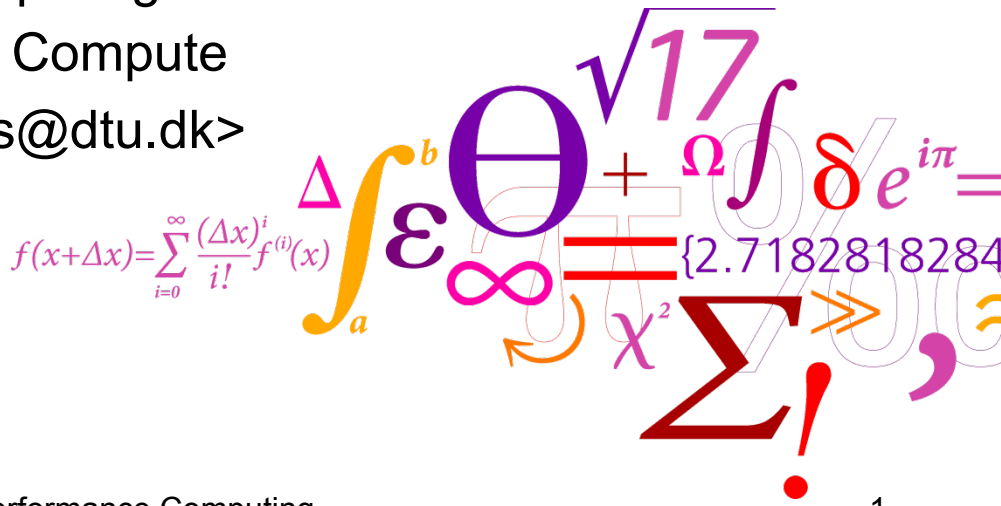
Introduction to GPU Computing

Hans Henrik Brandenborg Sørensen
DTU Computing Center
DTU Compute
<hhbs@dtu.dk>



Practicalities

- Teacher(s) - Week 3
  - Hans Henrik B. Sørensen
  - Andreas / Andrea
- Lectures / GPU Exercises – Mon-Tue
  - Learning by doing!
  - Complementary to Assignment 3 (should not be handed in)
  - Collaborate!
- Assignment #3: GPU Matrix Multiplication and GPU Poisson Problem – Wed-Fri
  - Collaborate on developing the code.
  - Write individual reports with analysis, discussions and results.

Learning objectives for week 3

- This week we focus on learning
  - Interplay of computer components like CPU, GPU, caches, and memory
  - Choose the optimal hardware platform for a given problem
  - Write parallel programs with OpenMP and CUDA
  - Write efficient programs for multi-core processors and many-core GPUs / Multi-GPUs
- Pre-requisites: access to a CUDA enabled GPU (e.g. DCC or home-pc with Nvidia cc.>=3.0).

Overview for week 3

- GPU computing – Mon
  - Introduction to GPU computing
  - CUDA programming model
  - CUDA memory model
- Performance tuning of CUDA applications – Tue
  - Basic tuning techniques
  - Memory access optimization
  - Control flow and instruction optimizations
  - Profiling tools / demo
- Assignment 3 – Wed

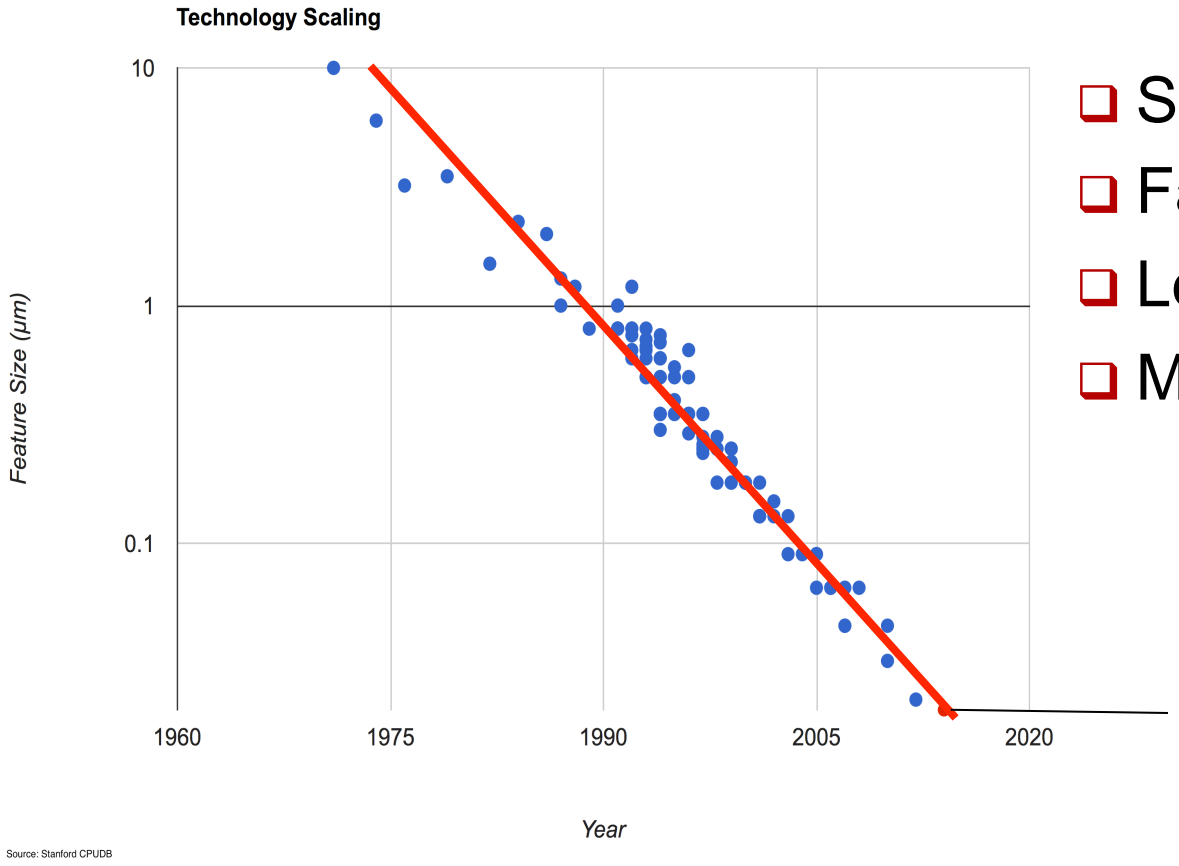
Introduction to GPU computing

Outline

- Why GPUs are important in HPC?
- SIMD units
- SIMT execution model
- Trends in GPU computing
- Motivation slides

