COMP2000

An Introduction to Parallel Programming

University of Plymouth
School of Engineering, Computing and Mathematics

Dr. Vasilios Kelefouras

Email: v.kelefouras@plymouth.ac.uk
Website: https://www.plymouth.ac.uk/staff/vasilios-kelefouras

Welcome

What is Parallel Computing?

- Most of the programmers nowadays are familiar with ***serial computation***
  - Code is developed in a high level language
  - Code Instructions are executed one after another
  - 1 CPU runs the code
- ***Parallel computing*** is a type of computation where many calculations or the execution of processes are carried out simultaneously
  - Code or task must be broken down into multiple smaller pieces
  - Those pieces are executed at the same time, each on a different hardware resource

Applications – Data Explosion

20 hours
of video
uploaded to YouTube
every minute

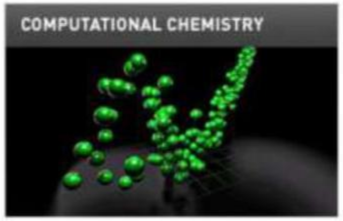
Approximately
9 billion
video files owned are
high-definition

50 million +
digital media files
added to personal content libraries
every day

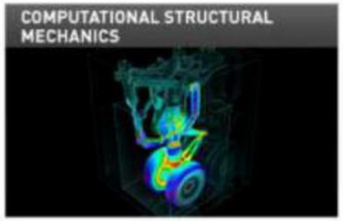
1000
images
are uploaded to Facebook
every second

Applications – Scientific Computing

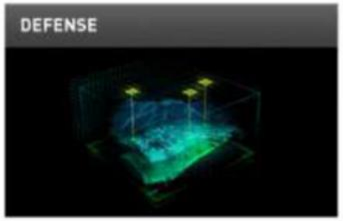
BIOINFORMATICS

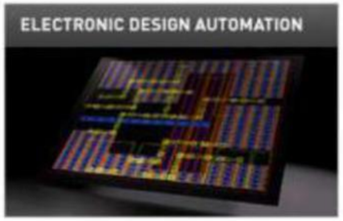

COMPUTATIONAL CHEMISTRY

COMPUTATIONAL FLUID DYNAMICS


COMPUTATIONAL STRUCTURAL MECHANICS

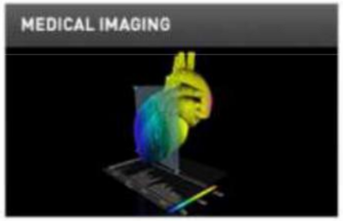
DATA SCIENCE


DEFENSE

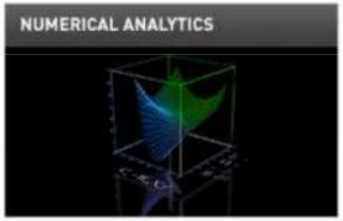

ELECTRONIC DESIGN AUTOMATION

COMPUTATIONAL FINANCE

IMAGING & COMPUTER VISION


MEDICAL IMAGING

WEATHER AND CLIMATE


NUMERICAL ANALYTICS

- Example fields
  - Chemistry
  - Life sciences
  - Bioinformatics
  - Astrophysics
  - Finance
  - Medical imaging
  - Natural language processing
  - Social sciences
  - Weather and climate
  - Computational fluid dynamics
  - Machine learning
  - etc…

High Performance Computing (HPC) market

- The HPC Market Map (next slide) demonstrates the **rapidly growing importance of HPC to industrial competitiveness of both the UK and Europe**
- High performance software is critical in modern computer systems ranging from small embedded devices to big supercomputers and datacenters
- Companies need employers to write efficient software
- Engineers with that knowledge are desirable in Industry

Special Interest Group

e-infrastructure

Introduction

e-Infrastructure was described in a recent report from the Department for Business, Innovation & Skills as the "ecosystem for innovation", underpinning a wide range of future technologies. e-Infrastructure enables computational science & engineering methods and the potential contribution to the UK economy of e-Infrastructure technologies is enormous.

Although higher education currently dominates the UK's HPC resources, enterprises of all sizes are recognising the necessity for HPC in the commercial arena. Few companies are able to make the required investment on their own. Therefore, a need has developed for e-Infrastructure to provide the necessary resources for HPC service providers to meet the rapidly growing demands of UK enterprises.

Trends & Numbers

$10.3 Billion	Worldwide revenue value of the HPC Servers market in 2013 [3]
30%	European market share of total worldwide HPC revenues in 2013 [3]
$430 Million	Value of German Supercomputer segment in 2011or 43% of total HPC revenues [4]
$260 Million	Value of French Supercomputer segment in 2011or 44% of total HPC revenues [4]
83%	of HPC application software used in Europe is indigineous and 66% is based on IP generated in Europe [2]
<5%	European share of HPC system vendors in the global HPC system market [2]
€100 Million	Amount committed by PRACE hosting partners (Germany, France, Italy & Spain) to petascale computing cycles between 2010-2014 [1]

Sources

[1] e-Infrastructure Leadership Council 2012
[2] ETP4HPC Strategic Research Agenda 2013
[3] IDC Worldwide High Performance Technical Server Overview 2014
[4] IDC Report on HPC market Trends 2013

Prepared by
FirstPartner
call us on +44 (0) 870 874 8700
find us at www.firstpartner.net

High Performance Computing (HPC)

Verticals Served

Energy

- Advanced combustion modelling
- Plasma physics

Environmental & Geosciences

- Weather forecasting
- Climate modeling

Defence & Security

- Military vehicle design
- Processing of intelligence

Astrophysics

- Astrophysical simulation
- Telescope data analysis

Life Sciences

- DNA sequence analysis
- Proteomics and toxicology

Chemistry & Materials Science

- Nanoscale material science
- Quantum chemistry

Automotive / Engineering

- Simulated prototype testing

Economics & Finance

- Macro/micro economic modelling
- Derivatives trading

Supply Chain

- Planning and optimisation of distribution systems

Creative & Media

- Computer graphic generation

HPC Users

Research

Public and privately funded research bodies and institutions, typically in nuclear, meteorological, space sciences and biomedical industries

cea, ascs, AWE, ECMWF, JÜLICH Forschungszentrum

Industry & Business

Large multinational enterprises with the resources to invest in private MPC facilities. Includes banks, automotive manufacturers, aerospace, FMCG, energy, oil and gas industries

produban, AIRBUS, Unilever, eni, e.on, TOTAL, EDF Energy

SME

Small and medium-sized enterprises in scientific and data-intensive industries, e.g. healthcare, engineering

engys, CALON Cardio, IES, CVIS, BHR Group Experts in Fluid Engineering

Educational

Educational institutions, e.g. universities and technical colleges, using HPC facilities to conduct research involving analysis of large datasets, e.g. brain scan data

KIT, Technische Universität Darmstadt, LRZ, BSC

Government

Governmental departments and agencies, e.g. Department of Defence, as well as other health and regulatory agencies. Governments are key direct and indirect investors in HPC technology

GCHQ, DGSE, Bundesnachrichtendienst

HPC Usage

Big Data Workloads

Manipulating vast datasets that are too large to be processed using commercial software and hardware in short time frames

Simulation & Optimisation

Simulating real-world phenomena to gain deeper understanding or minimise risk, e.g. financial and molecular simulation

Predictive Behaviour

Processing data to create forecast models or predict outcomes, e.g. weather forecasting

Data Informatics

Computationally or numerically intensive processing of vast or complex datasets to aid decision making in a variety of fields

Classification & Textual Analysis

Document or file classification, e.g. music, as well as analyse text, e.g. sentiment analysis

Graphics Generation

Creating computer-generated animation for the Media industry and modelling output representation

HPC Access

Academic

International connectivity: Peering Points ↔ Internet Exchanges ↔ GÉANT

NREN - National connectivity: janet, SURFNET, RENATER, Consortium GARR, DFN, RedIRIS, NORDUnet

HPC Resources

Owners
- Commercial
- Public Research
- Academic
- Defense & Security

Access Models
- Pay By Use
- Contracted Use
- Captive In House Use

Commercial

Physical Transfer: amazon web services, FedEx Express, DHL

Private High Capacity Service: Virgin Media Business, sohonet

Data Centres: LDEX Data Centres, CITY-lifeline

MPC Access Node
- Peering points to NREN
- Direct connection

Point-To-Point Dedicated Connection: BT wholesale, Viatel, eunetworks

Internet Exchanges: France-IX, DE-CIX, AMS-IX, LONAP, NYIIX, linxdatacenter, MIX

Complimentary HPC Approaches

Grid

Distributed computing resources dynamically pooled and built on-top-of high performance networks.

