High Performance Computing

Why it is important

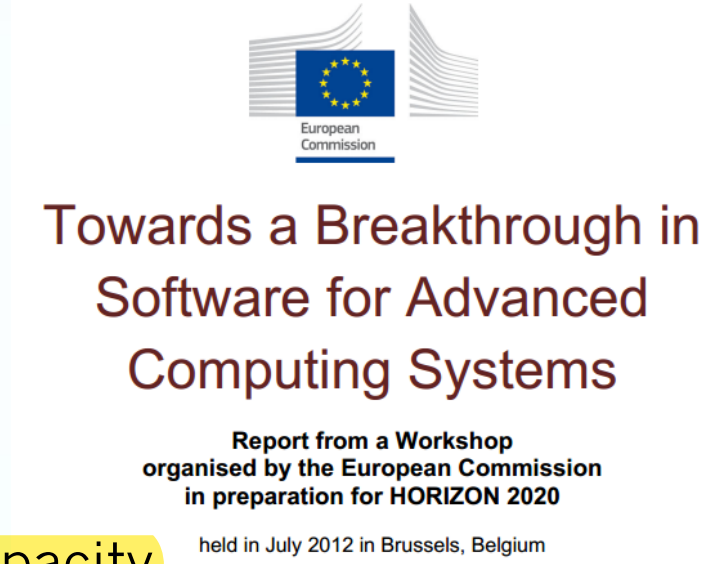
Introduction

HPC = 

?

NO

Introduction



Software development has not evolved as fast as hardware capability and network capacity.

Nominal and sustained performance of computing systems is further diverging, unless they are manually optimised which limits portability to other systems.

Development and maintenance of software for advanced computing systems is becoming increasingly effort-intensive requiring dual expertise, both on the application side and on the system side.

...

In order to program the next generation of computing systems, everyone must become a parallel programmer!

1965

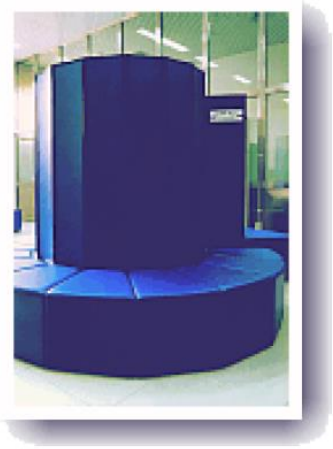
8Mb

2015

128Gb

STORAGE

1975



400 Mflops

2015

173 Gflops (GPU)

PERFORMANCE

1970

```
  PROGRAM HELLO
C
      REAL A(10,10)
      DO 50 I=1,10
        CALL DGEMM(N,10,I,J,A)
50    CONTINUE
```

2018

```
  PROGRAM HELLO
C
      REAL A(10,10)
      DO 50 I=1,10
        CALL DGEMM(N,10,I,J,A)
50    CONTINUE
```

SOFTWARE

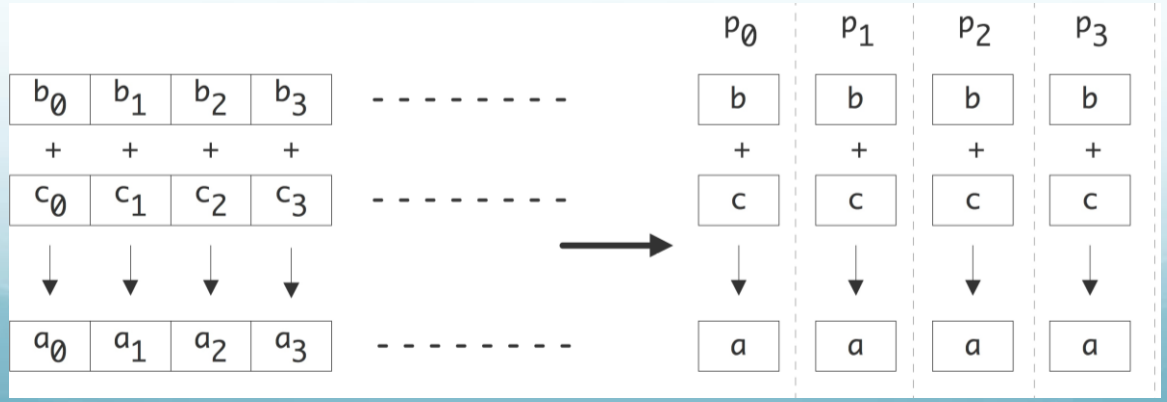
What is Parallel Computing?

Simultaneous use of multiple resources to solve a computational problem.

In scientific codes, there is often a large amount of work to be done, and it is often regular to some extent, with the same operation being performed on many data.

```
for (i=0; i<n; i++)

a[i] = b[i] + c[i];
```



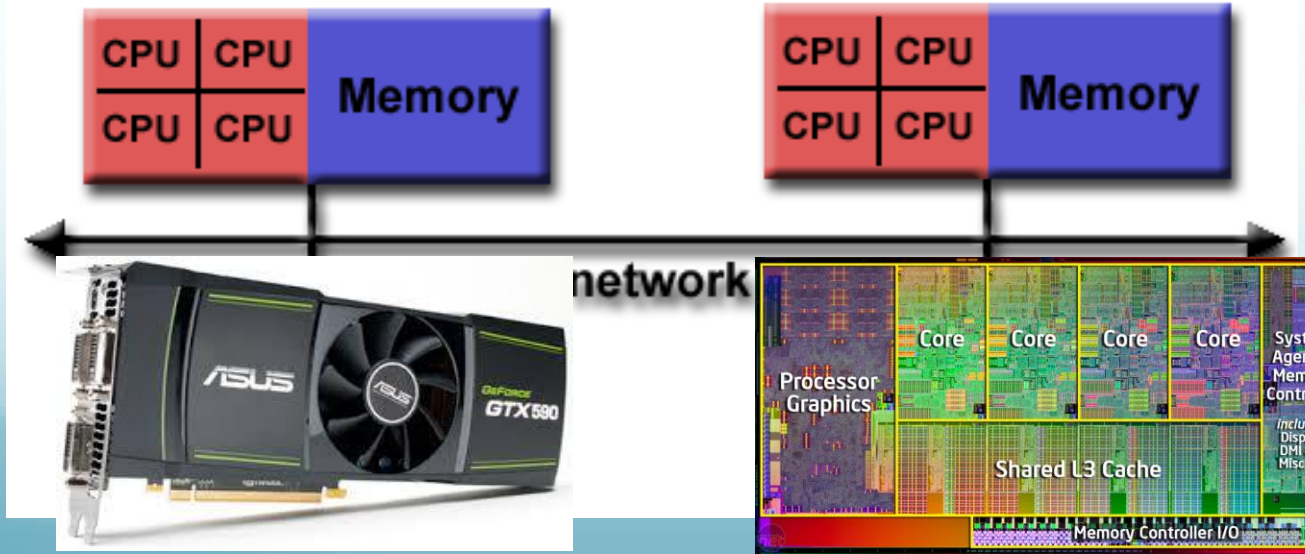
Why use Parallel Computing?

- Save time
- Solve larger problems
  - Many problems are so large and/or complex that it is impractical or impossible to solve them on a single computer
  - More memory available
  - Higgs discovery “only possible because of the extraordinary achievements of Grid computing” — Rolf Heuer, CERN DG
- Use of non-local resources
  - Using compute resources on a wide area network, or even the Internet when local compute resources are scarce. E.g. SETI@home.

Actual CPUs and GPUs

- Modern CPUs have vector instructions that can perform multiple instances of an operation simultaneously. On Intel processors this is known as SIMD Streaming Extensions (SSE) or Advanced Vector Extensions (AVX). GPUs are vector processors.
- By far the most common parallel computer architecture are MIMD (also known as Single Program Multiple Data, SPMD): the processors execute multiple, possibly differing instructions, each on their own data. There is a great variety in MIMD computers. Some of the aspects concern the way memory is organized, and the network that connects the processors.


IT'S A
JUNGLE
OUT THERE!


CPU
CPU
CPU
CPU
Memory
CPU
CPU
CPU
CPU
Memory
network
ASUS
GeForce
GTX 590
Processor Graphics
Core
Core
Core
Core
System Agent & Memory Controller
Including Display, DMI and Misc. I/O
Shared L3 Cache
Memory Controller I/O

Parallel Programming

The simplest case is the **embarrassingly parallel problem**, where little or no effort is required to speedup the process

- No dependency/ communication between the parallel tasks

Examples :

- Distributed relational database queries
- Rendering of computer graphics
- Event simulation and reconstruction in particle physics
- Brute-force searches in cryptography
- Ensemble calculations of numerical weather prediction

BUT normally you have to work (hard) to parallelize an application!

To summarize

In order to write efficient scientific codes, **it is important to understand the resource architecture**.

The difference in speed between two codes that compute the same result can range from a few percent to orders of magnitude, depending only on factors relating to how well the algorithms are coded for the specific architecture.

Clearly, it is not enough to have an algorithm and “put it on the computer”: some knowledge of computer architecture is useful, sometimes crucial.