Recap from week 1: Good news

- Transistor size over the years; still decreases

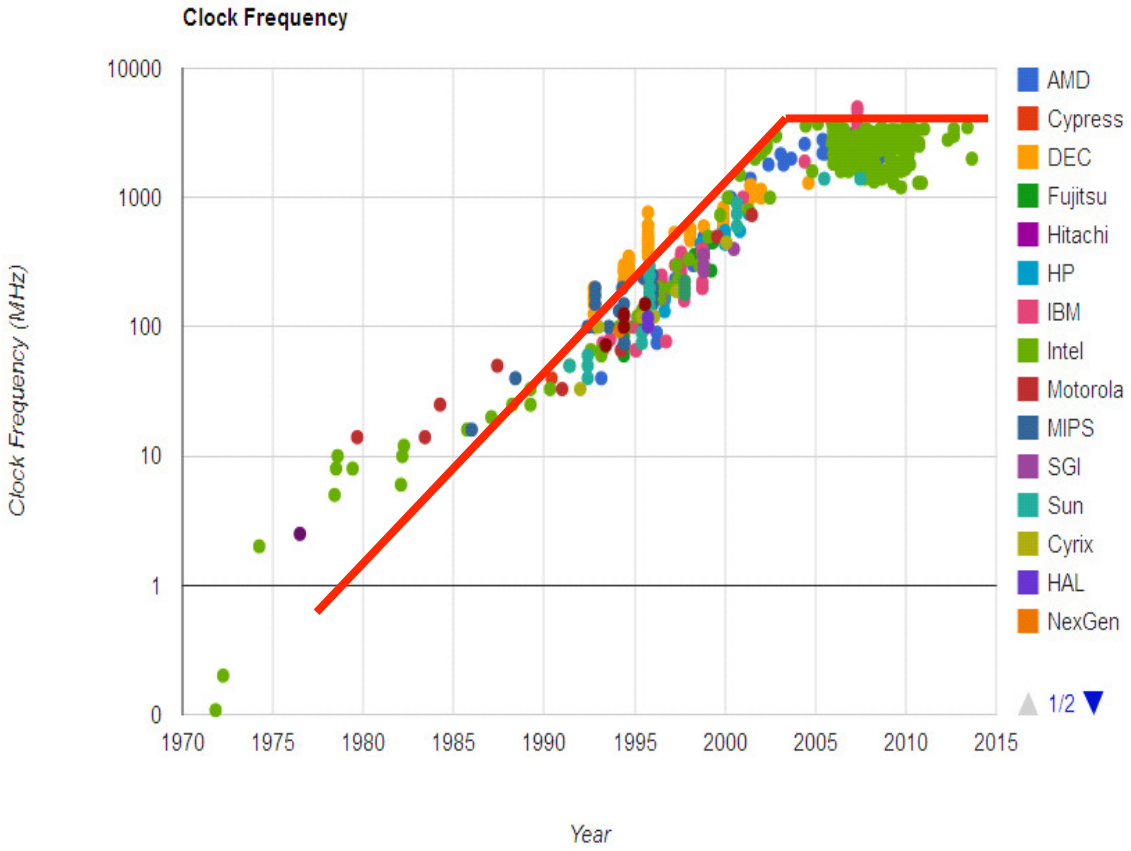


- Smaller
- Faster
- Less power
- More on chip

Currently best: 14 nm

Recap from week 1: Bad news

- Clock speed over the years; stagnated since 2005



- Increasing over many years
- However, over the last decade the clocks speeds have essentially remained constant

Why not increase clock speed?

Why not increase clock speed?

- In order to increase the clock speed we need to decrease the power consumption, otherwise ...

  - “Nard scaling”
    - Reduce transistor voltage proportional to higher clock speed
    - Broke down 2005 because of weakened current in the wires

Why not increase clock speed?

- In order to increase the clock speed we need to decrease the power consumption, otherwise ...

  - "Nard scaling"
    - Reduce transistor voltage proportional to higher clock speed
    - Broke down 2005 because of weakened current in the wires
- What matters today in chips: power consumption!
  - Trade-off favors smaller more efficient processors,
  - rather than faster less efficient processors

Building an efficient processor

- What is the goal?

Building an efficient processor

- What is the goal? Roughly two choices...

A.	B.
Latency (time to complete a task) [e.g. seconds]	Throughput (tasks completed per unit time) [e.g. jobs/hour]

Building an efficient processor

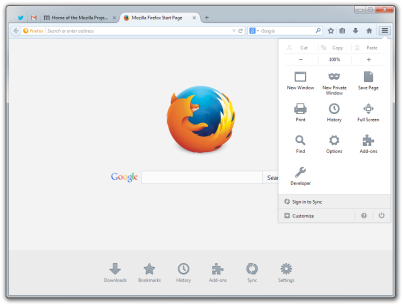
- What is the goal? Roughly two choices...

A.	B.
Latency (time to complete a task) [e.g. seconds]	Throughput (tasks completed per unit time) [e.g. jobs/hour]

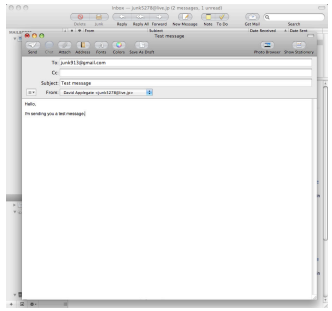
- Unfortunately these goals are not always aligned

CPUs target latency (traditionally)

- Usual task of traditional CPUs
  - Desktop applications
    - Lightly threaded
    - Lots of branches
    - Lots of (indirect) memory accesses



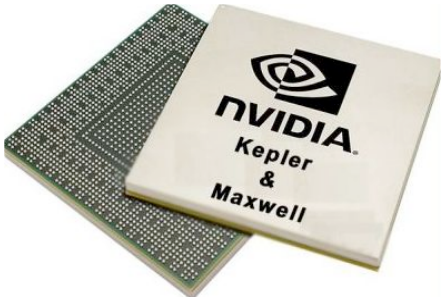
Microsoft® Office™



  - Modern branch predictors > 90% accuracy
- CPUs try to minimize the time to complete a particular task – often to support user interaction!

GPUs target throughput

- They are designed to compute pixels – fast!
  - Rendering video games in real-time
  - Play HD movies on iPhone
  - Render visual effects for movies…



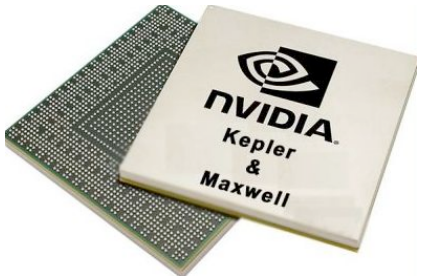
- More concerned about the number of pixels per second than the latency of any particular pixel!