UKngi – UK Grid Initiative; EGI – European Grid Infrastructure; LCG – Worldwide LHC Computing Grid

On-Premises

Very large clusters of Servers configured for parallel computing
- Workgroup
- Divisional
- Departmental

Cloud

Cloud services are widely used for application-intensive purposes by commercial organisations.

EC2 HPC amazon web services, Google, Microsoft Azure

HPC Systems

HPC Software

Application & Development Tools

Vertical sector specific applications & general purpose debugging & optimisation toolkits.

RogueWave Software, NVIDIA CUDA, scilab enterprises, SILKAN, DISTENE, allinea, Bull

Middleware

Software to manage & optimise multiprocessor clusters to achieve maximum computational performance.

OpenIS, NICE, SC, SysFera, Bull, Bright Computing, Adaptive Computing, ParTec

European R&D Resources

Vendors

Commercial HPC system vendors who also undertake original HPC research in Europe.

hp, IBM, CRAY, Bull

Institutional

Academic & grant aided centres of research excellence.

Queen's University Belfast, BSC, CIMNE, Fraunhofer, JÜLICH

EESI: Project co-funded by European Commission to build a European vision and roadmap to address the challenges of new generation multi-Petaflop & Exaflop systems.

HPC Hardware

Complete HPC Hardware Systems

Top 4 Global HPC System Vendors by Annual Revenue in 2013

IBM	$2.8 Bn	DELL	$1.5 Bn
hp	$3.3 Bn	CRAY	$0.4 Bn

European MPC System Vendors: bullx scs, EUROTECH, E4 Computer Engineering

Memory & Storage

New non-volatile forms of memory technology are emerging which can significantly improve energy efficiency and enable the necessary massive throughout and highly parallel access.

CRAY SONEXION, DELL, DataDirect Networks, Seagate, HITACHI, TERASCALA, Bull lustre

Local Interconnects

Communication bandwidth scaling in line with other system components to minimise latency and energy consumption.

molex, FCI, numascale, Finisar, REFLEX PHOTONICS, FUJITSU

Core Processors

High numbers of clusters of cores are used to achieve high throughput in compute-intensive tasks, such as finite element analysis.

ARM, clustervision, KALRAY, NVIDIA, intel

System Resilience

The rapidly increasing number of system components requires improving resilience of individual system components to prevent scaling failure.

MAXELER Technologies, Bull

HPC Scale

Advances in energy efficiency and power over the next decade, driven by iltiatives such as DEEP and Mont Blanc, are key to achieving an exponential growth in computer processing capacity.

Peta: *One quadrillion floating point operations per*
Energy & Power Initatives: DEEP, MONT BLANC
Exa: *Exponential increase in the number of*

Existing European Resources

Country		Academic	Research	Industry	Total CPU Cores
🇧🇪	BE	1		1	19,944
🇩🇰	DK			1	15,672
🇫🇮	FI	2			20,976
🇫🇷	FR	5	7	14	776,468
🇩🇪	DE	10	6	6	1,113,012
🇮🇹	IT	1	2	2	268,400
🇳🇱	NL	2		3	62,160
🇳🇴	NO	3			54,400
🇪🇸	ES	1	1		65,280
🇸🇪	SE	2		1	73,248
🇨🇭	CH	2	4		263,888
🇬🇧	UK	5	15	10	925,152

Source: TOP500.org - 06/2014

HPC Initiatives

UK

UK e-Infrastructure Initiative

HPC Wales, Supercomputing Scotland

SME Initiatives: A number of initiatives in the UK and Europe include SME access to HPC facilities in their scope.

SICOS, HLRS, Initiative HPC-PME, Ter@tec

Network of Excellence on High Performance and Embedded Architecture & Compilation: HiPEAC

NESUS: Network for Ultrascale Computing

PRACE: Academic Partnership for Advanced Computing in Europe

ETP4HPC: Industry European Technology Platform 4 HPC

Horizon 2020

EESI: European Exascale Software Initiative

Europe

Fastest Supercomputers (Jan. 2022) (1)

- Fugaku (Japan)
  - 7.3 million cores, 415 peta (10^{15}) FLOPs
- Summit (US)
  - 148 peta FLOPs
- Sierra (US)
  - 1.5 million cores, 94 peta FLOPs
- Sunway TaihuLight (China)
  - 10.6 million cores, 93 peta FLOPs
- Tianhe-2 (China)
  - 5 million cores, 61peta FLOPs

Fastest Supercomputers (Jan. 2022) (2)

- These supercomputers are drawing 10-40 megawatts of power
  - This is enough for an entire city of 40.000 people
  - Each megawatt costs about 1 million dollars per year…
  - We need fast computers but consuming low energy consumption
  - **In Plymouth University there is a supercomputer too**

Why the current trend is towards more and more CPU cores?

- Since 2006, PCs, laptops and servers support more and more cores, why?
  - **Question:** What changed back in 2006?
  - **Answer:** The CPU frequency has ceased to grow… but why?
  - The CPU frequency could not be further increased as the power consumption becomes too high

ISSCC 2001, Keynote

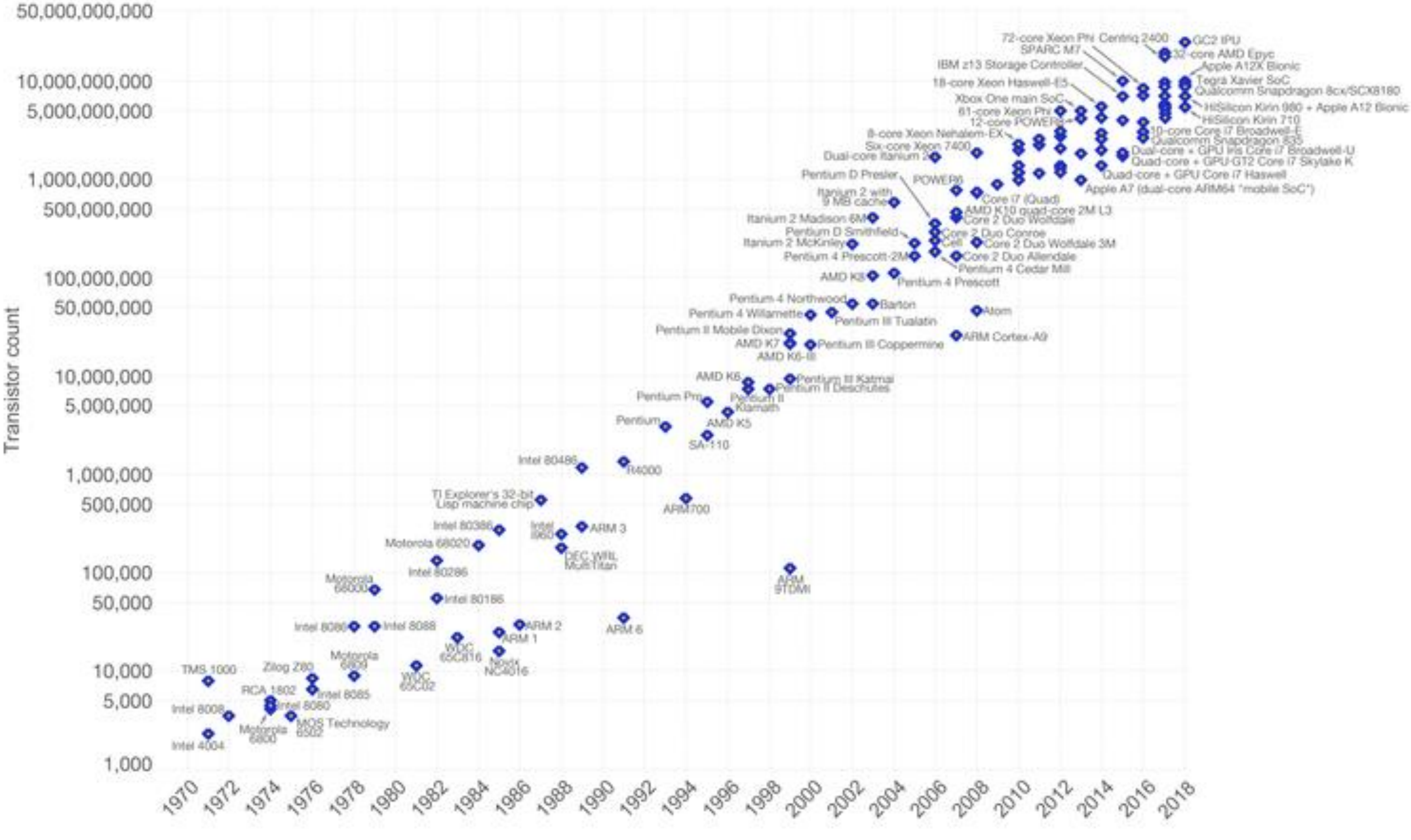
Patrick P. Gelsinger
Senior Vice President
General Manager
Digital Enterprise Group
INTEL CORP.