TOP500® SUPERCOMPUTER SITES

JUNE 2013 -NOVEMBER 2015

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
				62%		
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
				65%		
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
				85%		

88% of the cores

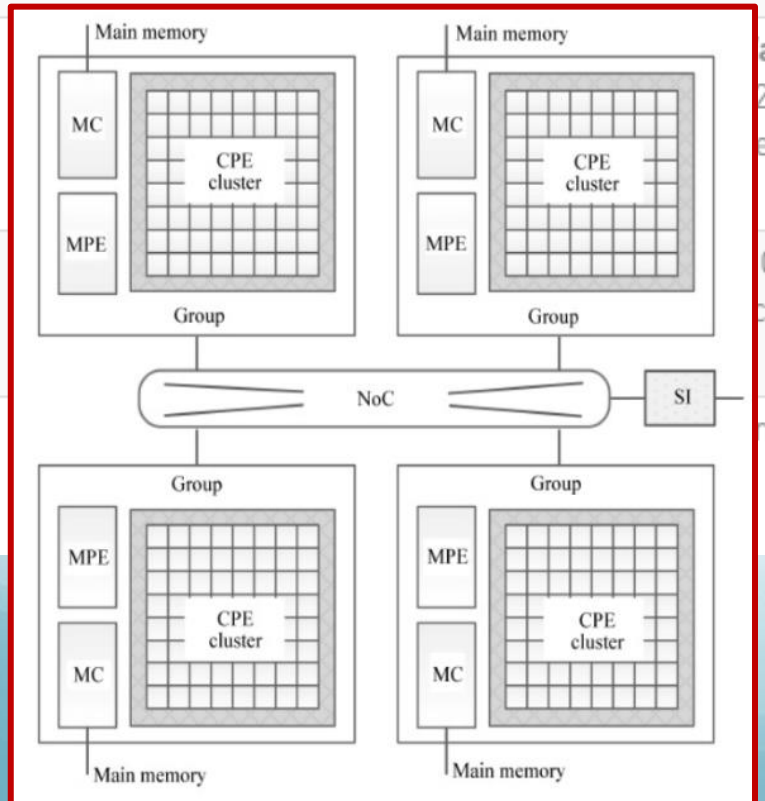
46% of the cores

Homogeneous cores

TOP 500
The List.

NOVEMBER 2016

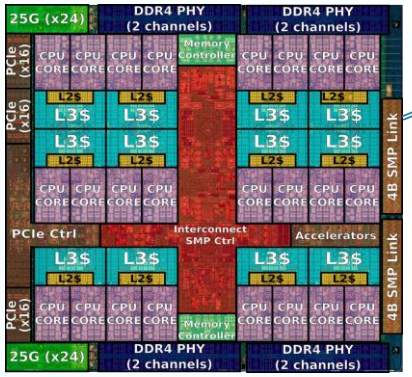
Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6 **74%**	125,435.9	15,371
2		**ay-2)** - TH-IVB-FEP Cluster, 2 12C 2.200GHz, TH eon Phi 31S1P	3,120,000	33,862.7	54,902.4	17,808
3		Opteron 6274 16C 2.200GHz, connect, NVIDIA K20x	560,640	17,590.0	27,112.5	8,209
4						



The ShenWei 26010 is a 260-core, 64-bit RISC chip that exceeds 3 teraflops at maximum tilt, putting it on par with Intel's Knight's Landing Xeo Phi.

TOP 500
The List.

JUNE 2019



Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/SC/Oak Ridge National Laboratory United States	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM	2,414,592	148,600.0	200,794.9	10,096
2	DOE/NNSA/LLNL United States	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM / NVIDIA / Mellanox	1,572,480	94,640.0	125,712.0	7,438
3	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
4	National Super Computer Center in Guangzhou China	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 NUDT	4,981,760	61,444.5	100,678.7	18,482
5	Texas Advanced Computing Center/Univ. of Texas United States	**Frontera** - Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR Dell EMC	448,448	23,516.4	38,745.9	
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray Inc.	387,872	21,230.0	27,154.3	2,384
7	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect	979,072	20,158.7	41,461.2	7,578

74%

https://en.wikichip.org/wiki/ibm/microarchitectures/power9

Today

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442,010.0	537,212.0	29,899
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096

...

119	**davinci-1** - BullSequana X, AMD EPYC 7402 24C 2.8GHz, NVIDIA A100, Mellanox HDR Infiniband, Atos Leonardo SpA Italy	38,400	3,449.0	6,412.0	

- https://www.arm.com/blogs/blueprint/fujitsu-a64fx-arm
- https://www.top500.org/news/fugaku-holds-top-spot-exascale-remains-elusive/

Cloud and High Performance Computing

TOP500 SUPERCOMPUTER SITES

Rank	System	Vendor	Total Cores	Rmax (TFlops)	Rpeak (TFlops)
101	Amazon EC2 Cluster, Intel Xeon E5-2680v2 10C 2.800GHz, 10G Ethernet	Self-made	26,496	484.2	593.5

18 hours, $33K, and 156,314 cores: Amazon cloud HPC hits a "petaflop"

1.21 petaflops? Great scott!

To get all those cores, Cycle's cluster ran simultaneously in Amazon data centers across the world, in Virginia, Oregon, Northern California, Ireland, Singapore, Tokyo, Sydney, and São Paulo. The bill from Amazon ended up being $33,000.

$4,829-per-hour supercomputer built on Amazon cloud to fuel cancer research

A 50,000-core supercomputer deployed on Amazon shows the cloud's potential

The cluster used a mix of 10 Gigabit Ethernet and 1 Gigabit Ethernet interconnects. However, the workload was what's often known as "embarrassingly parallel," meaning that the calculations are independent of each other. As such, the speed of the interconnect didn't really matter.

- The advantage of pay-as-you-go computing has been an industry goal for many years.
- The Globus Project has shown the power of Grid computing.
- Cloud computing takes Grid computing to a whole new level by using virtualization to encapsulate an operating system (OS) instance

Operational costs 1: energy costs

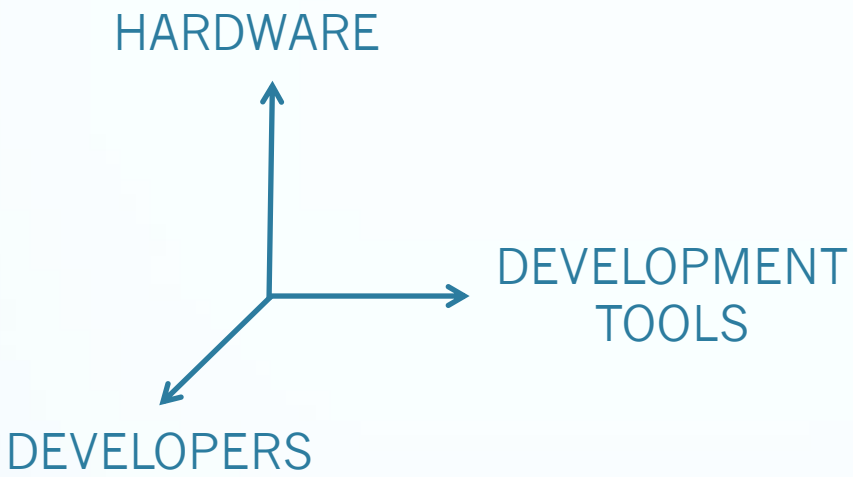
Total cost of ownership: total cost of acquisition and operating costs

	MFLOPS/W			KW
38	2,176.58	DOE/NNSA/LLNL	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom	7,890.00
43	2,143.03	DOE/SC/Oak Ridge National Laboratory	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x	8,208.00
49	1,901.54	National Super Computer Center in Guangzhou	Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P	17,808.00

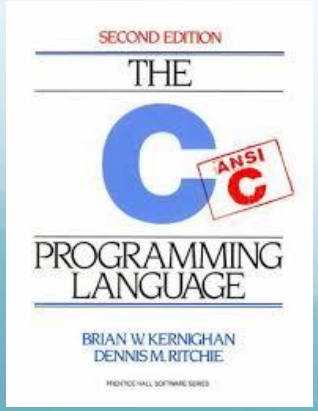
Supercomputer‘s lifelong energy costs almost equal the investment costs

Tianhe-2: ≈ 20MW ≈ $20 million/year for electricity

Operational costs 2: human resources and tools



- Development tools
  - GNU gcc – free
  - Intel Parallel Studio XE (oneAPI) – ~~from 699 US$~~ NOW FREE
  - PGI Accelerator Workstation – ~~from 759 US$~~ again NOW FREE
  - NVIDIA HPC SDK – free



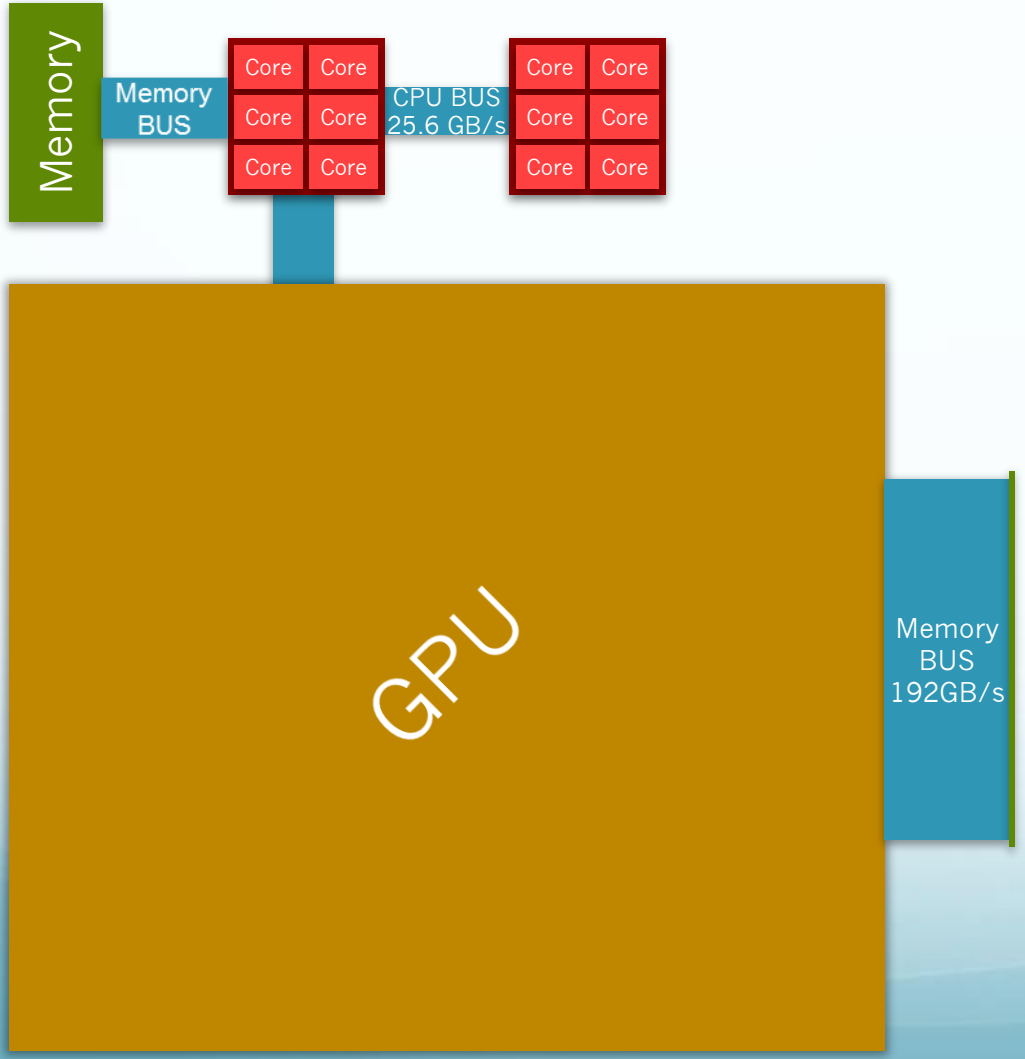
OpenMP MPI NVIDIA CUDA OpenACC DIRECTIVES FOR ACCELERATORS