CPUs vs. GPUs

- Complex control hardware
  - + Flexibility in performance
  - − Expensive in terms of power

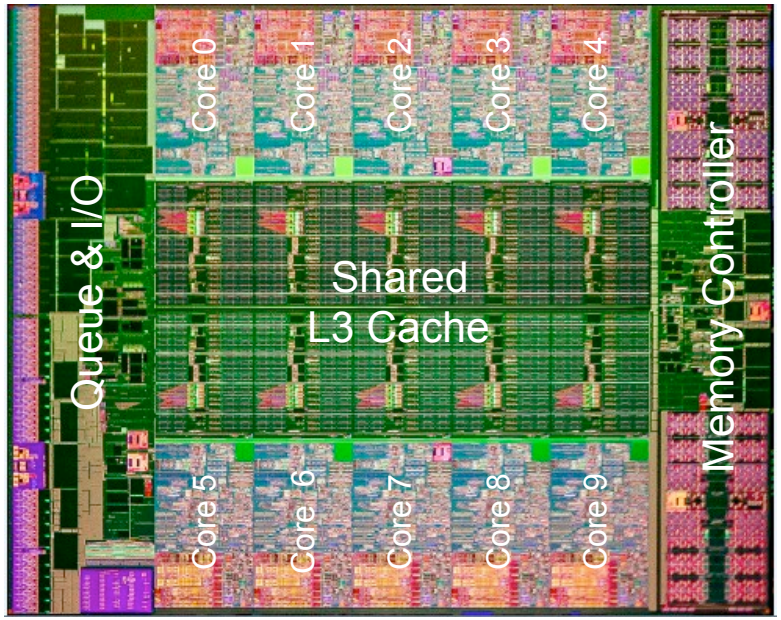
CPUs vs. GPUs

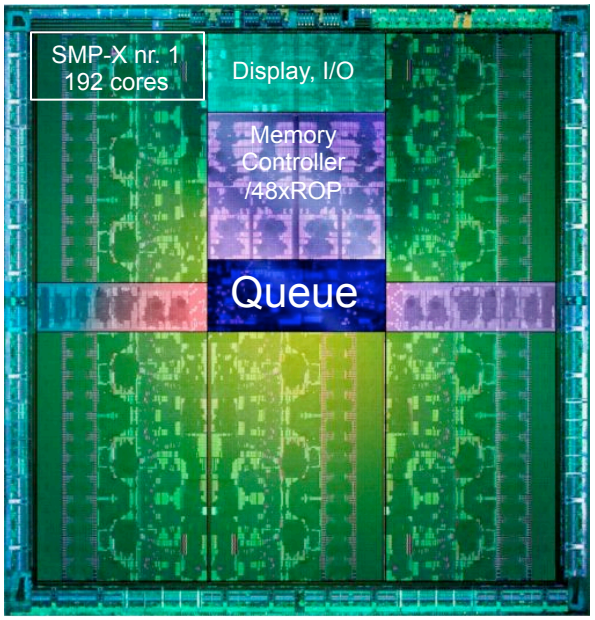
- Complex control hardware
  - + Flexibility in performance
  - − Expensive in terms of power



- Simpler control hardware
  - + More transistors for computation
  - + Potentially more power efficient (ops/watt)
  - − More restrictive programming model

CPUs vs. GPUs





Intel Xeon E5-2680 v2

224 Gflop DP peak performance

10 cores

25MB L3 cache (25% of die).

59.7 GB/s bandwidth

115 watt

NVIDIA Kepler K40

1.4 Tflop DP peak performance

2880 cores (FP32), 960 cores (FP64)

1536KB L2 cache (2% of die)

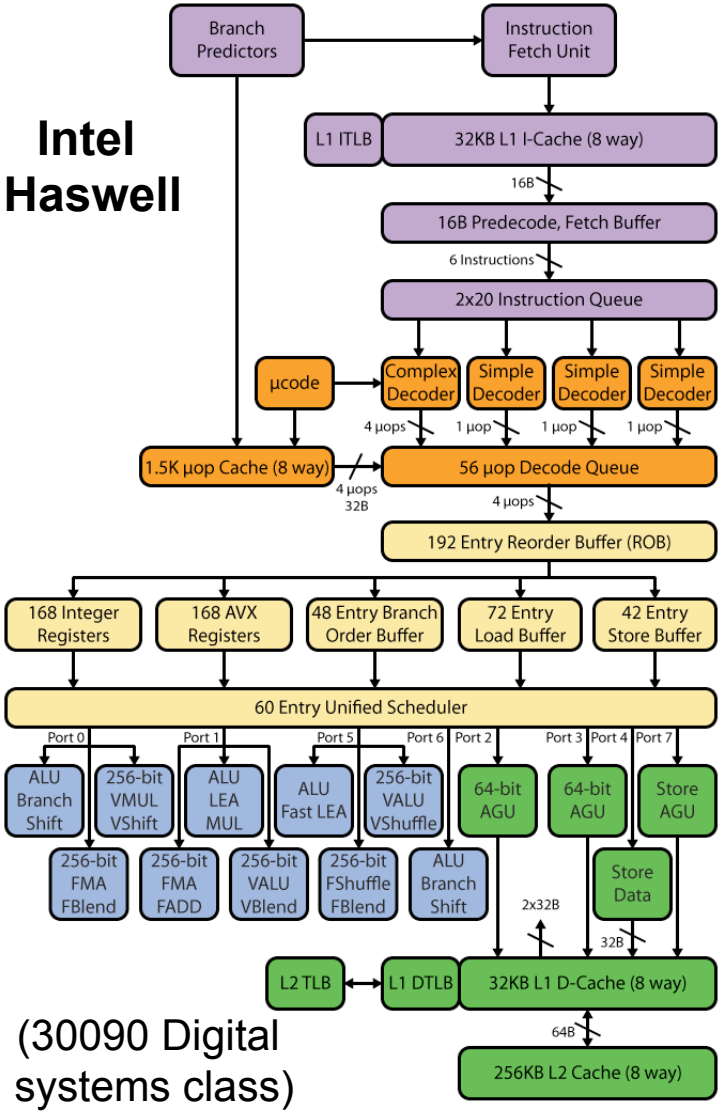
288 GB/s bandwidth

233 watt

SIMD units

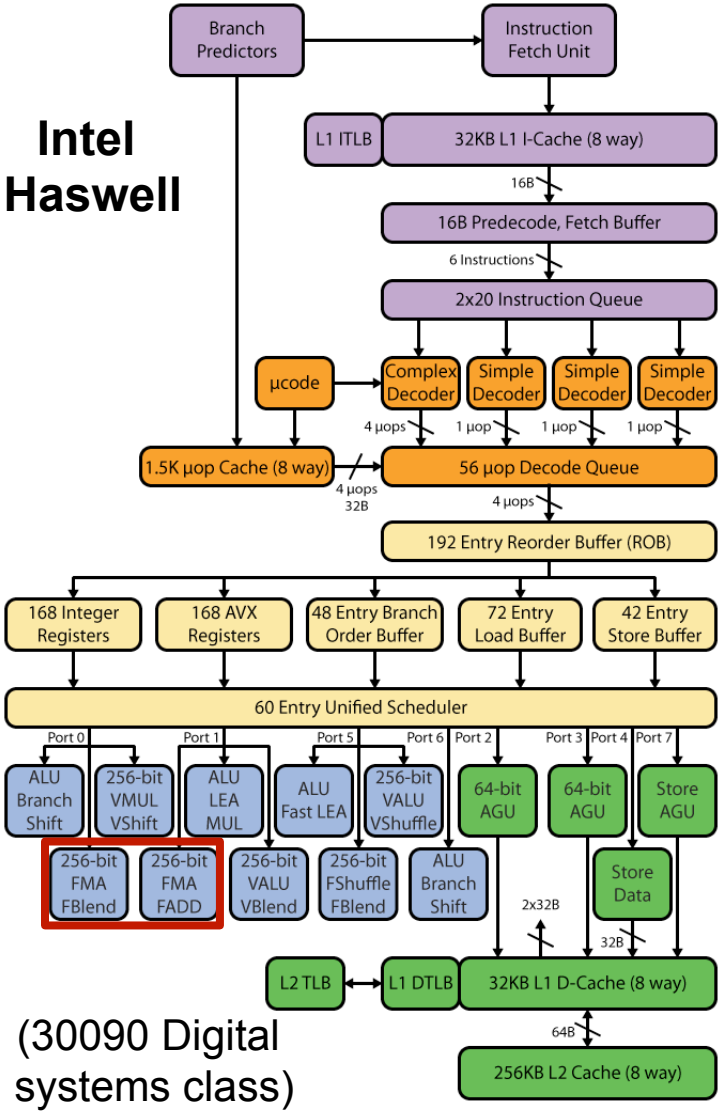
What is a core?