“Ten years from now, microprocessors will run at 10GHz to 30GHz and be capable of processing 1 trillion operations per second -- about the same number of calculations that the world's fastest supercomputer can perform now.

“*Unfortunately, if nothing changes these chips will produce as much heat, for their proportional size, as a nuclear reactor. . . .*”

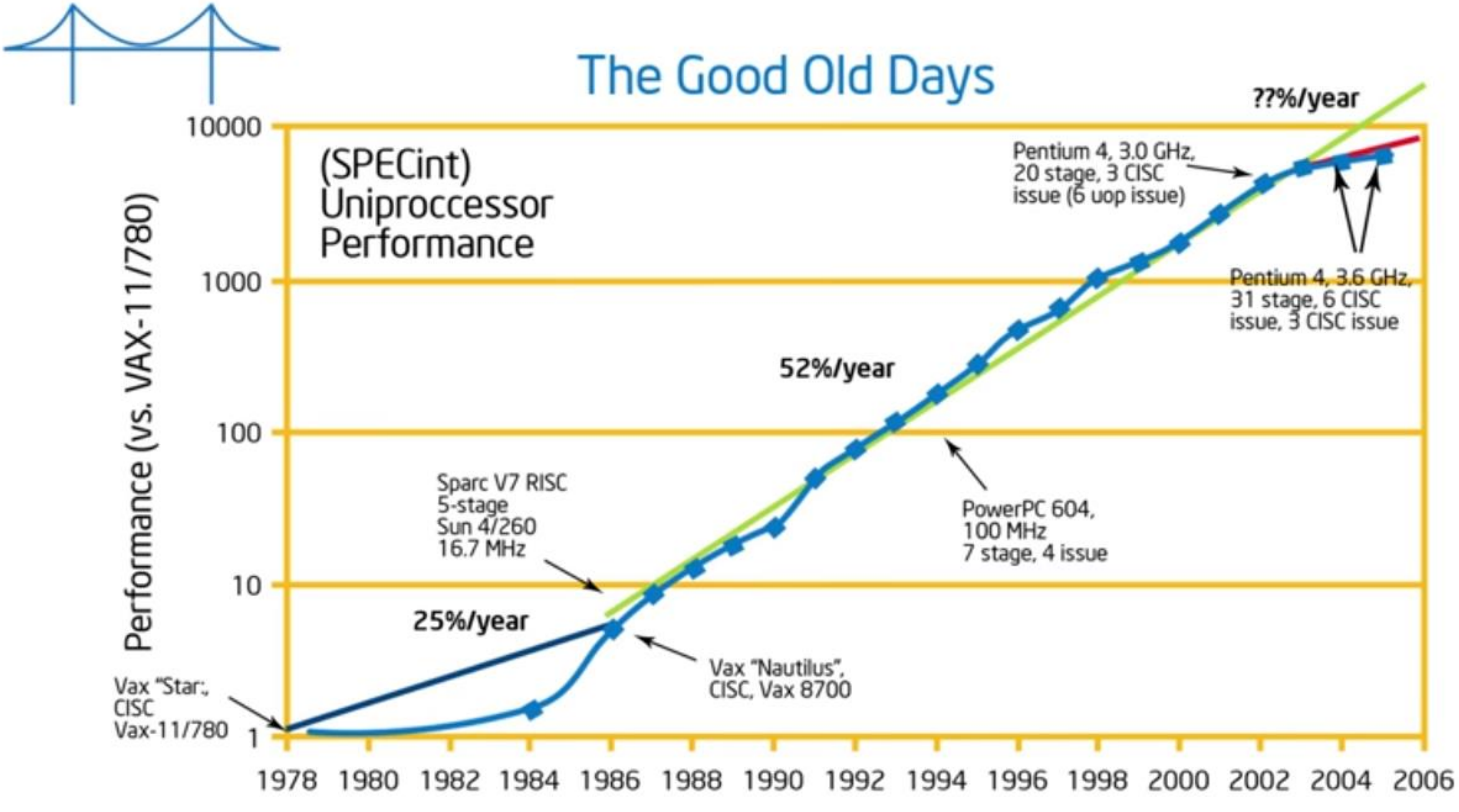
Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Performance **used to** increase according to the number of transistors (1)



The Good Old Days

(SPECint) Uniproccessor Performance

Vax "Star:, CISC Vax-11/780

Sparc V7 RISC 5-stage Sun 4/260 16.7 MHz

Vax "Nautilus", CISC, Vax 8700

PowerPC 604, 100 MHz 7 stage, 4 issue

Pentium 4, 3.0 GHz, 20 stage, 3 CISC issue (6 uop issue)

Pentium 4, 3.6 GHz, 31 stage, 6 CISC issue, 3 CISC issue

From Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 4th edition, Sept. 15, 2006

Performance **used to** increase according to the number of transistors (2)

- The fact that performance used to increase by increasing the number of transistors, trained people to expect that performance comes from the hardware
- Programmers used to write software without thinking about performance
- They counted on the hardware to do the work - **Optimization was left to the HW**
- This model used to work fine…but…back in 2006, something changed…
  - ***The Power Wall problem***

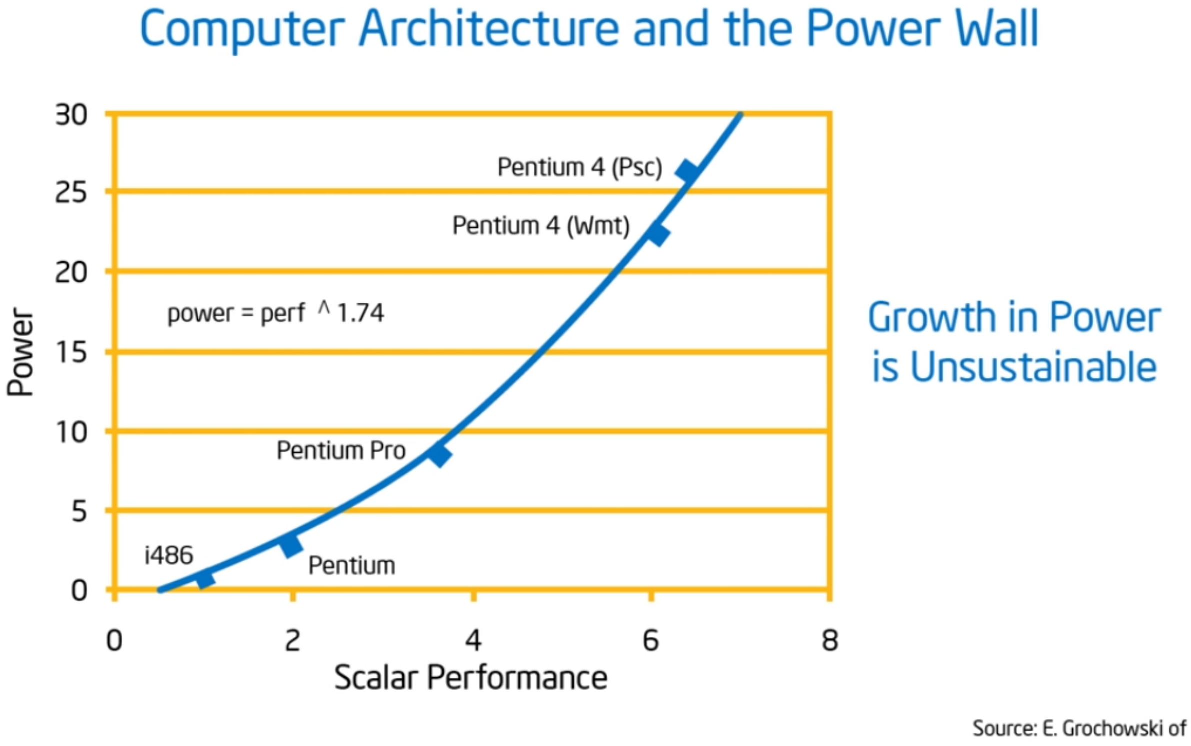
Do not expect your serial program to run faster on new processors

Performance **used to** increase according to the number of transistors (3)