Raw numbers or real performances ?

- A workstation with dual 12-cores CPUs and 4 GPUs
  - 139 GFLOPs SP / 69 GFLOPs DP per CPU
  - 3.5 TFLOPs SP / 1.2 TFLOPs DP per GPU
  - = 14.3 TFLOPs SP / 4.9 TFLOPS DP at about 12,000 US$
  - 1.6 KWatt
- First position in June 2001,
  at the bottom of the list in June 2008
- How can programmers exploit such performance ?
  - programming paradigms and languages
  - development tools

Raw numbers or real performances ?

Architecture of test workstation (fp32)



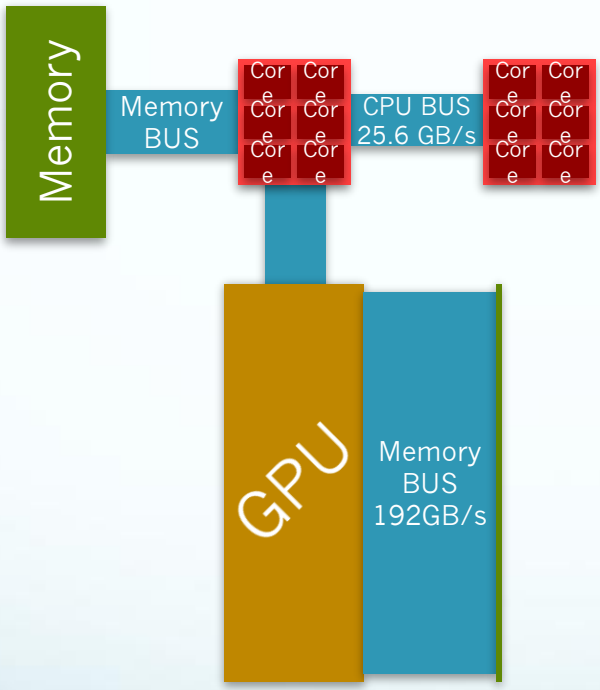
2 Intel Xeon E5645 (115 Gflops, 9.6 per core)

64 GB main memory

nVidia GTX 580 (1581 Gflops)

1.5 GB GPU memory

Architecture of test workstation (fp64)



2 Intel Xeon E5645 (58 Gflops)

64 GB main memory

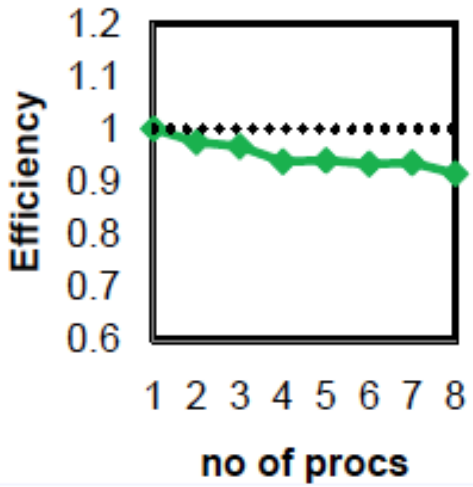
nVidia GTX 580 (198 Gflops)

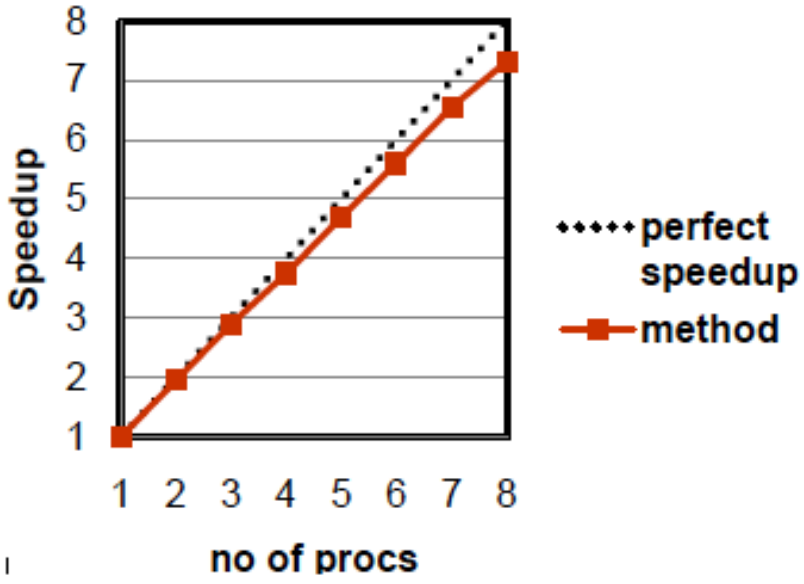
1.5 GB GPU memory

Parallel Computing Performance Metrics

Let T(n,p) be the time to solve a problem of size n using p processors

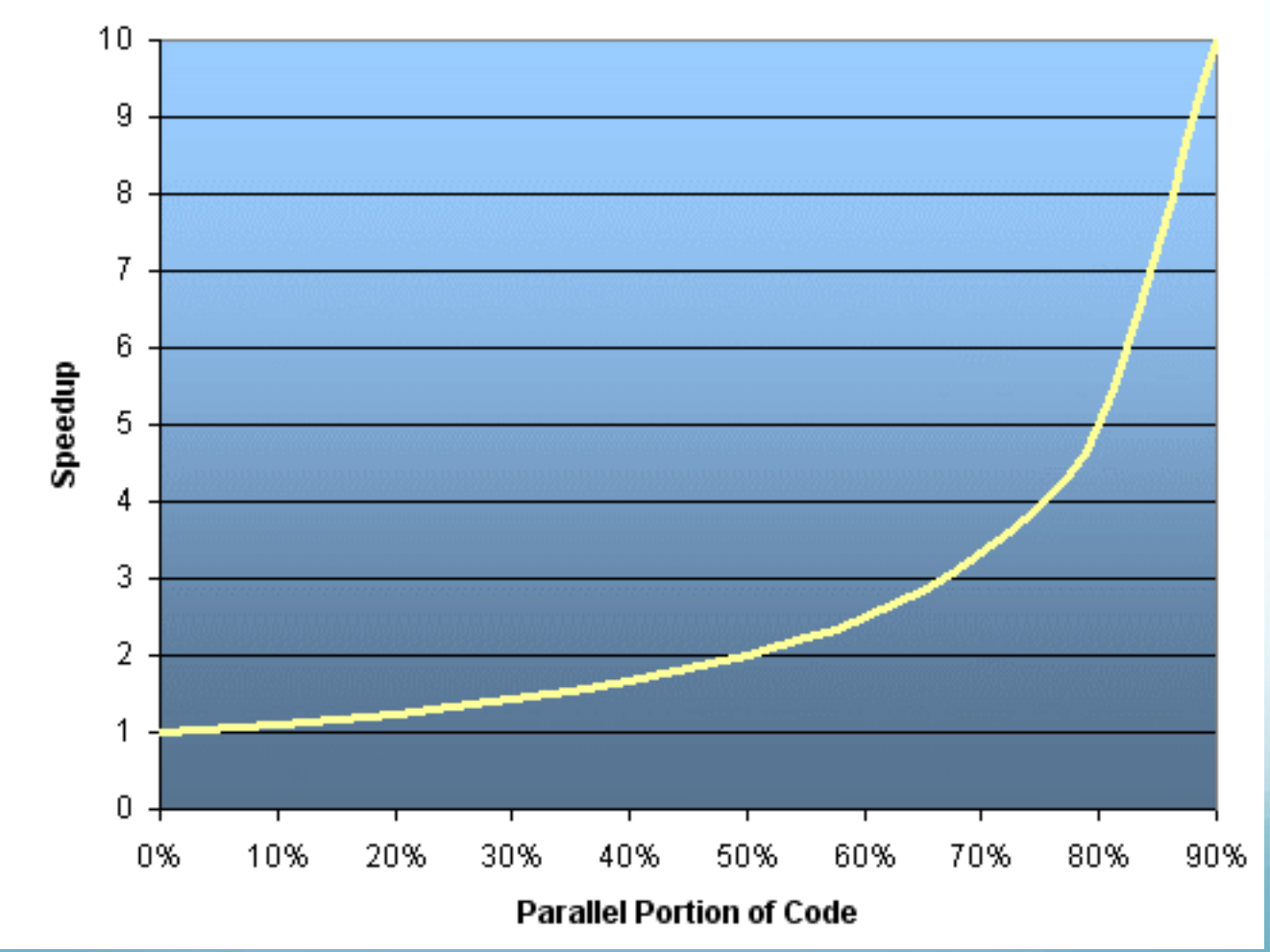
- Speedup: S(n,p) = T(n,1)/T(n,p)
- Efficiency: E(n,p) = S(n,p)/p