Intel Haswell



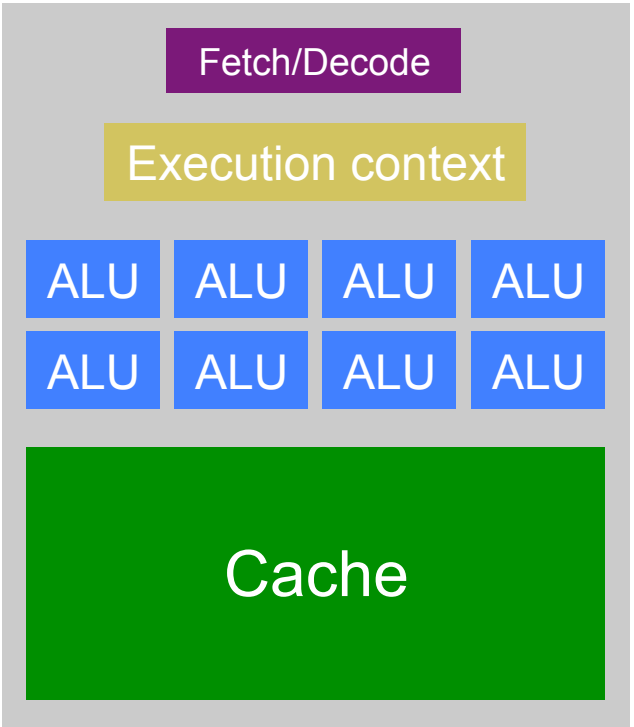
(30090 Digital systems class)

What is a core?



(30090 Digital systems class)

Our generalized view (02614 HPC class)

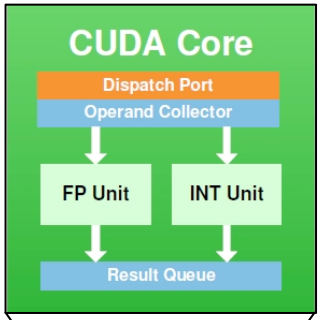


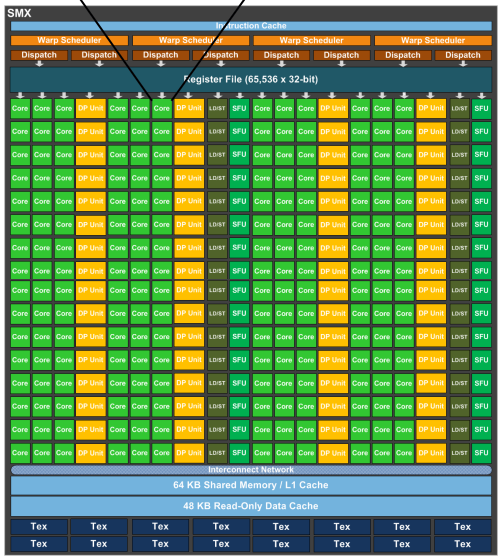
"SIMD unit with 8 lanes"

What is a core?

Nvidia Kepler SMX

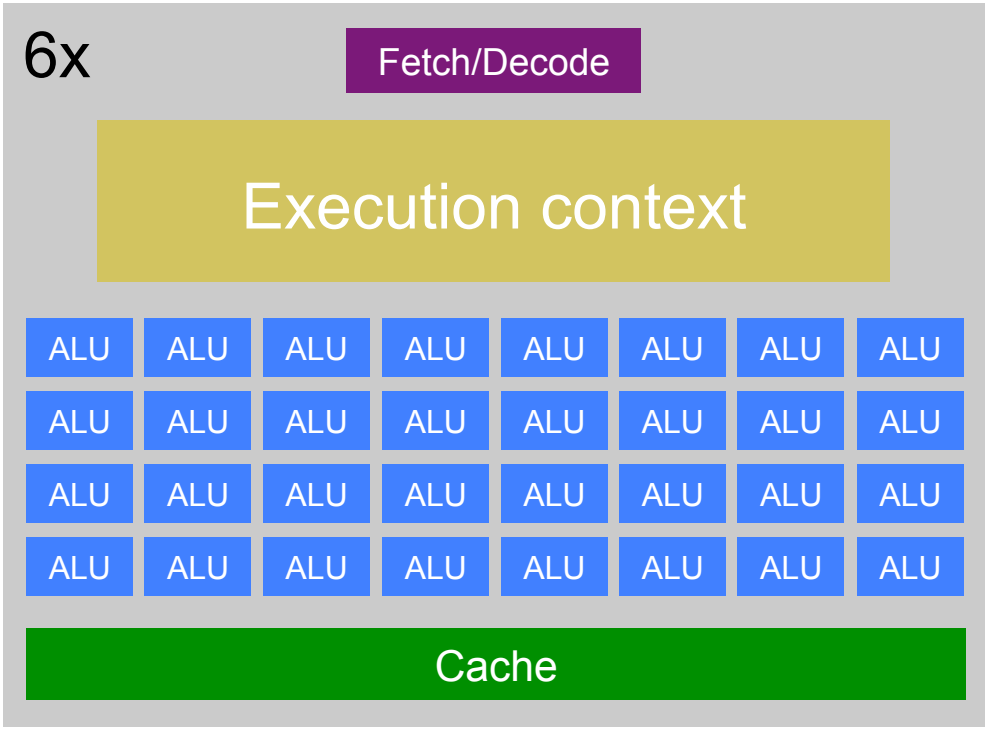
What is a core?