- Power Wall Problem
  - The CPU design goal for the late 1990's and early 2000's was to increase the CPU frequency.
    - This was a way to improve system performance
    - This was done by adding more transistors to a smaller chip.
    - However, this increased the power dissipation of the CPU chip beyond the capacity of inexpensive cooling techniques.
    - The last years the ***CPU frequency has ceased to grow***

Performance **used to** increase according to the number of transistors (4)

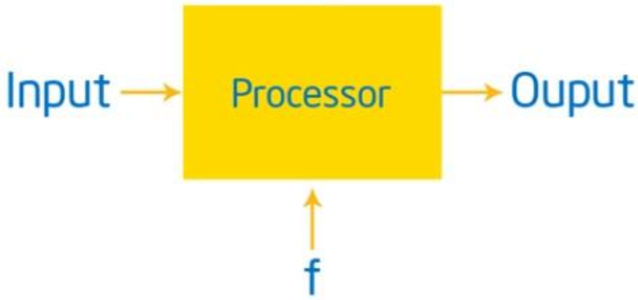
- **By increasing performance, power consumption increases even more** (next slide)
- This is not sustainable
- What to do? ***The solution is Parallel hardware architectures***



The solution to the Power Wall Problem (1)

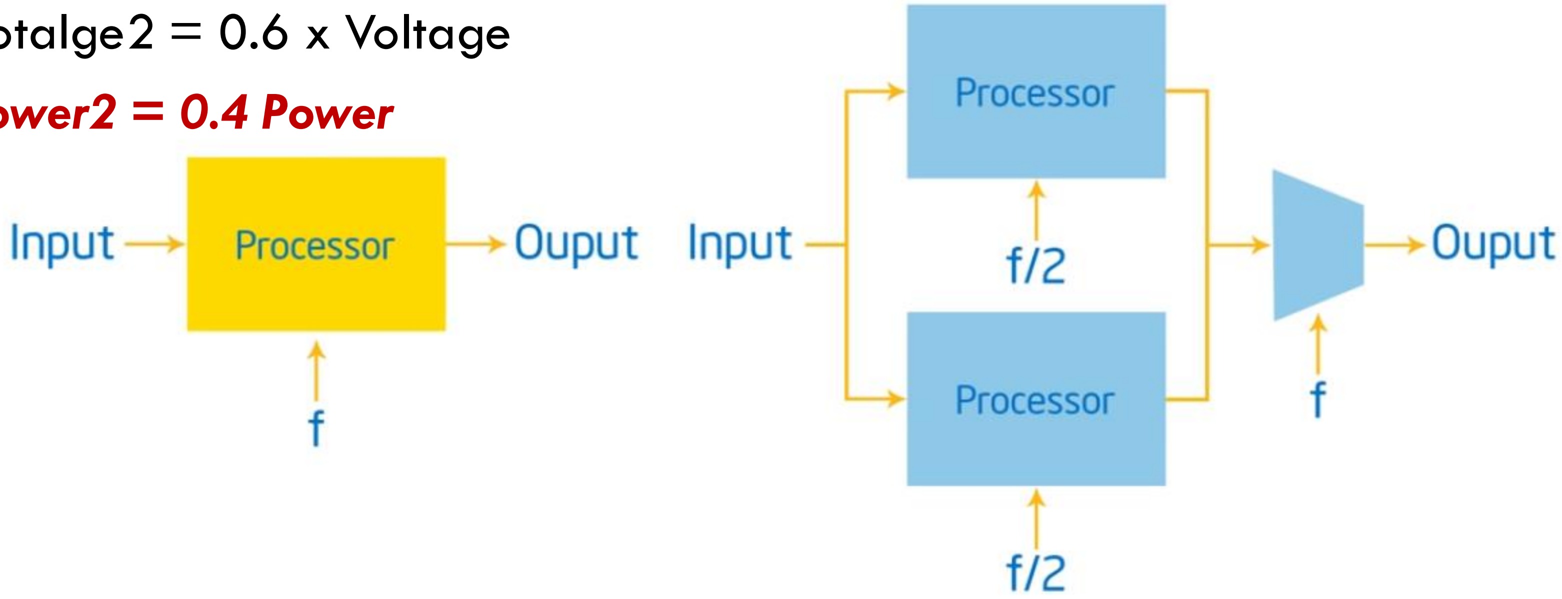
- The power of a processor is given by

Power=Capacitance x Voltage x Frequency^2



The solution to the Power Wall Problem (2)

- The power of a processor is given by **Power=Capacitance x Voltage x Frequency^{2}**
- By using two processors inside the same chip, with half the frequency each, then:
  - Capacitance2 = 2.2 x Capacitance
  - Frequency2 = F/2
  - Votalge2 = 0.6 x Voltage
  - ***Power2 = 0.4 Power***



Parallel computing gives us the ability to give the same performance with lower power

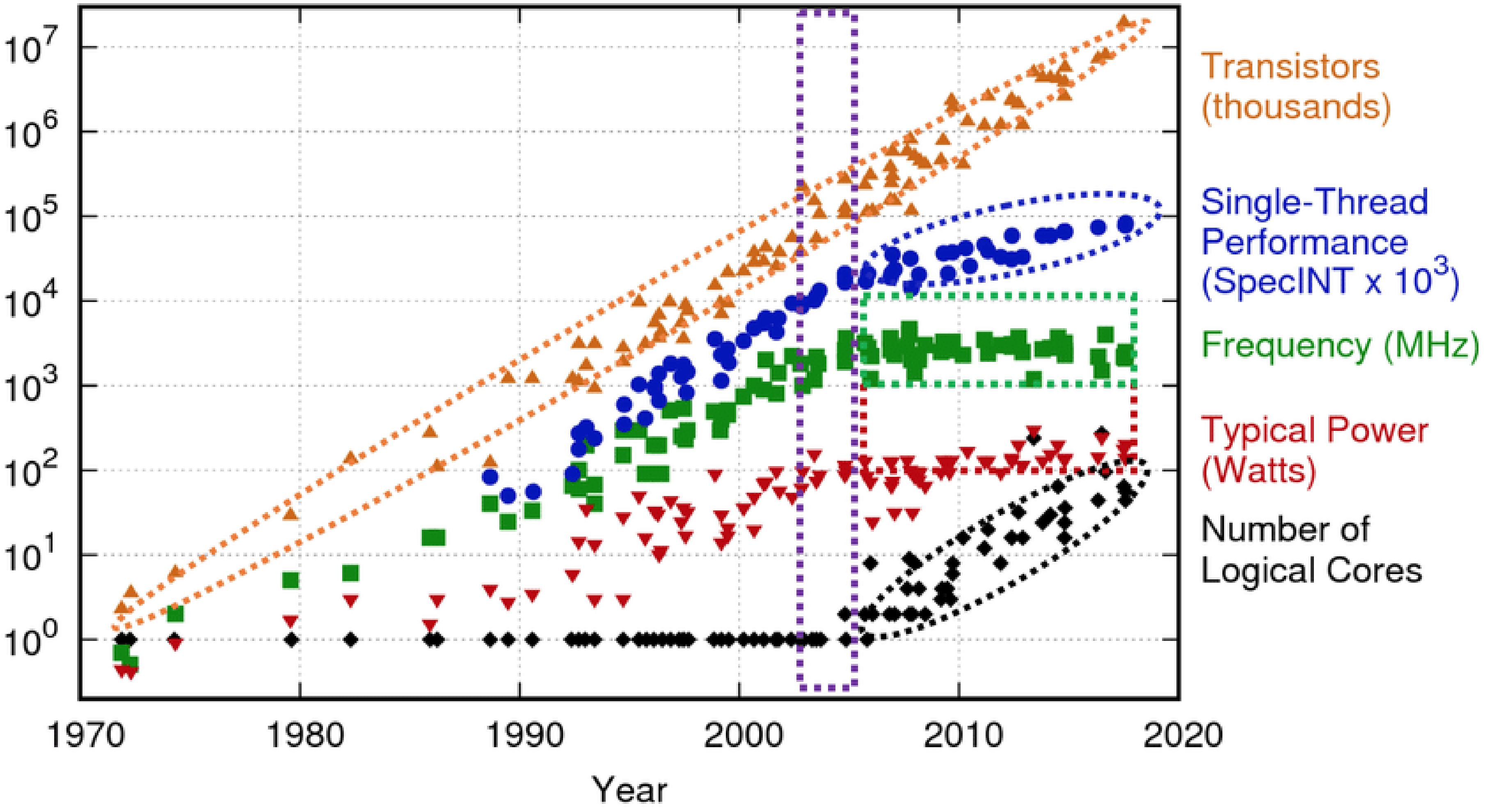
Chandrakasan, A.P.; Potkonjak, M.; Mehra, R.; Rabaey, J.; Brodersen, R.W., "Optimizing power using transformations," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,, vol.14, no.1, pp.12-31, Jan 1995

