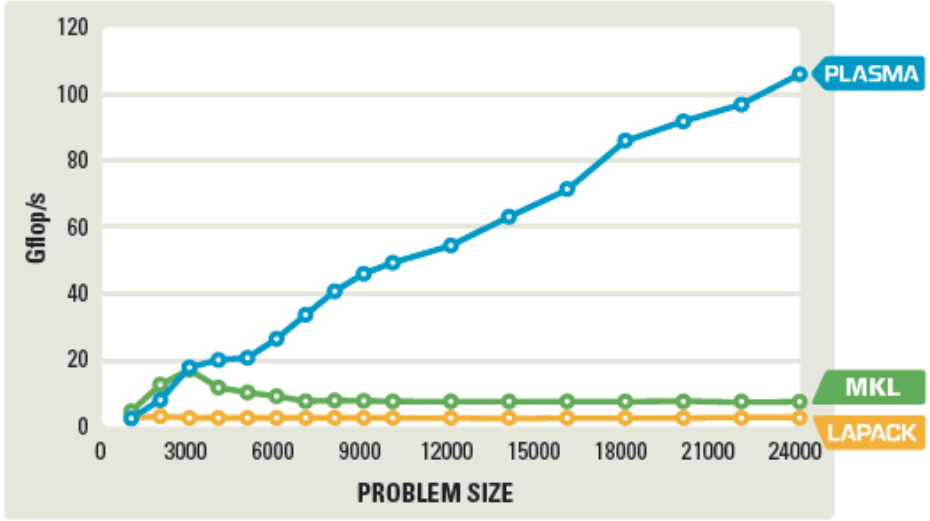
Solving Symmetric EVP (DSYEV)
48-core, 2.1 GHz AMD Magny-Cours System



Solving Least Squares Problem (DGELS)
48-core, 2.1 GHz AMD Magny-Cours System



Solving Singular Value Problem (DGESVD)
48-core, 2.1 GHz AMD Magny-Cours System



University of BRISTOL

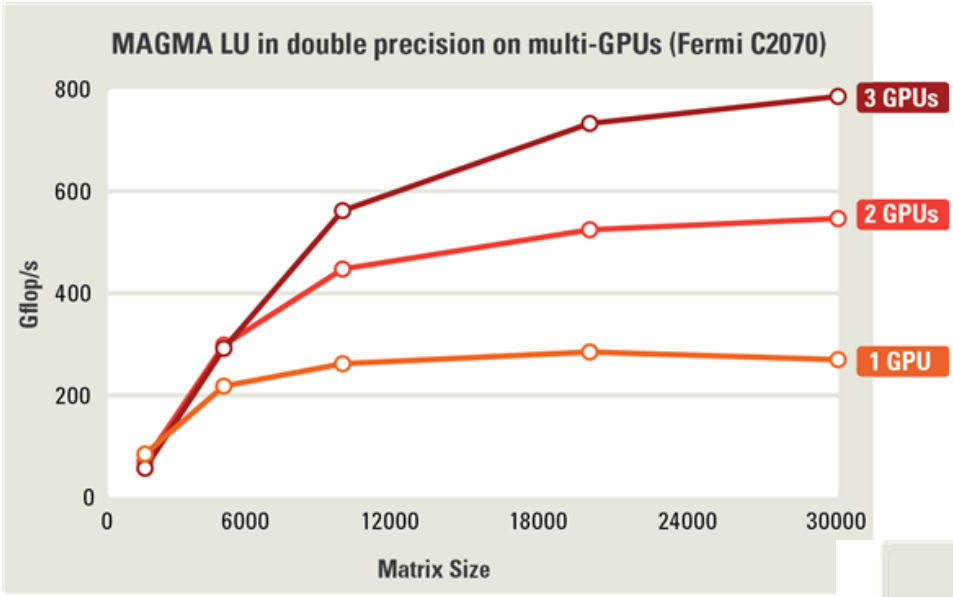
MAGMA

- Extends PLASMA to support heterogeneous systems (GPUs et al)
- Host of extra considerations:
  - Where does the data live?
  - Data formats? (Natural, blocked, …)
  - Multiple accelerators
  - Streaming?