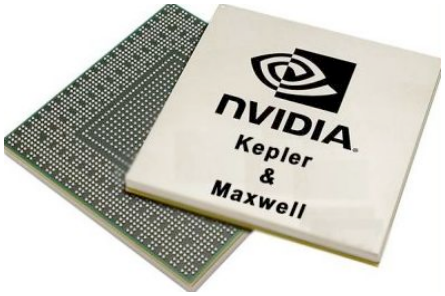
Nvidia Kepler SMX

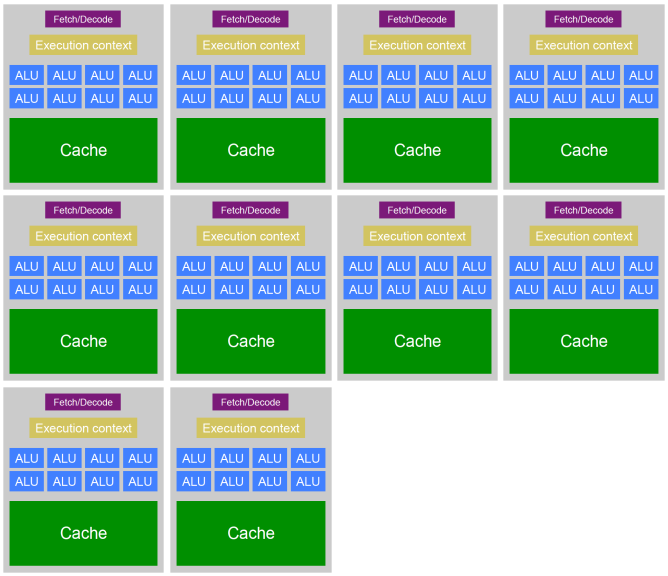
Our generalized view
(02614 HPC class)

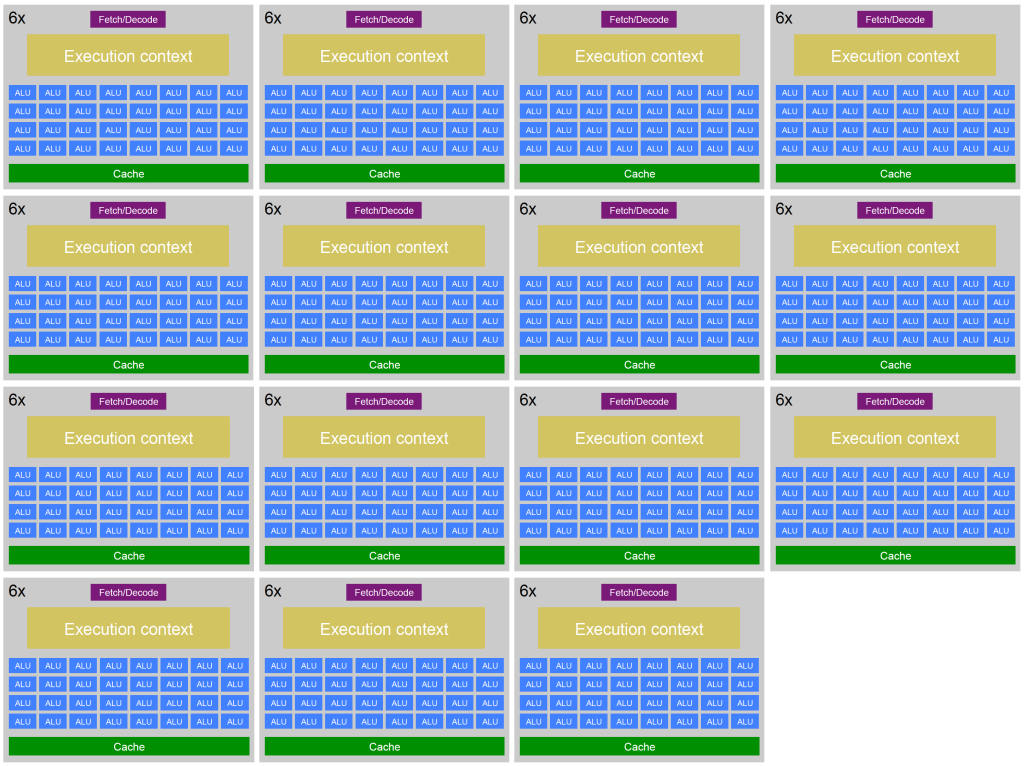


"6 x SIMD unit with 32 lanes"

Modern SIMD processors







Execution model

Execution model

- Maps code to SIMD units
  - SIMD – Single Instruction, Multiple Data
  - SIMT – Single Instruction, Multiple Threads

Execution model

- Maps code to SIMD units
  - SIMD – Single Instruction, Multiple Data
  - SIMT – Single Instruction, Multiple Threads
- CPU (SIMD execution model)
  - The compiler takes care of mapping (scalar) code to the underlying SIMD units
  - We are – supposedly – only meant to give pragmas

Execution model

- Maps code to SIMD units
  - SIMD – Single Instruction, Multiple Data
  - SIMT – Single Instruction, Multiple Threads
- CPU (SIMD execution model)
  - The compiler takes care of mapping (scalar) code to the underlying SIMD units
  - We are – supposedly – only meant to give pragmas
- GPU (SIMT execution model)
  - The compiler takes care of mapping (scalar) code to the underlying SIMD units
  - We are expected to explicitly help the compiler!

What does this mean for you?

$$\pi = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} \frac{4}{1+\left(\frac{i-0.5}{N}\right)^2}.$$

```
double pi(long N)
{
    double sum = 0.0;
    double h = 1.0/N;
    #pragma omp parallel for \
        reduction(+: sum)
    for(long i=1; i<=N; i++) {
       double x = h*(i-0.5);
       sum += 4.0/(1.0+x*x);
    }
    return h*sum;
}
```

What does this mean for you?

$$\pi = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} \frac{4}{1+\left(\frac{i-0.5}{N}\right)^2}.$$


```
double pi(long N)
{
    double sum = 0.0;
    double h = 1.0/N;
    #pragma omp parallel for \
        reduction(+: sum)
    for(long i=1; i<=N; i++) {
       double x = h*(i-0.5);
       sum += 4.0/(1.0+x*x);
    }
    return h*sum;
}
```

```
gcc -O3 -fopenmp pi_cpu.cpp
```

OMP_NUM_THREADS	gcc (s)
1	42.5
2	21.3
4	10.9
8	5.9
16	3.0
20	2.4

What does this mean for you?

$$\pi = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} \frac{4}{1+\left(\frac{i-0.5}{N}\right)^2}.$$


```
double pi(long N)
{
    double sum = 0.0;
    double h = 1.0/N;
    #pragma omp parallel for \
        reduction(+: sum)
    for(long i=1; i<=N; i++) {
        double x = h*(i-0.5);
        sum += 4.0/(1.0+x*x);
    }
    return h*sum;
}
```

```
gcc -O3 -fopenmp pi_cpu.cpp
icc -O3 -fopenmp pi_cpu.cpp
```

OMP_NUM_THREADS	gcc (s)	icc (s)
1	42.5	21.4
2	21.3	10.7
4	10.9	5.4
8	5.9	3.0
16	3.0	1.5
20	2.4	1.2

What does this mean for you?

$$\pi = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} \frac{4}{1+\left(\frac{i-0.5}{N}\right)^2}.$$


```
double pi(long N)
{
    doub
    doub
    #pra                                  l for \
                                          n)
    for(                                    i++) {
        d                                 0.5);
        s                                 x*x);
    }
    retu
}
```

```
subsd %xmm4, %xmm0
mulsd %xmm5, %xmm0
mulsd %xmm0, %xmm0
addsd %xmm3, %xmm0
divsd %xmm0, %xmm7
addsd %xmm7, %xmm1
```

```
subpd %xmm2, %xmm6
mulpd %xmm0, %xmm6
mulpd %xmm6, %xmm6
addpd %xmm1, %xmm6
divpd %xmm6, %xmm9
addpd %xmm9, %xmm3
```

```
gcc -O3 -fopenmp pi_cpu.cpp
icc -O3 -fopenmp pi_cpu.cpp
```