Source:
Vishwani Agrawal

The Era of Parallel Computing is here

- Nowadays, performance comes from the software
- There are no smart-enough tools to efficiently parallelize serial software on the parallel hardware
- Free lunch is over…
- We must learn how to write parallel applications…

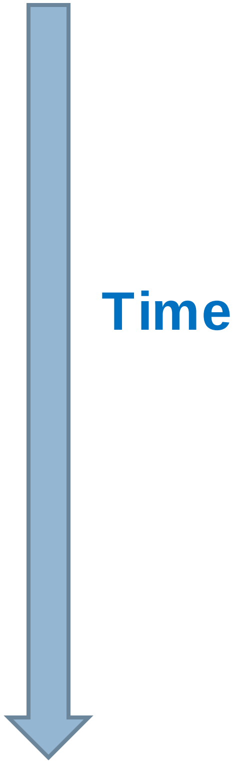
Hardware Architecture Trends



Taken from https://www.researchgate.net/publication/336577121_BACKUS_Comprehensive_High-Performance_Research_Software_Engineering_Approach_for_Simulations_in_Supercomputing_Systems

Hardware Evolution

- Scalar Processors
- Pipelined Processors
- Superscalar Processors
- Out of order Processors
- Vectorization
- Hyper-Threading
- Multicore Processors
- Manycore Processors
- Heterogeneous systems
  - with more cores, more threads, wider vectors



Heterogeneous computing (1)

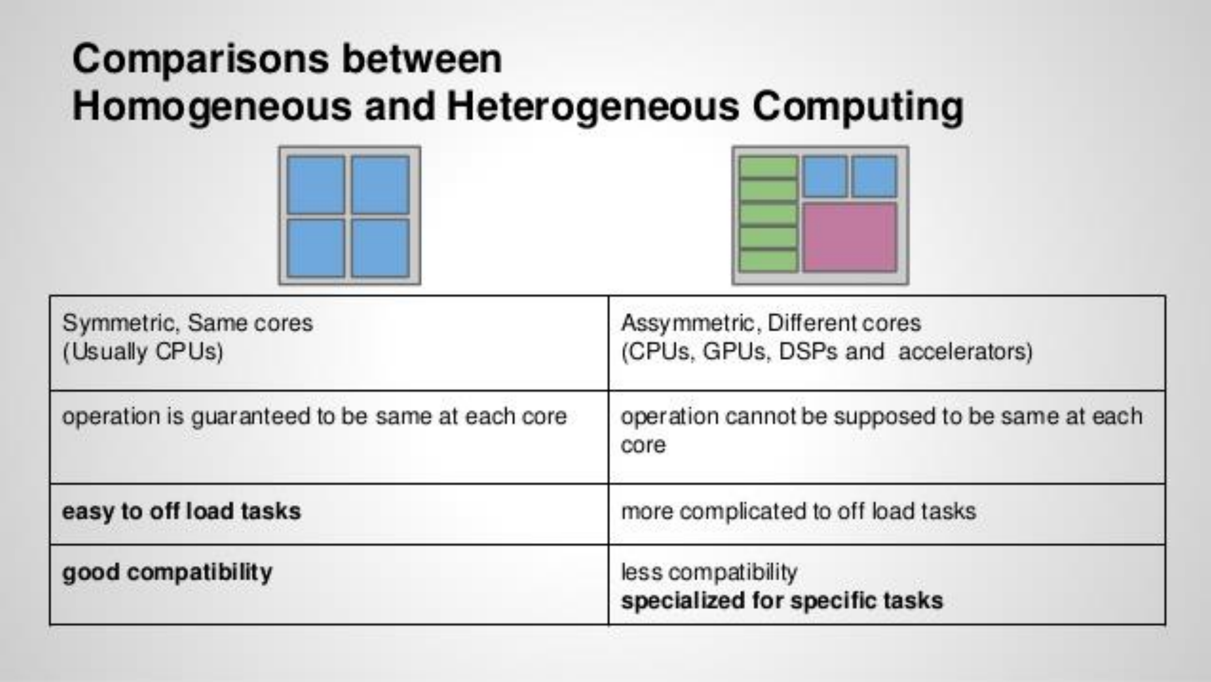
Single core Era **->** Multi-core Era **->** Heterogeneous Systems Era

- **Heterogeneous computing refers to systems that use more than one kind of processors or cores**
  - These systems gain performance or energy efficiency not just by adding the same type of processors, but by adding dissimilar (co)-processors, usually incorporating specialized processing capabilities to handle particular tasks
  - Systems with General Purpose Processors (GPPs), GPUs, DSPs, ASIPs etc.

- **Heterogeneous systems offer the opportunity to significantly increase system performance and reduce system power consumption**

Heterogeneous computing (2)

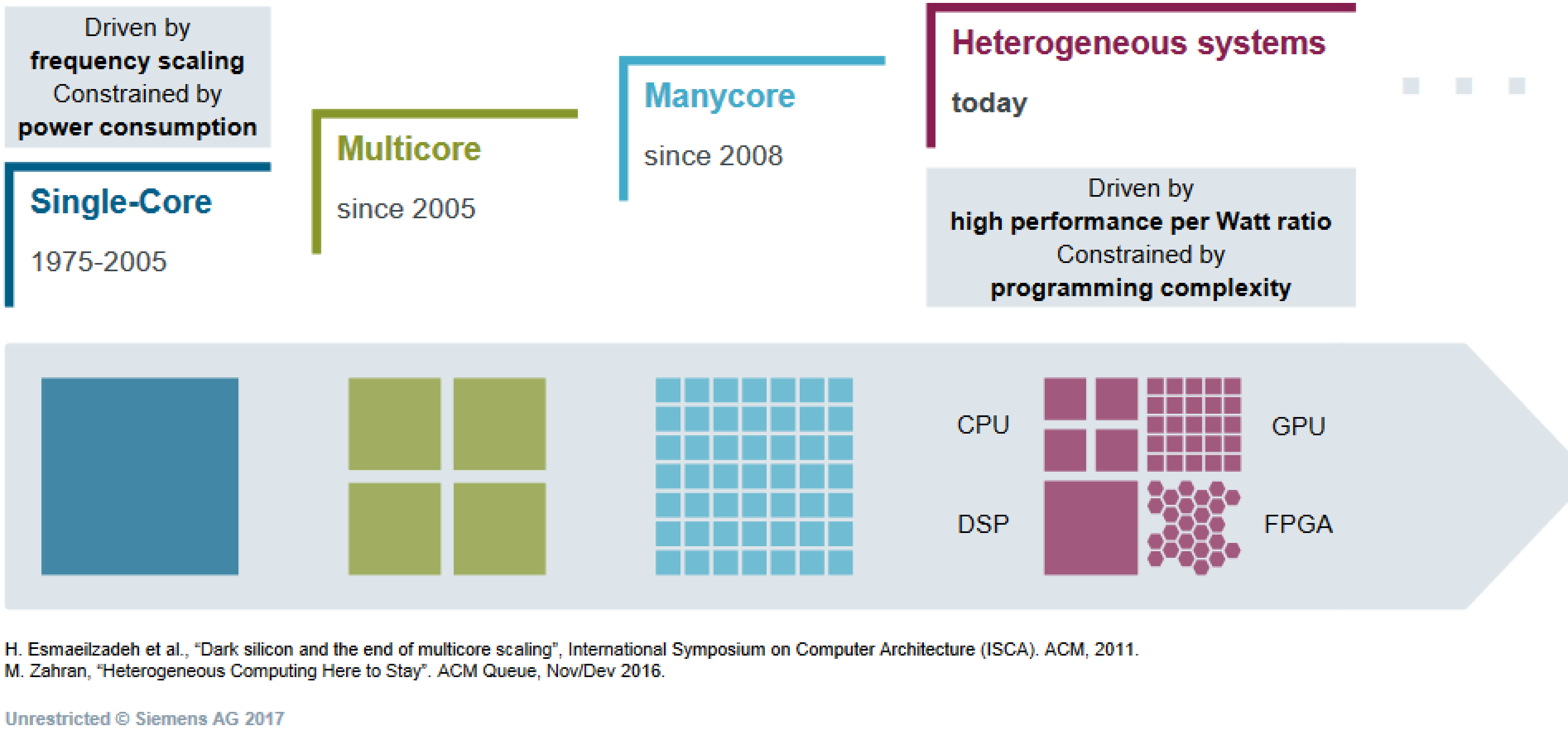
- Software issues:
  - Offloading
  - Programmability – think about CPU code (C code), GPU code (CUDA), FPGA code (VHDL)
  - Portability - What happens if your code runs on a machine with an FPGA instead of a GPU