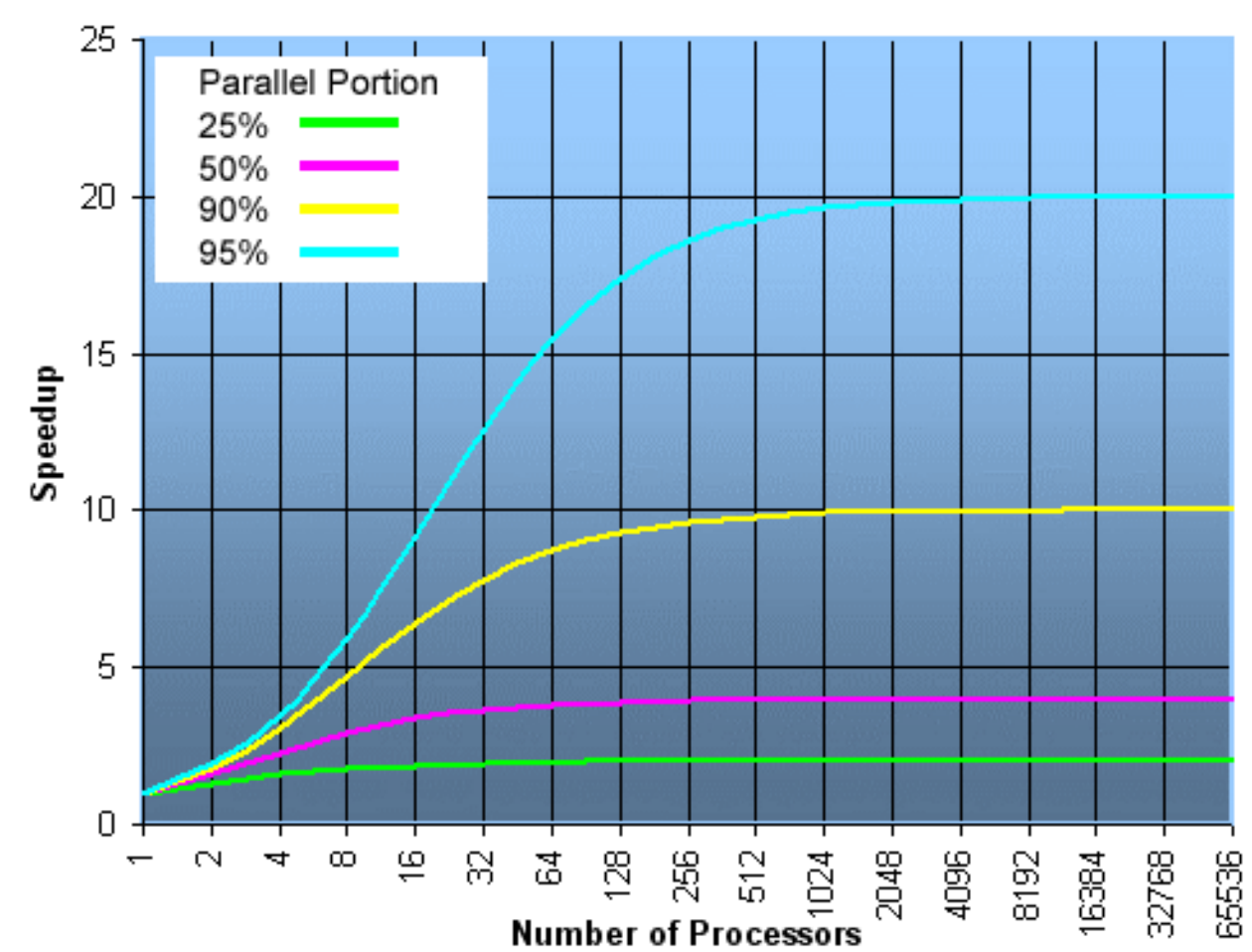
Amdahl's Law

Maximal Speedup = 1/(1-P), P parallel portion of code



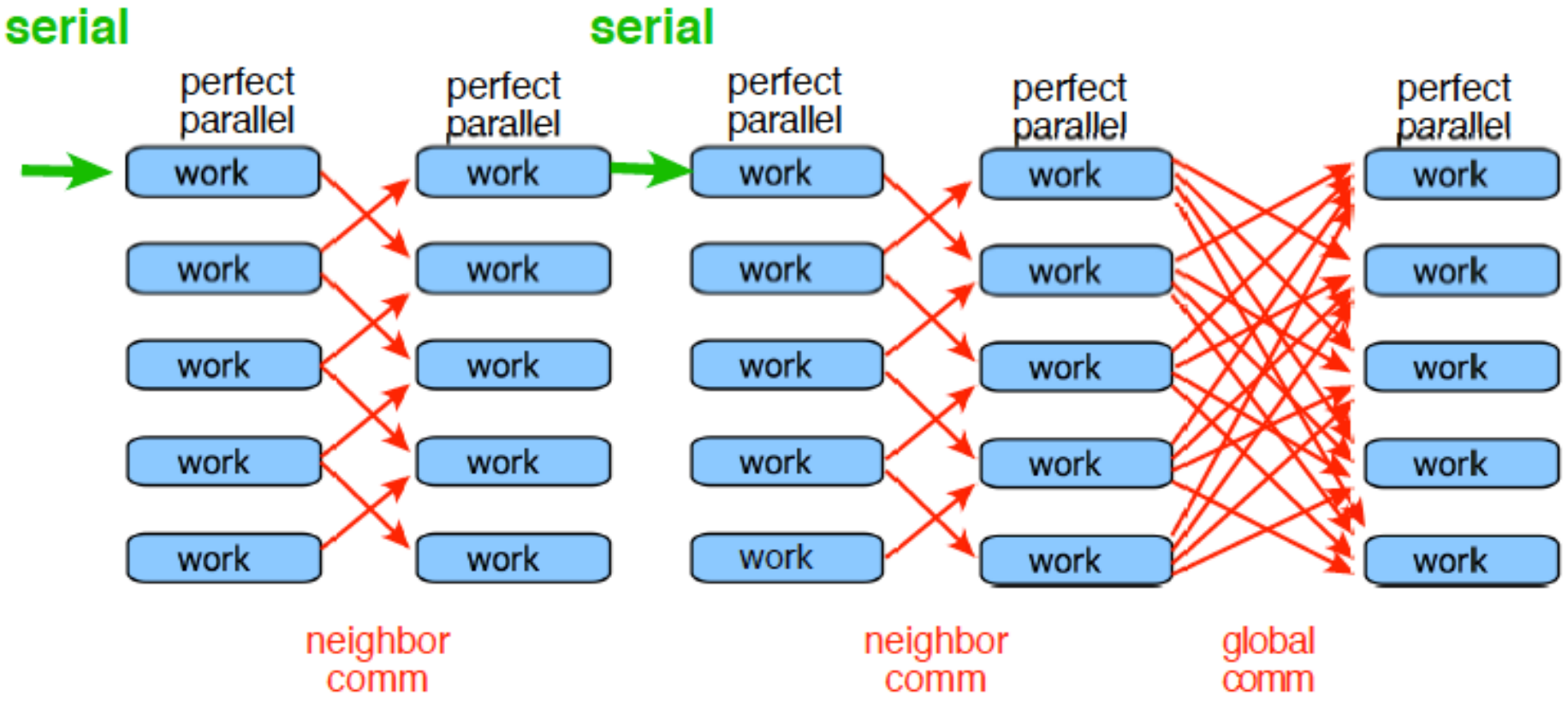
Amdahl's Law

Speedup = 1/((P/N)+S),
N no. of processors, S serial portion of code



Amdahl Was an Optimist

Parallelization usually adds communications/overheads

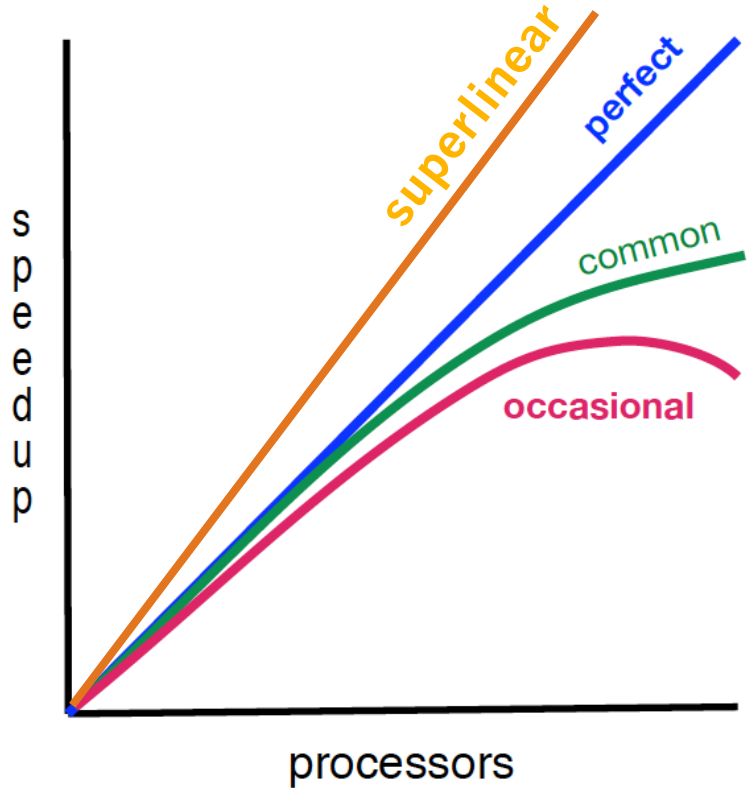


To summarize

$0 < \text{Speedup} \leq p$

$0 < \text{Efficiency} \leq 1$

Linear speedup : speedup = p.