OMP_NUM_THREADS	gcc (s)	icc (s)
1	42.5	21.4
2	21.3	10.7
4	10.9	5.4
8	5.9	3.0
16	3.0	1.5
20	2.4	1.2

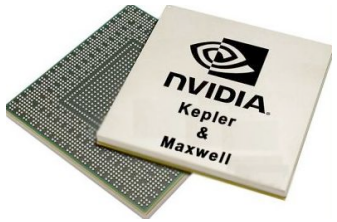
What does this mean for you?

$$\pi = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} \frac{4}{1+\left(\frac{i-0.5}{N}\right)^2}.$$

```
__global__
void pi(long N, double *res)
{
    long idx = threadIdx.x+
        blockDim.x*blockIdx.x;
    double sum = 0.0;
    double h = 1.0/N;
    for(long i=idx+1; i<=N;
     i+=blockDim.x*gridDim.x){
        double x = h*(i-0.5);
        sum += 4.0/(1.0+x*x);
    }
    res[idx] = h*sum;
}
```

What does this mean for you?

$$\pi = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N} \frac{4}{1+\left(\frac{i-0.5}{N}\right)^2}.$$




```
__global__
void pi(long N, double *res)
{
    long idx = threadIdx.x+
        blockDim.x*blockIdx.x;
    double sum = 0.0;
    double h = 1.0/N;
    for(long i=idx+1; i<=N;
     i+=blockDim.x*gridDim.x){
        double x = h*(i-0.5);
        sum += 4.0/(1.0+x*x);
    }
    res[idx] = h*sum;
}
```

```
nvcc -O3 -lgomp pi_gpu.cu
```

blockDim	gridDim	nvcc (s)
1	1024	25.1
32	1024	1.2
64	1024	0.8
128	1024	0.6
256	1024	0.6

Trends in GPU computing

Trends in GPU architectures

- Four generations of high-performance GPUs:

	# GPUs	Name	Year	Architecture	CUDA cap.	CUDA cores	Clock MHz	Mem GiB	SP peak GFlops	DP peak GFlops	Peak GB/s
Fermi	8	Tesla M2050	2012	GF100 (Fermi)	2.0	448	575	2.62	1030	515	148.4
	6	Tesla M2070Q	2012	GF100 (Fermi)	2.0	448	575	5.25	1030	515	150.3
	2	*GeForce GTX 680	2012	GK104-400 (Kepler)	3.0	1536	1058	1.95	3090	128	192.2
Kepler	3	Tesla K20c	2013	GK110 (Kepler)	3.5	2496	745	4.63	3524	1175	208
	5	Tesla K40c	2013	GK110B (Kepler)	3.5	2880	745 / 875	11.17	4291 / 5040	1430 / 1680	288
	8	Tesla K80c (dual)	2014	GK210 (Kepler)	3.7	2496	562 / 875	11.17	2796 / 4368	932 / 1456	240
Maxwell	1	*GeForce GTX TITAN X	2015	GM200-400 (Maxwell)	5.2	3072	1076	11.92	6144	192	336
Pascal	8	*TITAN X	2016	GP102 (Pascal)	6.1	3584	1417 / 1531	11.90	10157 / 10974	317.4 / 342.9	480

Source: http://www.hpc.dtu.dk/?page_id=2129

- More cores, less clock rate, more bandwidth.

Trends for GPU algorithms

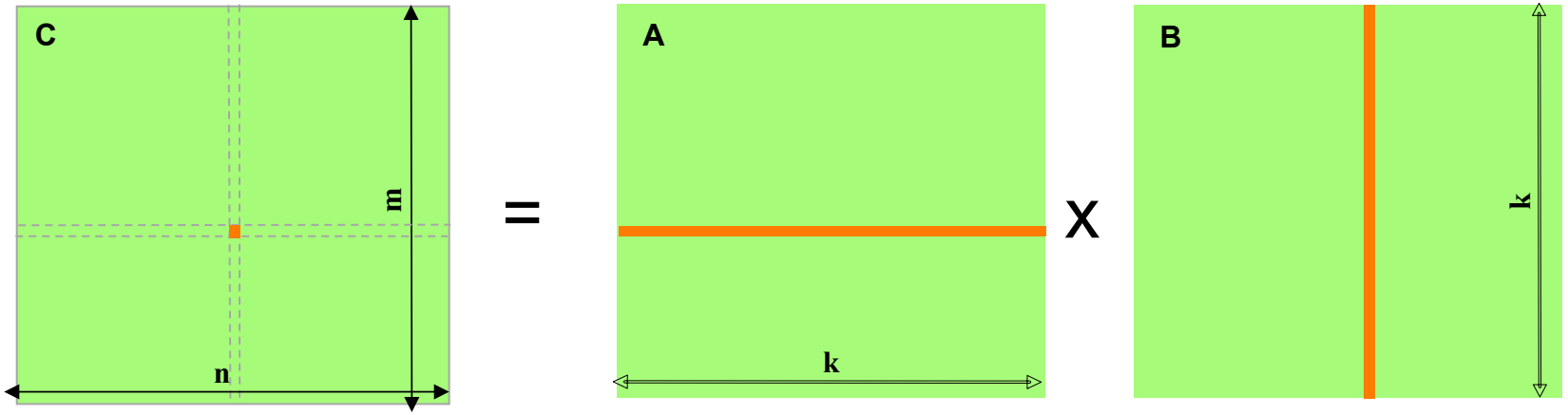
- Synchronization-reducing algorithms
  - Embarrassingly parallel techniques
- Communication-reducing algorithms
  - Use methods which have lower bound on communication
- Mixed precision methods
  - >3x speed of ops and 2x speed for data movement
- Autotuning
  - Today's machines are too complicated, build "smarts" into software to adapt to the hardware

Heterogeneous computing

- Heterogeneous computing means using all the available hardware for a given task
  - So that each processor type can do what it does best
- a) CPU is the master
  - Manages resources (disk, input, etc.) and the GPU
- b) GPU is the ”accelerator” used for special tasks
  - c) Data must be transferred to / from the GPU memory
- Programming in a heterogeneous model requires handling both a), b) and c).
- Real challenge is to know what task goes where

Which problems fit the GPU?