Comparisons between Homogeneous and Heterogeneous Computing

Symmetric, Same cores (Usually CPUs)	Assymmetric, Different cores (CPUs, GPUs, DSPs and accelerators)
operation is guaranteed to be same at each core	operation cannot be supposed to be same at each core
easy to off load tasks	more complicated to off load tasks
good compatibility	less compatibility **specialized for specific tasks**

Hardware Trends
From single core processors to heterogeneous systems on a chip



Driven by
frequency scaling
Constrained by
power consumption

Single-Core

1975-2005

Multicore

since 2005

Manycore

since 2008

Heterogeneous systems

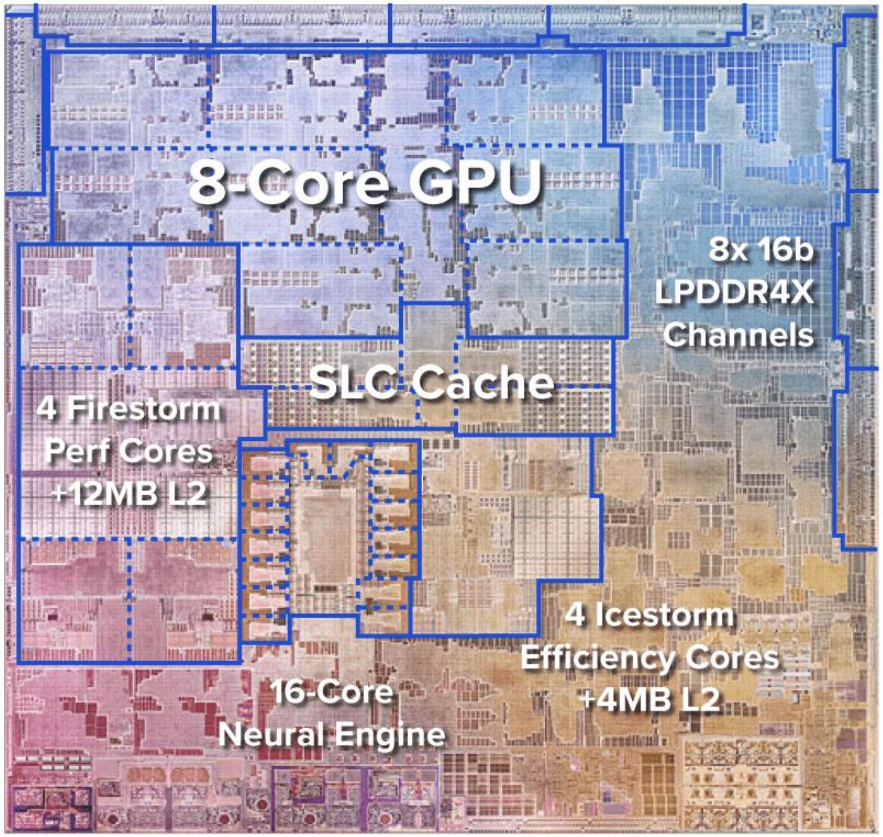
today

Driven by
high performance per Watt ratio
Constrained by
programming complexity

H. Esmaeilzadeh et al., "Dark silicon and the end of multicore scaling", International Symposium on Computer Architecture (ISCA). ACM, 2011.
M. Zahran, "Heterogeneous Computing Here to Stay". ACM Queue, Nov/Dev 2016.

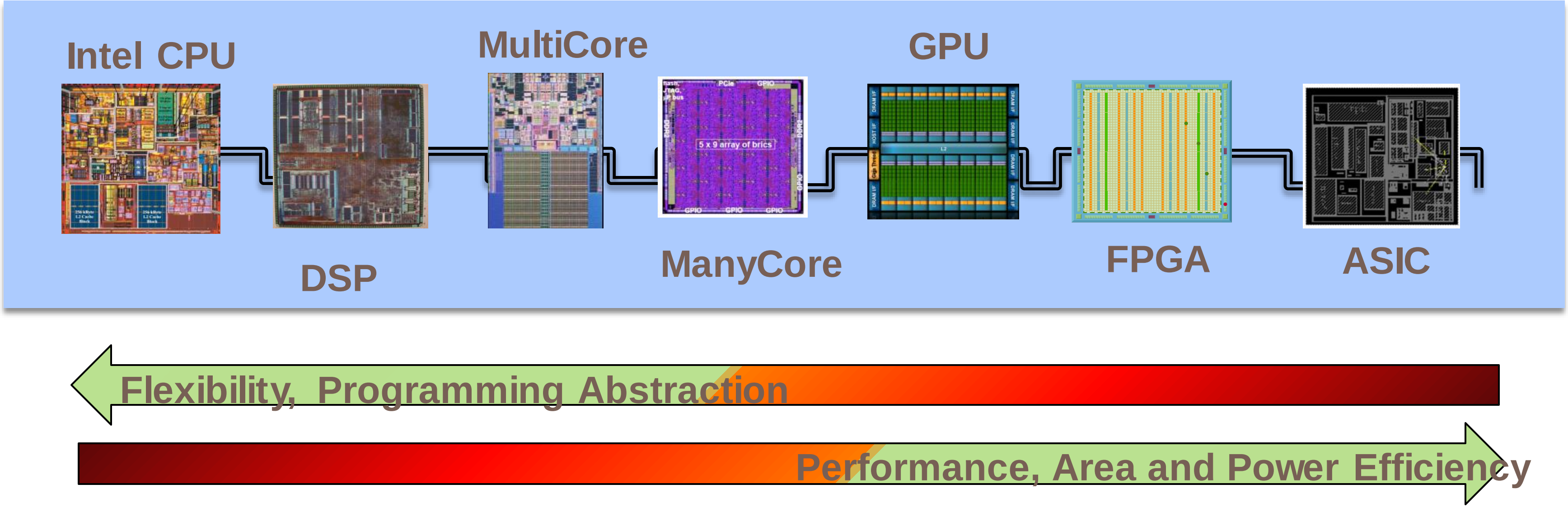
Taken from https://embb.io/downloads/MTAPI_EMBB.pdf

How modern processors look like?


8-Core GPU
8x 16b LPDDR4X Channels
SLC Cache
4 Firestorm Perf Cores +12MB L2
4 Icestorm Efficiency Cores +4MB L2
16-Core Neural Engine

Arm M1 processor.
Taken from https://www.toptal.com/apple/apple-m1-processor-compatibility-overview

Comparison of Hardware Architectures



Intel CPU

DSP

MultiCore

ManyCore

GPU

FPGA

ASIC

Flexibility, Programming Abstraction

Performance, Area and Power Efficiency

CPU:

- Market-agnostic
- Accessible to many programmers (Python, C++)
- Flexible, portable

FPGA:

- Somewhat Restricted Market
- Harder to Program (VHDL, Verilog)
- More efficient than SW
- More expensive than ASIC

ASIC

- Market-specific
- Fewer programmers
- Rigid, less programmable
- Hard to build (physical)

What language do you think is most used in High Performance Computing

- Chose the right answer
  1. C#
  2. C/C++
  3. Java
  4. Python

High Performance Computing (HPC) Programming Languages

- HPC is all about performance
- The most used languages in HPC are
  - C/C++
  - Fortran – there are many old massive applications which are still running, e.g., weather forecast (MetOffice)

Parallel Programming Models/Frameworks/Libraries

- There are too many parallel programming models to write parallel applications
  - Which one to use?
    - Ease of use
    - Performance
    - Portability
  - Parallel programming libraries/frameworks include OpenMP, MPI, OpenCL, OpenACC, CUDA, …