Amdahl was a Pessimist

Superlinear speedup is very rare. Some reasons for speedup > p (efficiency > 1)

- Parallel computer has p times as much RAM so higher fraction of program memory in RAM instead of disk.

An important reason for using parallel computers

- In developing parallel program a better algorithm was discovered, older serial algorithm was not best possible.

A useful side-effect of parallelization

- In general, the time spent in serial portion of code is a decreasing fraction of the total time as problem size increases.

The lesson is

- Linear speedup is rare, due to communication overhead, load imbalance, algorithm/architecture mismatch, etc.
- Further, essentially nothing scales to arbitrarily many processors.
- However, for most users, the important question is:

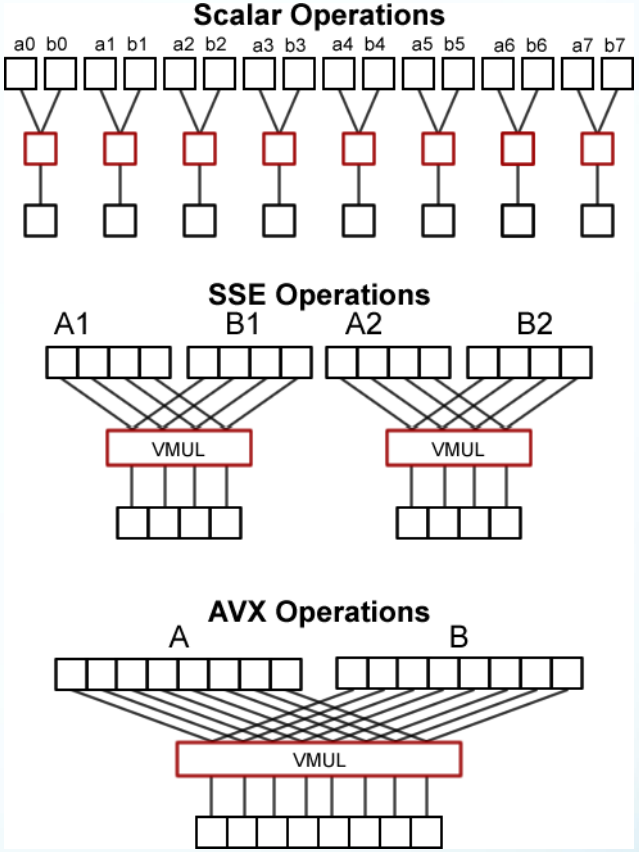
Have I achieved acceptable performance on my software for a suitable range of data and the resources I'm using?

The N-Body simulation

```
for (k=0; k<timesteps; k++) {
  swap(oldbodies,newbodies);
  for (i=0; i<N; i++) {
      tot_force_i[X] = tot_force_i[Y] = tot_force_i[Z] = 0.0;
      for (j=0; j<N; j++) {
          if (j==i) continue;
          //20 floating point operations
          r[X] = oldbodies[j].pos[X] - oldbodies[i].pos[X];
          // analogous for r[Y] and r[Z]
          distSqr = r[X]*r[X] + r[Y]*r[Y] + r[Z]*r[Z] + EPSILON2;
          distSixth = distSqr * distSqr * distSqr;
          invDistCube = 1.0/sqrtf(distSixth);
          s = oldbodies[j].mass * invDistCube;
          tot_force_i[X] += s * r[X];
          // analogous for Y and Z
      }
      //24 flops
      dv[X] = dt * tot_force_i[X] / oldbodies[i].mass;
      newbodies[i].pos[X] += dt * ( oldbodies[i].vel[X] + dv[X]/2 );
      newbodies[i].vel[X] = oldbodies[i].vel[X] +dt * dv[X];
      // analogous for Y and Z
  }
}
```

N-Body – sequential algorithm

- Maximum theoretical performances:
  9.6 GFLOPs (single core – **SIMD** instructions – 551$ CPU, 700$ icc)
- All-pairs algorithm $O(N^2)$ x timesteps
- We consider timestep=100 and N=1K, 10K




```
void multiply(float *A, float *B, float * C, int size) {
        for (i = 0; i < size; i++) {
                C[i] = A[i] * B[i];
        }
}
```

Compiler	GFLOPs		Efficiency		MFLOPs/US$	
	1K	10K	1K	10K	1K	10K
gcc (5.1)	1.59	1.67	16.6%	17.4%	3	2
Intel	2.70	5.71	28.1%	59.5%	2	5

N-Body – OpenMP and MPI

```
#pragma omp parallel for private(j, r, distSqr,
        distSixth, invDistCube, s, tot_force_i, dv)
  for (i=0; i<N; i++) {
    tot_force_i[X] = tot_force_i[Y] = tot_force_i[Z] = 0.0;
    for (j=0; j<N; j++) {
      if (j==i) continue;
      //20 floating point operations
      r[X] = oldbodies[j].pos[X] - oldbodies[i].pos[X];
      // analogous for r[Y] and r[Z]
      distSqr = r[X]*r[X] + r[Y]*r[Y] + r[Z]*r[Z] + EPSILON2;
      distSixth = distSqr * distSqr * distSqr;
      invDistCube = 1.0/sqrtf(distSixth);
      s = oldbodies[j].mass * invDistCube;
      tot_fo
      // ana
    }
    //24 flop
    dv[X] = d
    newbodies
    newbodies
    // analog
  }
}
```

```
bnum = N % numprocs;
bstart=(N/numprocs*id)      + ((id>=bnum) ? bnum   : id);
bstop =(N/numprocs*(id+1)) + ((id>=bnum) ? bnum-1 : id);
bnum = bstop-bstart+1;

MPI_Barrier(MPI_COMM_WORLD); //for timing purpose
for (k=0; k<num_steps; k++) {
  swap(oldbodies,newbodies);
  for (i=bstart; i<=bstop; i++) {  ...  }
  MPI_Allgatherv(MPI_IN_PLACE, 0, MPI_DATATYPE_NULL, newbodies,...);
}
```

N-Body – OpenMP and MPI

- Maximum theoretical performances:
  57.6 GFLOPs (1 CPU – 6 cores)
  115 GFLOPs (2 CPUs – 12 cores)
- Intel compiler only (cost disregarded)
- OpenMP is easier
- Think parallel

Compiler	GFLOPs		Efficiency		MFLOPs/US$	
	1K	10K	1K	10K	1K	10K
OpenMP - 1 cpu	25.49	33.1	44.3%	57.5%	23	30
OpenMP - 2 cpus	35.50	63.24	30.8%	54.9%	32	57
MPI - 1 cpu	27.21	29.12	47.2%	50.6%	25	26
MPI - 2 cpus	50.27	59.93	43.6%	52.0%	46	54

N-Body – CUDA and OpenACC

```
for (k=0; k<timesteps; k++) {
  swap(oldbodies,newbodies);
  for (i=0; i<N; i++) {
    tot_force_i[X] = tot_force_i[Y] = tot_force_i[Z] = 0.0;
    for (j=0; j<N; j++) {
      if (j==i) continue;
      //20 floating point operations
      distSixth = rsqrtf(distSqr);
      invDistCube = distSixth * distSixth * distSixth;
      distSixth = distSqr * distSqr * distSqr;
      invDistCube = 1.0/sqrtf(distSixth);
      s = oldbodies[j].mass * invDistCube;
      tot_force_i[X] += s * r[X];
      // analogous for Y and Z
    }
    //24
    dv[X]
    newbo
    newbo
    // an
  }
}
```