- Problems that use many flops but little data
- Problems that have a high degree of parallelism
- One kind of problem is “matrix computations”:
  - E.g., matrix-vector and matrix-matrix multiplication

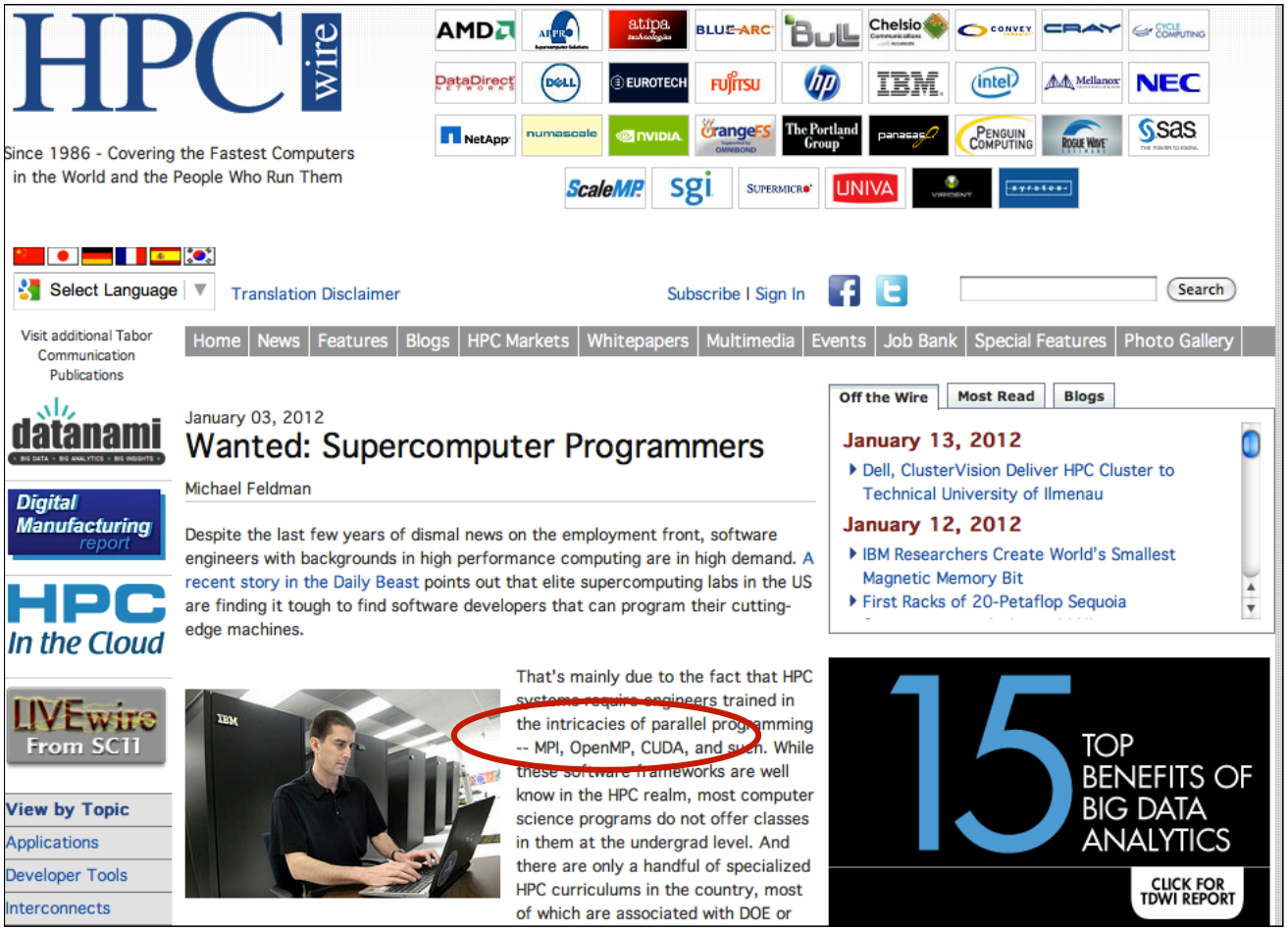


- So-to-say: “the heart” of scientific computing

Motivation to learn about GPUs

Motivation…

2012



http://www.hpcwire.com/hpcwire/2012-01-03/wanted:_supercomputer_programmers.html

More motivation…

2013

Job openings
January 2013

PhaseOne, Frederiksberg

Seeking new challenges.

You might already be employed within photographic image processing or a related field. You have in-depth experience within the above mentioned image processing algorithms, pipeline designs as well as great insight into CPU and GPU hardware architectures and I/O constraints, caching and usage of vector processors. You know how to write API code and design efficient C and C++ image processing code, and you may even have experience within OpenCL or CUDA development and optimization.

We offer a really exciting and challenging job in a startup-like company where R&D is regarded as "the key to success". We are regarded as the "best in class" by the world's top photographers, and we need to continue to stride for perfection. Recently we introduced Capture One 7, the world's first commercial raw converter utilizing OpenCL for both Mac and Windows, enabling speedup of up to 10 times compared to other CPU based programs.

Interested?

Send your application and resume to jobs@phaseone.com marked "ImageCore2013-1". If you need more information please go to www.phaseone.com.

Dantec Dynamics, Skovlunde

The Ideal Candidate

A successful candidate must have interests in working together with our scientific and industrial partners in an effort to most effectively support Dantec Dynamics' international customer base with their high end measurement applications. In addition, the ideal candidate must be creative, have strong analytical skills and some knowledge of GPU, SSE and/or 3D API (DirectX/OpenGL) programming as well as experience with instrumentation/hardware integration (digital cameras, laser systems, or sensors).

Application

Please send your application marked "***Software Developer - Denmark***" via email to our HR department: hr@dantecdynamics.com

Applications are handled on an ongoing basis.

Academic motivation…

2014 PhD stud. DTU Compute 02614 Teaching assistent DANSIS Graduate Prize of 2013

Some completed and ongoing GPULab projects – Jan 2014:

MSc: Max la Cour Christensen, M. and Eskildsen, K. L. *Nonlinear Multigrid for Efficient Reservoir Simulation*. 2012.

BSc: Mieritz, Andreas. *GPU-Acceleration of Linear Algebra using OpenCL*. 2012.

Special course: Leo Emil Sokoler and Oscar Borries, *Conjugate Gradients on GPU using CUDA*, 2012.

MSc: Høstergaard, Gade-Nielsen, Nicolai Fog, *Implementation and evaluation of fast computational methods for high-resolution ODF problems on multi-core and many-core systems*, 2010

PhD: Stefan L. Glimberg, *Designing Scientific Software for Heterogeneous Computing*, 2010-2013

PhD: Nicolai Fog Gade-Nielsen, *Scientific GPU Computing for Dynamical Optimization*, 2010-2014

PhD: Oscar Borries, *Large-Scale Computational Electromagnetics for Reflector Antenna Analysis*, 2011-

PostDoc DTU Physics Ind. PostDoc DTU Compute

Some available projects:

-Acceleration of Wind Turbine Vortex Simulation (in collaboration with DTU WIND).

-3D image reconstruction using GPU acceleration (in collaboration with University of Antwerp).

-Computational Electromagnetics for Reflector Antennas using Accelerators (in collaboration with TICRA).

-Fast Large-scale Banded Solver on the GPU (in collaboration with RESON A/S).