Design Patterns for Parallel Programming

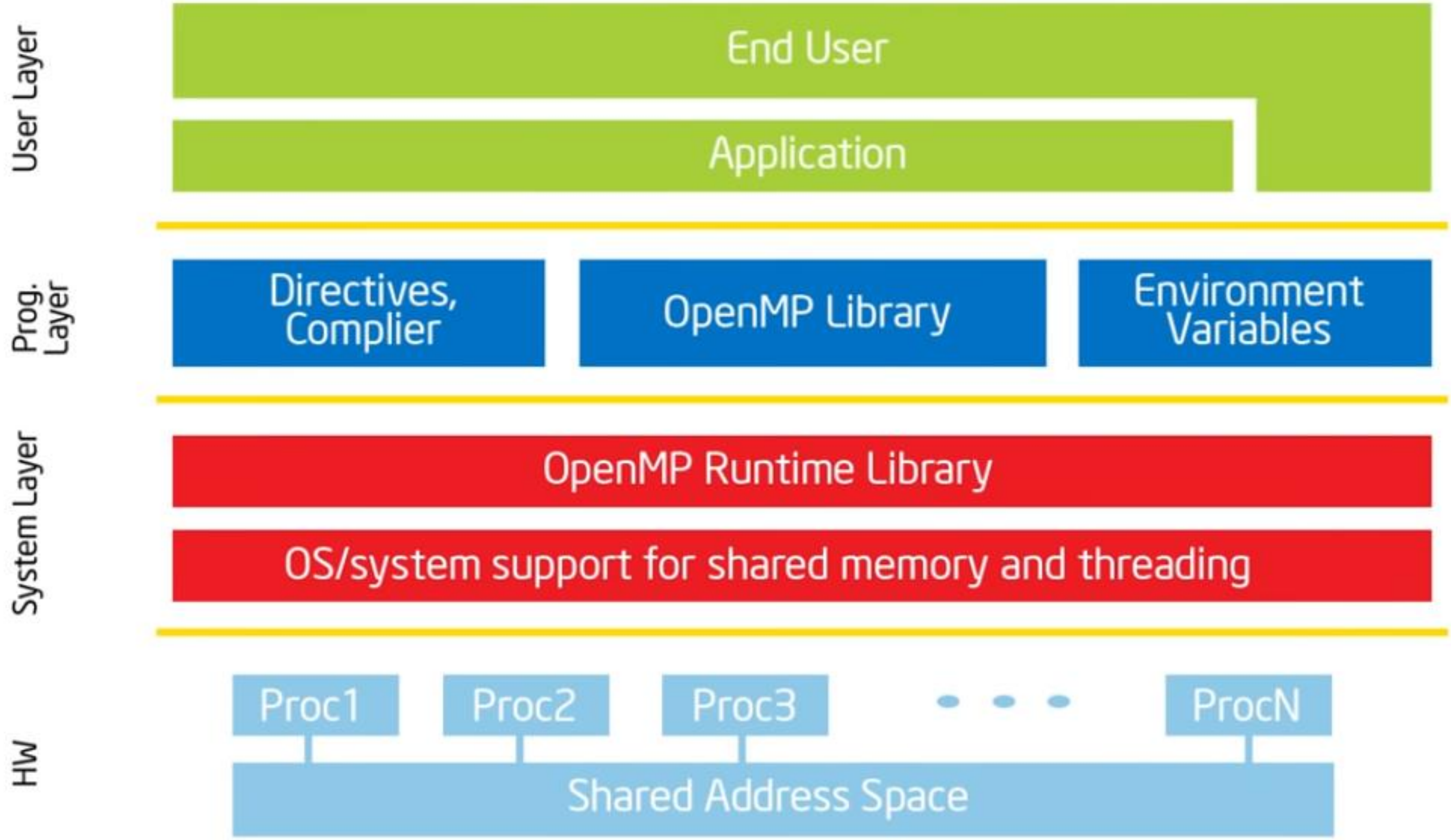
- No matter which programming language you use, there are specific algorithmic concepts that are universal
- These are the ***design patterns for parallel programming***

- **Examples of design patterns include:**
  - **Single Program Multiple Data (SPMD) pattern**
    - A single program runs on many processing elements
  - **Loop parallelism pattern**
    - Most used in OpenMP – we have seen many examples
  - **Task Parallelism**
  - **Divide and Conquer pattern**

What is OpenMP? (1)

- ***OpenMP (open multi-processing)*** : An API for writing multithreaded apps
- A set of compiler directives and library routines for parallel application programmers
- Greatly simplifies writing multithreaded programs in Fortran, C/C++

What is OpenMP? (2)



User Layer

End User

Application

Prog. Layer

Directives, Complier

OpenMP Library

Environment Variables

System Layer

OpenMP Runtime Library

OS/system support for shared memory and threading

HW

Proc1

Proc2

Proc3

• • •

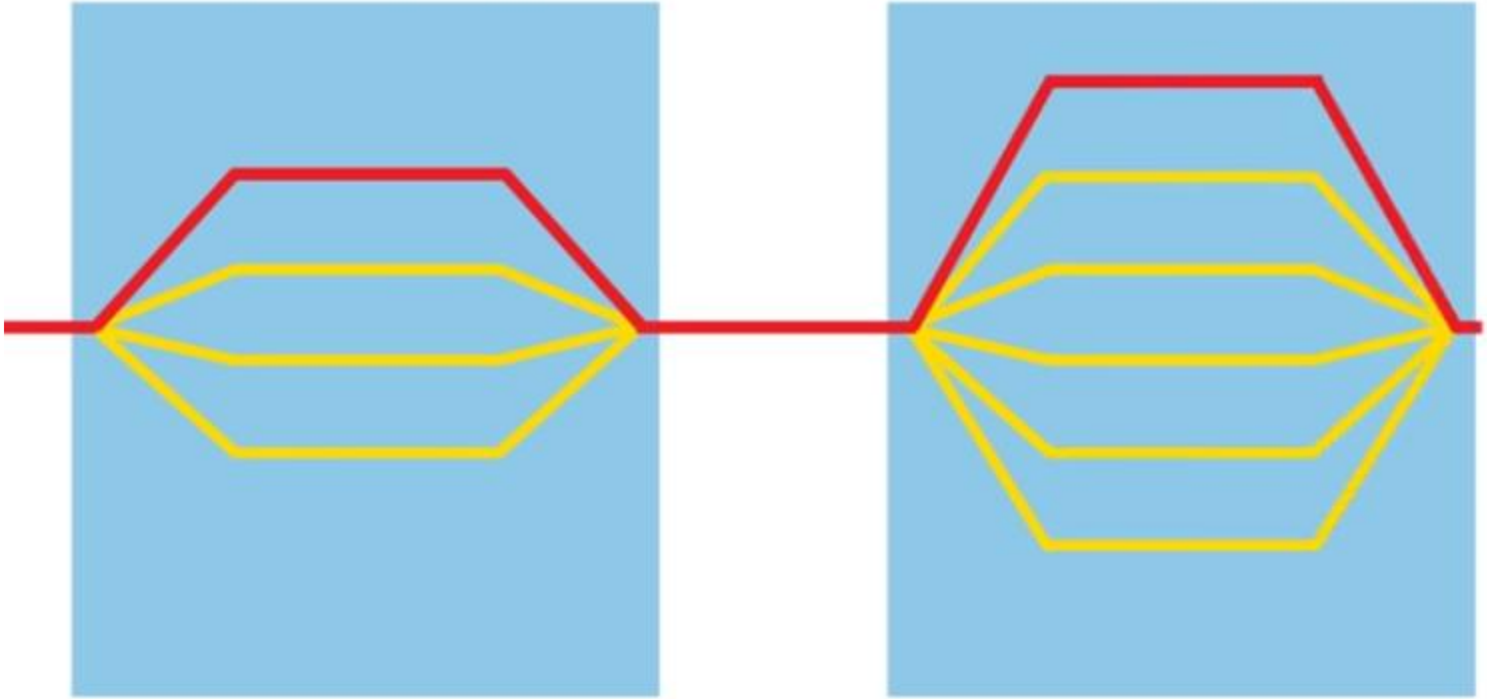
ProcN

Shared Address Space

Taken from https://www.youtube.com/watch?v=6jFkNjhJ-Z4&list=PLLX-Q6B8xqZ8n8bwjGdzBJ25X2utwnoEG&index=3

What is OpenMP? (3)

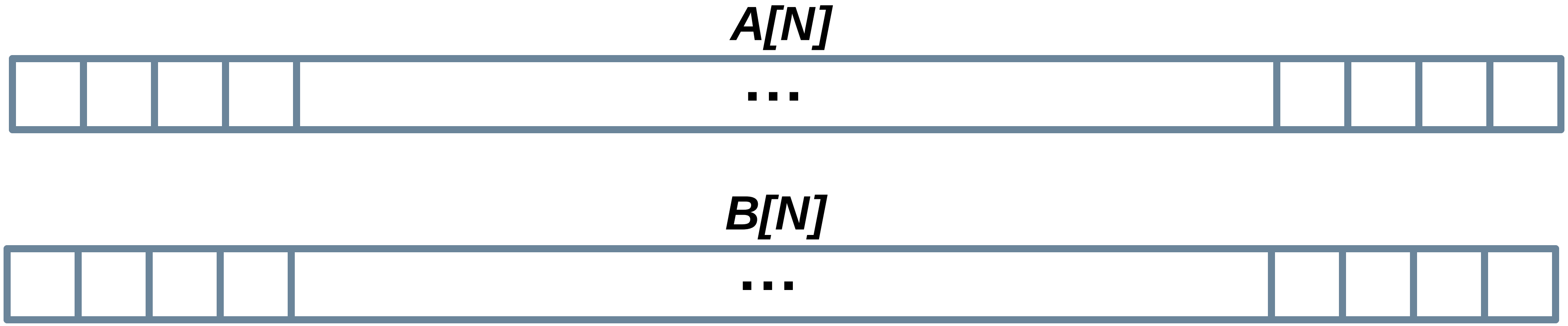
- Fork-Join Parallelism



Array Addition Example (1)

- How hard it is to parallelize this program on our PCs?