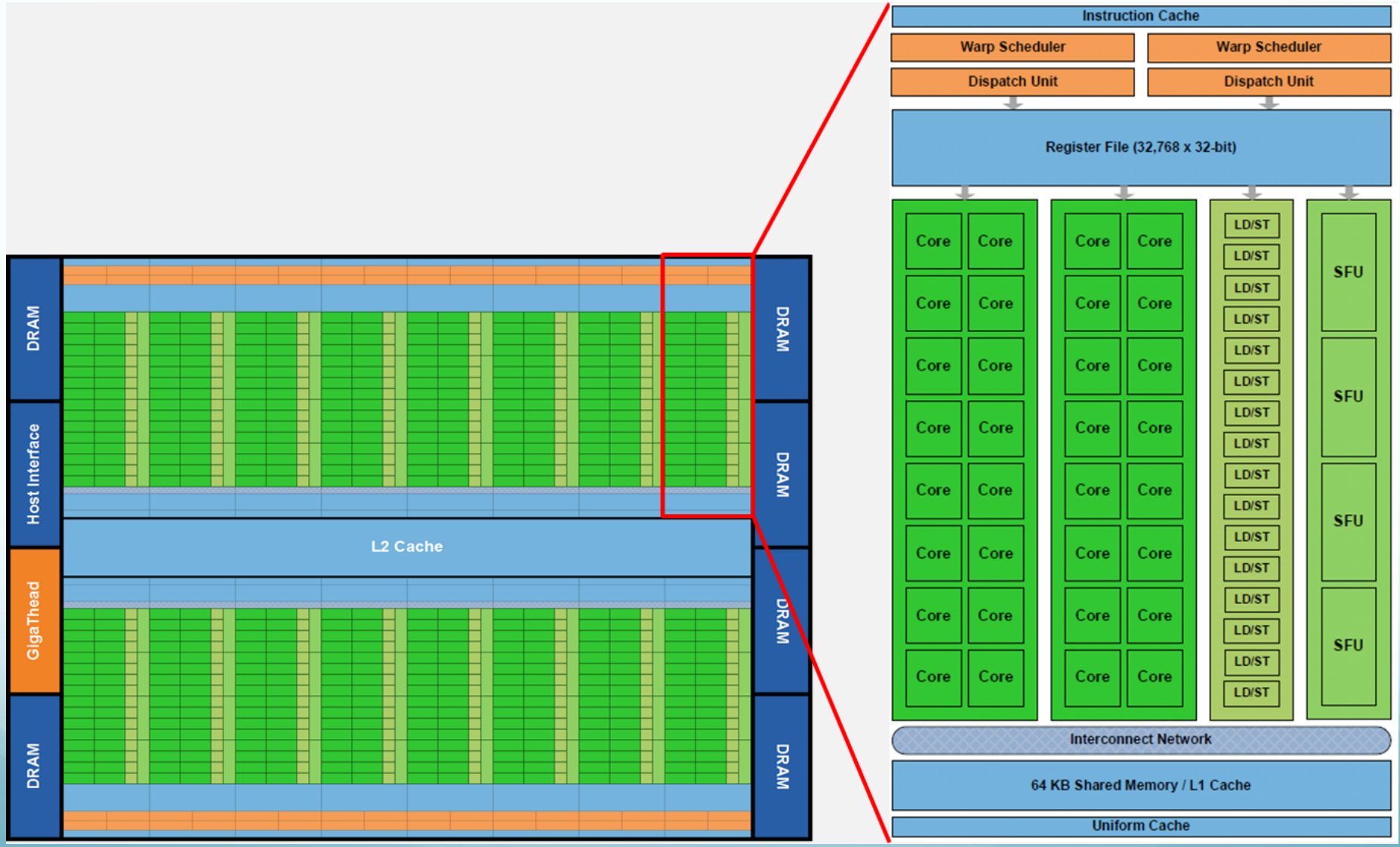
```
#pragma acc data copy(bodies1[0:N], bodies2[0:N])
                 copyin(bodiesvel[0:N], bodiesm[0:N], dt)
for (k=0; k<num_steps; k++) {
  #pragma acc kernels loop independent
  for (i=0; i<num_bodies; i++) {
    tot_force_x = 0.0; ...
    #pragma acc loop independent
                 reduction(+:tot_force_x,tot_force_y,tot_force_z)
```

N-Body – CUDA and OpenACC

- Maximum theoretical performances:
  1581 GFLOPs (1 GPU – 512 cores – 499 US$)
- OpenACC is much much easier
- Performances with and w/o *fastmath* (rsqrtf function), N=10K

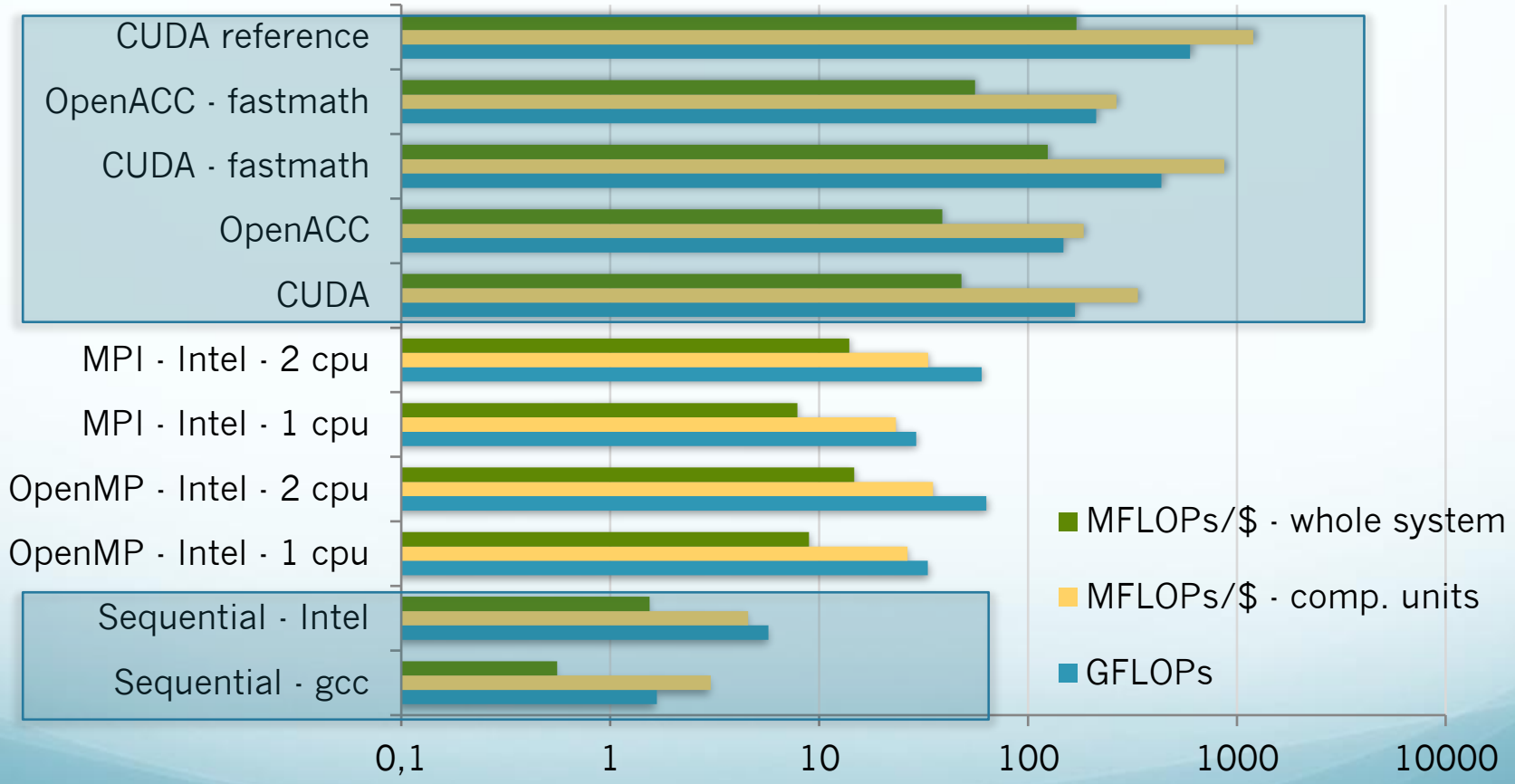
Algorithm	GFLOPs	Efficiency	MFLOPs/US$
CUDA	167.71	10.6%	336
OpenACC	147.57	9.3%	185
CUDA - fastmath	434.46	27.5%	871
OpenACC - fastmath	211.80	13.4%	265
CUDA reference	597.11	37.8%	1197

N-Body – CUDA and OpenACC

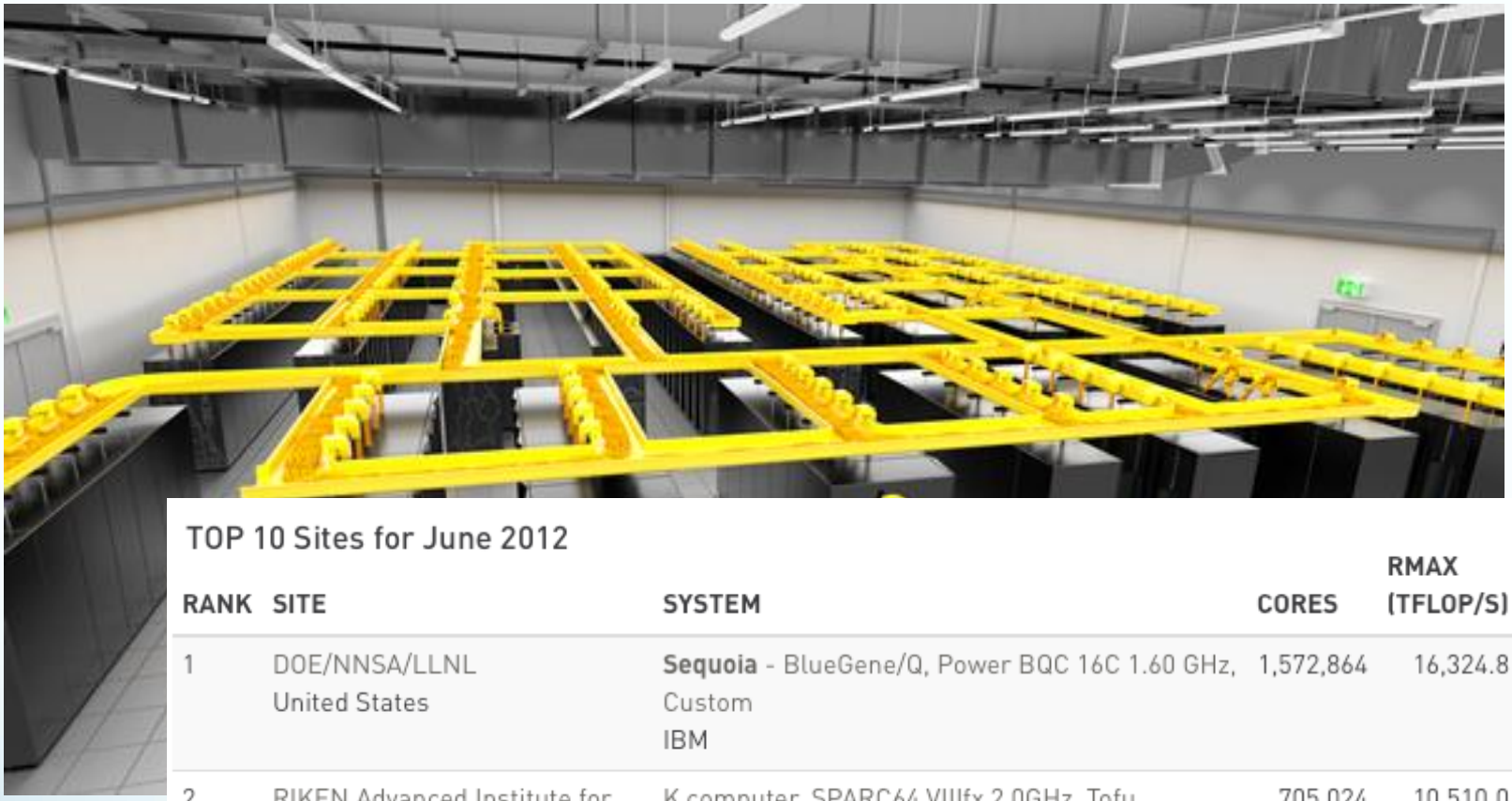


Performance/Price (10K)

Programmers and tools make the difference!