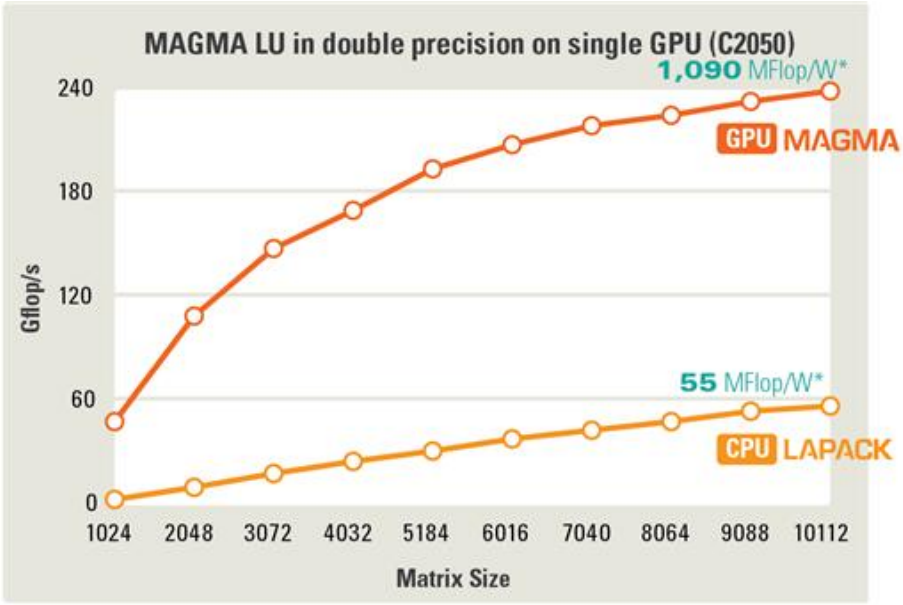
MAGMA 1.1 coverage

MAGMA 1.1 ROUTINES & FUNCTIONALITIES	SINGLE GPU	MULTI-GPU STATIC	MULTI-GPU DYNAMIC
One-sided Factorizations (LU, QR, Cholesky)	✓	✓	✓
Linear System Solvers	✓		✓
Linear Least Squares (LLS) Solvers	✓		✓
Matrix Inversion	✓		✓
Singular Value Problem (SVP)	✓		
Non-symmetric Eigenvalue Problem	✓		
Symmetric Eigenvalue Problem	✓		
Generalized Symmetric Eigenvalue Problem	✓		

SINGLE GPU	Hybrid LAPACK algorithms with static scheduling and LAPACK data layout
MULTI-GPU STATIC	Hybrid LAPACK algorithms with 1D block cyclic static scheduling and LAPACK data layout
MULTI-GPU DYNAMIC	Tile algorithms with StarPU scheduling and tile matrix layout



Keeneland system, using one node
3 NVIDIA GPUs (M2070 @ 1.1 GHz, 5.4 GB)
2 x 6 Intel Cores (X5660 @ 2.8 GHz, 23 GB)



University of BRISTOL

GPU Fermi C2050 (448 CUDA Cores @ 1.15 GHz)
+ Intel Q9300 (4 cores @ 2.50 GHz)
DP peak **515** + **40** GFlop/s
Power * ~220 W

CPU AMD Istanbul
[8 sockets x 6 cores (48 cores) @2.8GHz]
DP peak **538** GFlop/s
Power * ~1,022 W

* Computation consumed power rate (total system rate minus idle rate), measured with KILL A WATT PS, Model P430

Big Issue:

Composibility of Parallelism

University of BRISTOL

“Who owns the parallelism?”

- Multiple levels in the software stack:
  - Operating system / run-time
  - Libraries
  - Application

- Who decides what runs where?
- Who owns the resources?

Composibility

Consider the following example using a modern dual socket, multi-core server (12 to 16 cores today):

- Your application is written in OpenMP or MPI in order to use all these cores
- Then you want to call a parallel version of a numerical library, such as BLAS, LAPACK etc.
- Essentially have to “pass over” ownership of the hardware resources from the application to the library
- This problem gets worse as the width and depth of the parallelism increase – GPUs with OpenCL etc

Composibility continued

More issues:

- What if you want varying widths of parallelism? (Elastic widths)
- What effect do multiple users have on the available parallelism? Don't know how much you have until execution time…

More future issues for NA libs

From Dongarra et al, SIAM PP08:

- Dynamic Data Driven Execution
- Self Adapting
- Mixed Precision in the Algorithm
- Exploit Hybrid/Many-core Architectures
- Fault Tolerant Methods
- Communication Avoidance

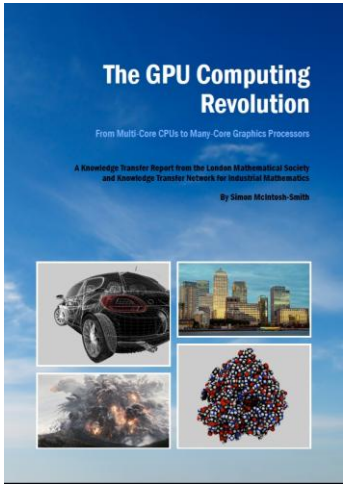
Summary and Conclusions

- Future hardware will see considerable increases in:
  - Width of parallelism (cores, vectors, …)
  - Depth of parallelism (heavyweight, lightweight, threads, SIMD, …)
  - Depth and complexity of memory hierarchy
  - Heterogeneity
- Core counts will increase faster than bandwidth, memory capacity and latency
- Future numerical libraries will need to adapt at run-time to exploit available resources
- Thus the very nature of software libraries will fundamentally change (ship as source?)
- Major unresolved issue around parallel composibility

For an introduction to GPUs

The GPU Computing Revolution – a Knowledge Transfer Report from the London Mathematical Society and the KTN for Industrial Mathematics

- https://ktn.innovateuk.org/web/mathsktn/articles/-/blogs/the-gpu-computing-revolution



ASEArch CCP

- New CCP just formed to help in this area:
  - Algorithms and Software for Emerging Architectures – ASEArch
  - Collaboration between Oxford, STFC, Bristol and Edinburgh
  - http://www.oerc.ox.ac.uk/research/asearch