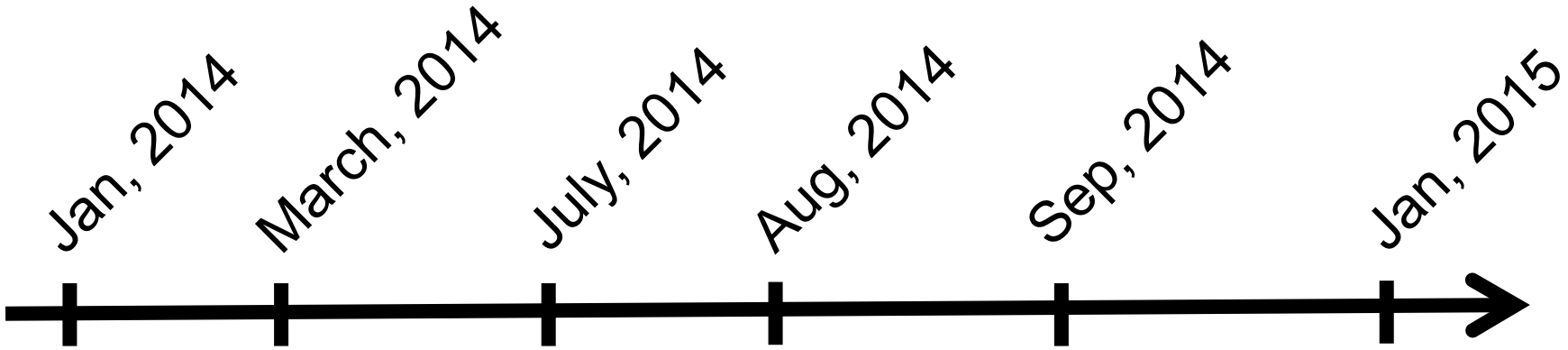
-Sparse matrix computations in genome-wide association studies (in collaboration with GenoKey).

-Your own project idea!

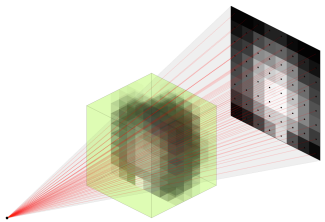
More motivation…

2015

Kenneth's GPU time line:

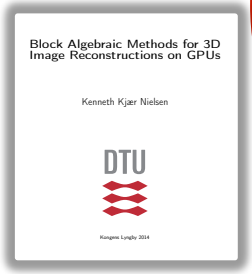


02614 MSc Project start



Job interview Xnovotech

MSc Project hand in



Hired by Xnovotech as GPU software developer

Proposal for a joint paper with ASTRA

xnovotech

GPU support in HPC – Nov 2015

2016

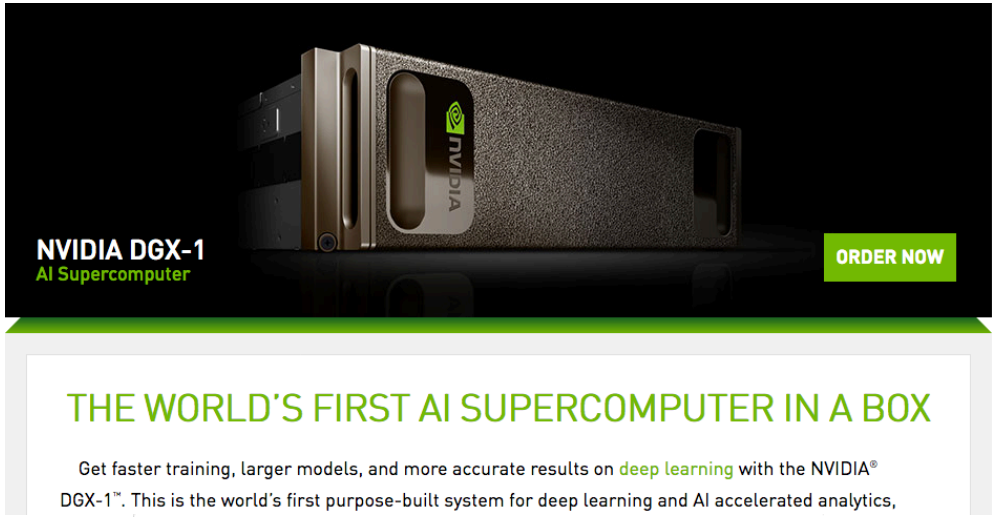
- Yearly report
  - http://www.intersect360.com/industry/reports.php?id=131
  - "For the most commonly accelerated codes, performance increases are usually in the 2x to 10x range."
  - See more detailed list: http://www.nvidia.com/content/gpu-applications/PDF/gpu-applications-catalog.pdf

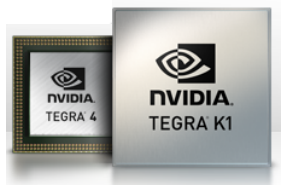
Table 1. GPU-Accelerated Applications

Software Supplier - Package Name	Mentions	Rank	GPU Accelerated
Gaussian – Gaussian	118	1	Under Development
ANSYS – Fluent	114	2	Accelerated
Gromacs.org – GROMACS	93	3	Accelerated
Dassault Systemes - SIMULIA Abaqus	91	4	Accelerated
U of Illinois, UC – NAMD	75	5	Accelerated
NCAR – WRF	72	6	Accelerated
U of Vienna – VASP	68	7	Accelerated[2]
OpenFoam Foundation – OpenFOAM	54	8	Accelerated
LSTC - LS-DYNA	53	9	Accelerated
AmberMD.org – AMBER	50	10	Accelerated
NCBI – BLAST	46	11	No
PNNL – NWChem	35	12	Accelerated
Iowa State University – GAMESS	30	13	Under Development
Quantum-espresso.org - Quantum ESPRESSO	30	14	Accelerated
Sandia Nat Lab – LAMMPS	28	15	Accelerated
ANSYS – CFX	17	16	No
Exelis – IDL	16	17	Accelerated
MSC Software – NASTRAN	16	18	Accelerated
CD-adapco - Star-CD	15	19	No
Schrodinger – Schrodinger	14	20	Accelerated
NCAR – CCSM	14	21	No
COMSOL – COMSOL	14	22	No
ANSYS - ANSYS Mechancial	12	23	Accelerated
Charmm.org – CHARMM	11	24	Accelerated
CD-adapco - Star-CCM+	10	25	No
CP2K.org - CP2k	10	26	Accelerated
Thermo Scientific – Sequest	9	27	No
Kitware – ParaView	9	28	Accelerated
USQCD.org – MILC	8	29	Accelerated
CERN – CMSSW	8	30	Accelerated
Scripps Research Institute – AutoDock	8	31	No
Open Source – MrBayes	8	32	No
Q-Chem - Q-Chem	8	33	Accelerated
LLNL – VisIT	8	34	Accelerated
Terraspark - Insight Earth	7	35	Accelerated
Open Source – Galaxy	7	36	No
Illumina – Casava	7	37	No
CMOP – SELFE	7	38	No
Tecplot, Inc. – Tecplot	6	39	Accelerated
SCM – ADF	6	40	Accelerated
USQCD.org – CHROMA	6	41	Accelerated
Dassault Systemes - Accelrys Materials Studio	6	42	No
Abinit.org – ABINIT	6	43	Accelerated
NASA – Overflow	6	44	No
CIG - Specfem3d	6	45	Accelerated
Kitware – VTK	6	46	Accelerated
Metacomp - CFD++	6	47	No
Open Source – Nbody	6	48	Accelerated
Altair Engineering – HyperWorks	6	49	Accelerated
U of Massachusetts – FVCOM	5	50	No

Google trends: GPUs still going up

2017





End of lecture