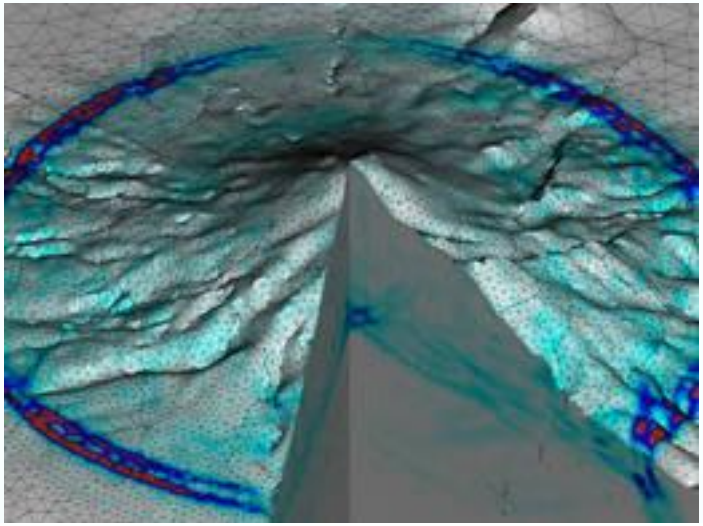
A real-world example



TOP 10 Sites for June 2012

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	16,324.8	20,132.7	7,890
2	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
3	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,162.4	10,066.3	3,945
4	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147,456	2,897.0	3,185.1	3,423

SeisSol Earthquake simulation SW



The extensive optimization and the complete parallelization of the 70,000 lines of SeisSol code results in a peak performance of up to 1.42 petaflops.

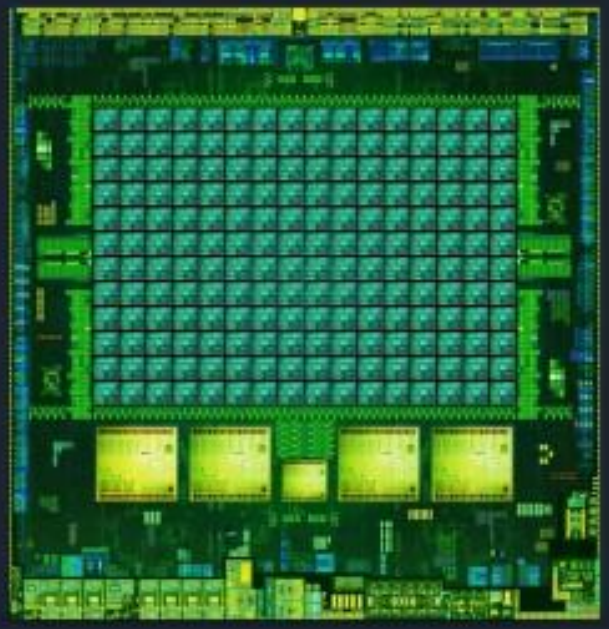
This corresponds to 44.5 percent of Super MUC's theoretically available capacity

Name	MPI	# cores	Description	TFlop/s/island	TFlop/s max
Linpack	IBM	128000	TOP500	161	2560
Vertex	IBM	128000	Plasma Physics	15	245
GROMACS	IBM, Intel	64000	Molecular Modelling	40	110
Seissol	IBM	64000	Geophysics	31	95
waLBerla	IBM	128000	Lattice Boltzmann	5.6	90
LAMMPS	IBM	128000	Molecular Modelling	5.6	90
APES	IBM	64000	CFD	6	47
BQCD	Intel	128000	Quantum Physics	10	27

Aspects we will not talk about

- We are not green: 1 week of molecular dynamics simulation on 512 cores = 3200 kWh, corresponding to
  - 1600 CO2 kg
  - 340 € energy bill
  - 13000 km by car

- A national supercomputing facility has a yearly CO2 footprint comparable to a takeoff of SATURN V



NVIDIA TEGRA K1

CPU: 2.2GHz 4+1 A1

GPU: 192 核 Kepler

CPU 4+1 A15

GPU 192 Kepler

TEGRA X1 MAXWELL GPU

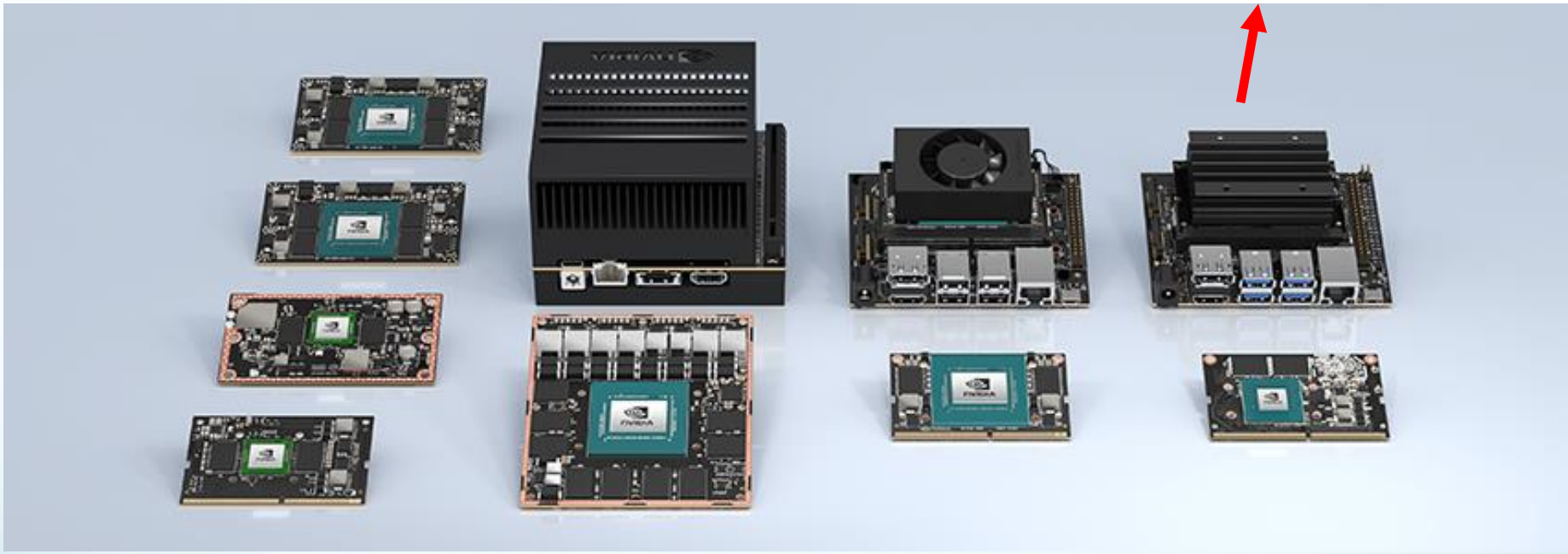
- 2x performance vs Tegra K1
- 2x perf/watt vs Tegra K1
- 2 SM
- 256 CUDA Cores

	Tesla K40 + CPU	Nvidia Tegra K1	
Single Precision Peak	4.2 TeraFlops	326 GFlops	13
Single Precision SGEMM	3.8 TeraFlops	290 GFlops	
Memory	12GB @ 288GB/s	2GB @ 14.9GB/s	
Power (CPU + GPU)	~ 385Watt	<11Watts	
Performance Per Watt	10SP GFlops Per Watt	26SP GFlops Per Watt	35
	$ 4,000	$ 200	20

Today

VIEW TECHNICAL SPECIFICATIONS >

GPU	128-core NVIDIA Maxwell™
CPU	Quad-core ARM® A57 @ 1.43 GHz
Memory	2 GB 64-bit LPDDR4 25.6 GB/s



Nvidia Jetson Dev Kit: 59 $, 472 GFLOPs FP16, 10 W, 45x70 mm

https://developer.nvidia.com/embedded/jetson-modules

Example: Functional Magnetic Resonance Imaging

- The science on that is pretty well established. They knew how to take the data that was coming from the MRI, and they could compute on it and create a model of what's going on inside the brain. But in 2012, when we started the project, they estimated it would take 44 years on their cluster
- they parallelized their code and saw huge increases in performance
- But they also looked at it algorithmically with machine learning and AI,
- They put it all together and ended up with a 10,000X increase in performance. They went from something requiring a supercomputing project at a national lab to something that could be done clinically inside a hospital in a couple of minutes.

https://www.princeton.edu/news/2017/02/23/princeton-intel-collaboration-breaks-new-ground-studies-brain

https://www.hpcwire.com/2017/06/08/code-modernization-bringing-codes-parallel-age/

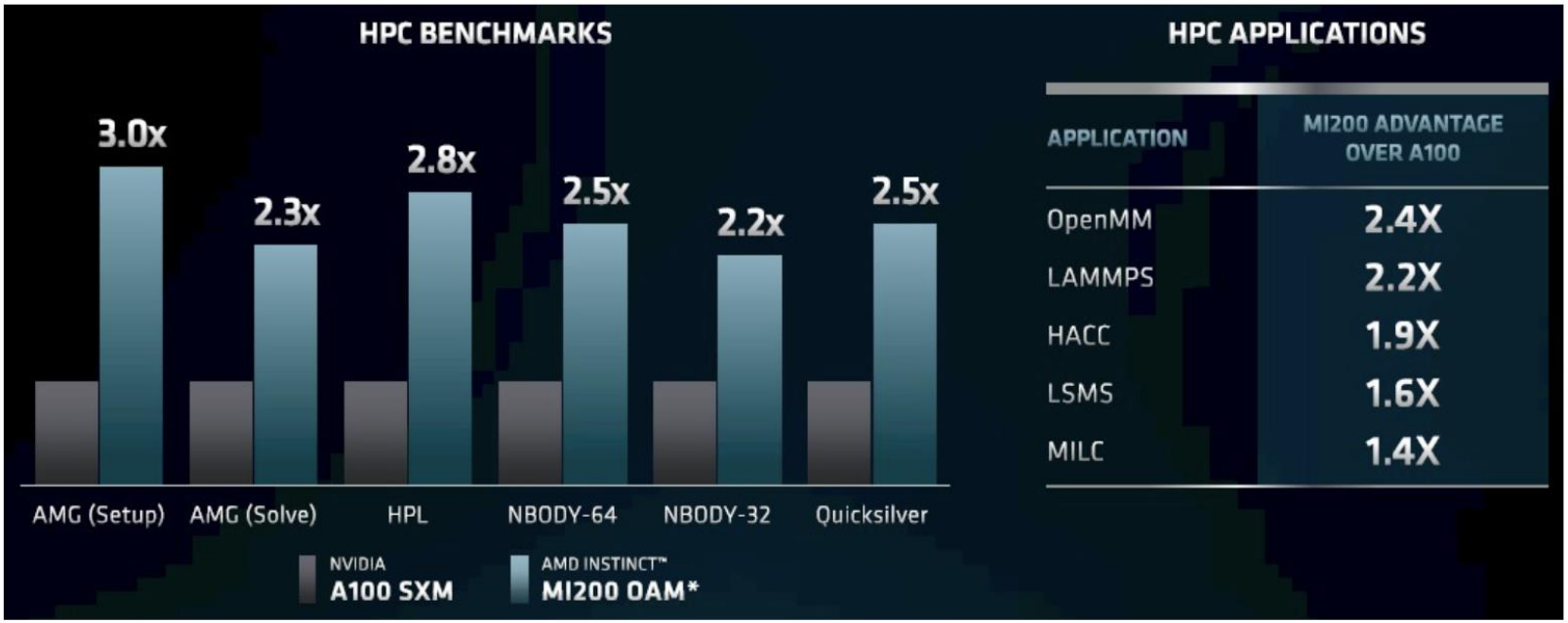
Again on SW

https://www.nextplatform.com/2021/12/06/stacking-up-amd-mi200-versus-nvidia-a100-compute-engines/?mc_cid=11aeb90192&mc_eid=e50c89e962

PEAK PERFORMANCE	A100	MI200*	INSTINCT™ ADVANTAGE
FP64 VECTOR	9.7 TF	47.9 TF	**4.9X**
FP32 VECTOR	19.5 TF	47.9 TF	**2.5X**
FP64 MATRIX	19.5 TF	95.7 TF	**4.9X**
FP32 MATRIX	N/A	95.7 TF	N/A
FP16, BF16 MATRIX	312 TF	383 TF	**1.2X**
MEMORY SIZE	80 GB	128 GB	**1.6X**
MEMORY BANDWIDTH	2.0 TB/s	3.2 TB/s	**1.6X**

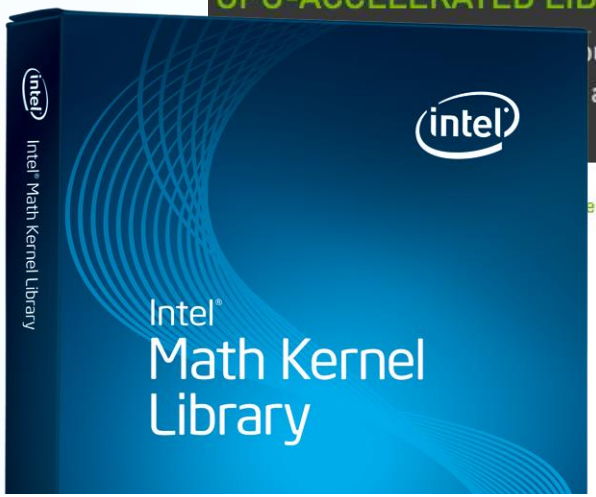
Again on SW

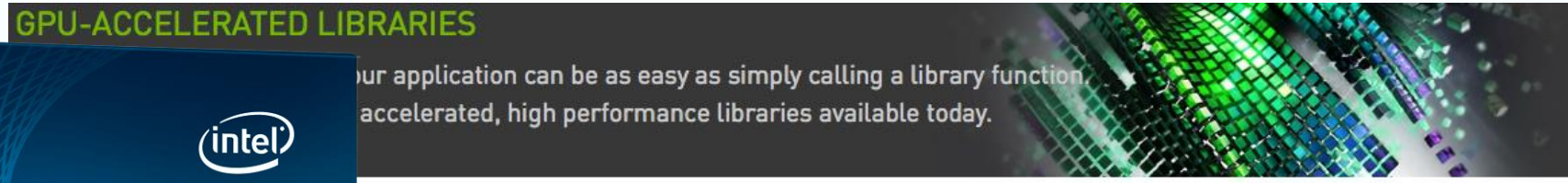
What matters, of course, is the performance of the MI200 versus the A100 on HPC benchmarks and real HPC applications. McCredie offered up this chart as food for thought:



As you can see, the performance numbers are definitely in favor of the Aldebaran GPUs, each half of which has a little bit more performance of the whole A100 on common HPC benchmarks and a little less than that on the HPC applications shown on the right part of this table. The gap is not as big as raw feeds and speeds show, and we think that has to do with the maturity of the compilers and math libraries in the ROCm stack from AMD for its own GPUs versus the CUDA stack from Nvidia for its own GPUs. There is $100 million in non-recurring engineering funds in the Frontier system alone to try to close some of that ROCm-CUDA gap.

Libraries





em > GPU-Accelerated Libraries

cuDNN

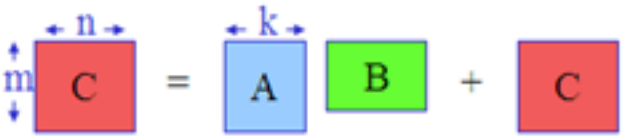
NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks, it is designed to be integrated into higher-level machine

cuFFT

NVIDIA CUDA Fast Fourier Transform Library (cuFFT) provides a simple interface for computing FFTs up to 10x faster, without having to develop

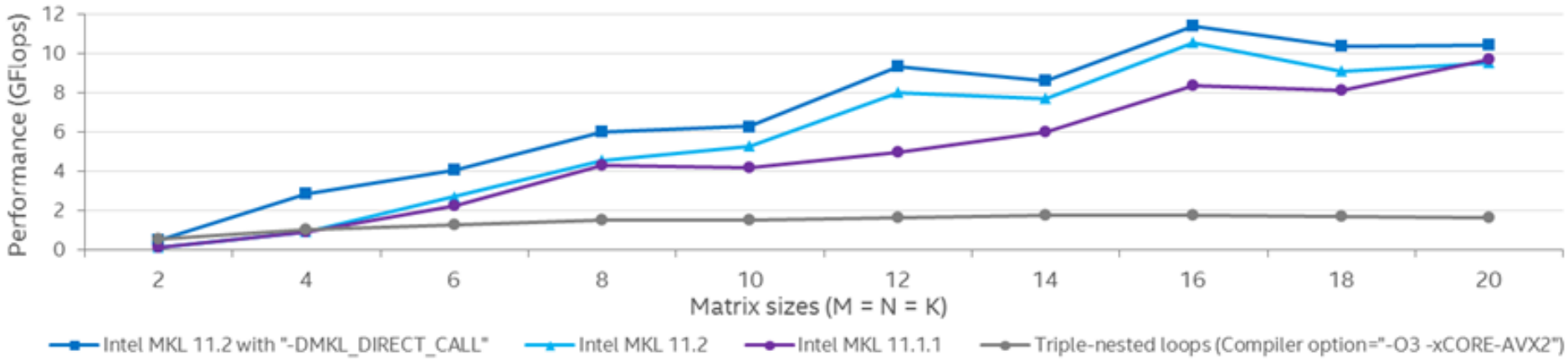
cuBLAS-XT

cuBLAS-XT is a set of routines which accelerate Level 3 BLAS (Basic Linear Algebra Subroutine) calls by spreading work across more than one



Faster Small Matrix Multiplication using Intel® MKL

For 4x4 to 20x20 matrices, S/DGEMM, Single thread, Intel® Xeon® Processor E5-2697v3



Conclusions

The efficient exploitation of current heterogeneous HPC solutions require good understanding of HW and SW features (architectures, instructions sets, sdk, ...)

- Not only HW
- Skilled developers
- State-of-the-art software libraries and programming tools.

Good tools and developers are worth the money

The course aims at presenting the basics to let you became good HPC developers!