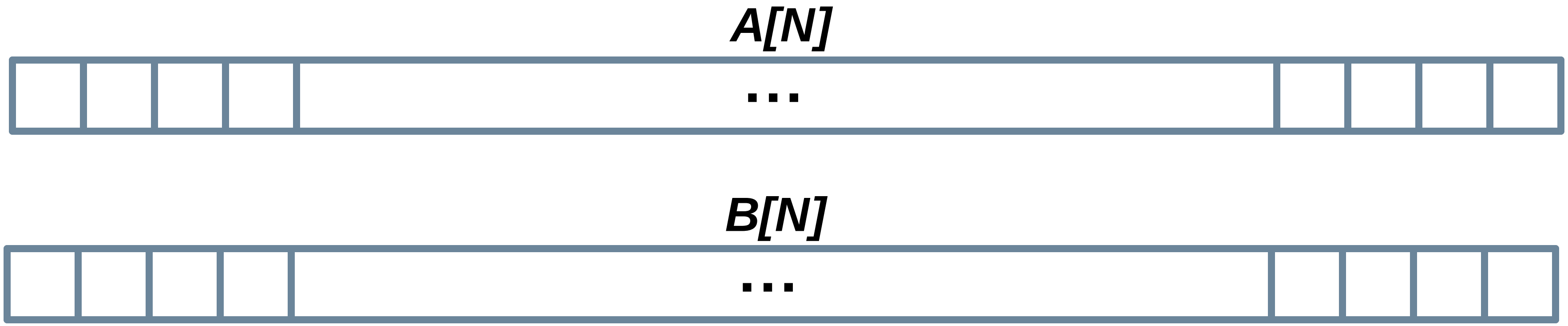
```
for (i=0; i<N; i++)
 A[i]=A[i] + B[i];
```



Array Addition Example (2)

```
for (i=0; i<N; i++)
 A[i]=A[i] + B[i];
```

```
#pragma omp parallel for
for (i=0; i<N; i++)
 A[i]=A[i] + B[i];
```



Array Addition Example (3)

- **But what OpenMP actually does?**
  - Considering there are four threads, the i loop will be split into four parts and each core will execute its part

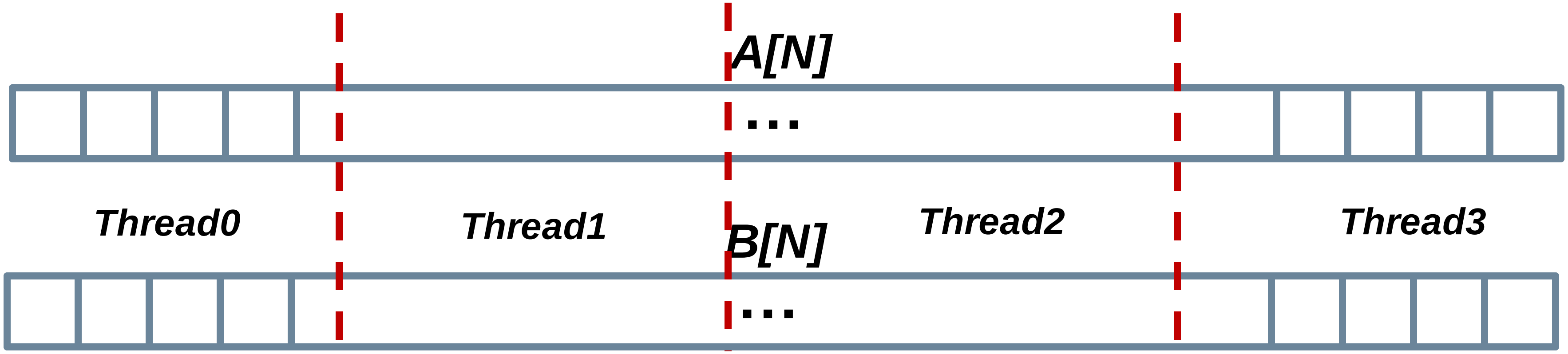
```
#pragma omp parallel for
for (i=0; i<N; i++)
 A[i]=A[i] + B[i];
```

```
for (i=0; i<250; i++)
 A[i]=A[i] + B[i];
```

```
for (i=500; i<750; i++)
 A[i]=A[i] + B[i];
```

```
for (i=750; i<1000; i++)
 A[i]=A[i] + B[i];
```

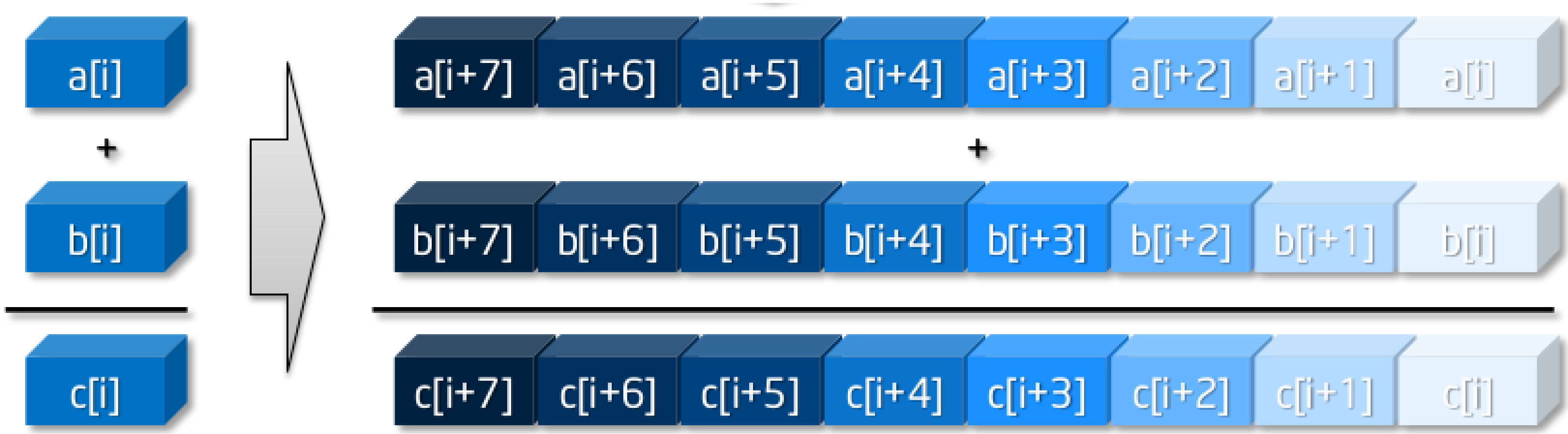
```
for (i=250; i<500; i++)
 A[i]=A[i] + B[i];
```



Array Addition Example (4)
Parallelization + Vectorization

```
for (i=0; i<N; i++)
 A[i]=A[i] + B[i];
```

```
#pragma omp parallel for simd
for (i=0; i<N; i++)
 A[i]=A[i] + B[i];
```



Serial VS Parallel version
See how elegant OpenMP is

```
double un_opt(){
 int i;
 double x, pi, sum=0.0;
 double step;

  step=1.0/(double) num_steps;

 for (i=0; i<num_steps; i++){
  x=(i+0.5)*step;
  sum = sum + 4.0 / (1.0 + x*x);
}
pi = step * sum;

return pi;
}
```

```
double version6(){
int i;
double x, pi, sum=0.0;
double step;

  step=1.0/(double) num_steps;

#pragma omp parallel for private(x) reduction(+:sum)
for (i=0; i<num_steps; i++){
  x=(i+0.5)*step;
  sum = sum + 4.0 / (1.0 + x*x);
}
pi = step * sum;

return pi;

}
```

Why GPUs?

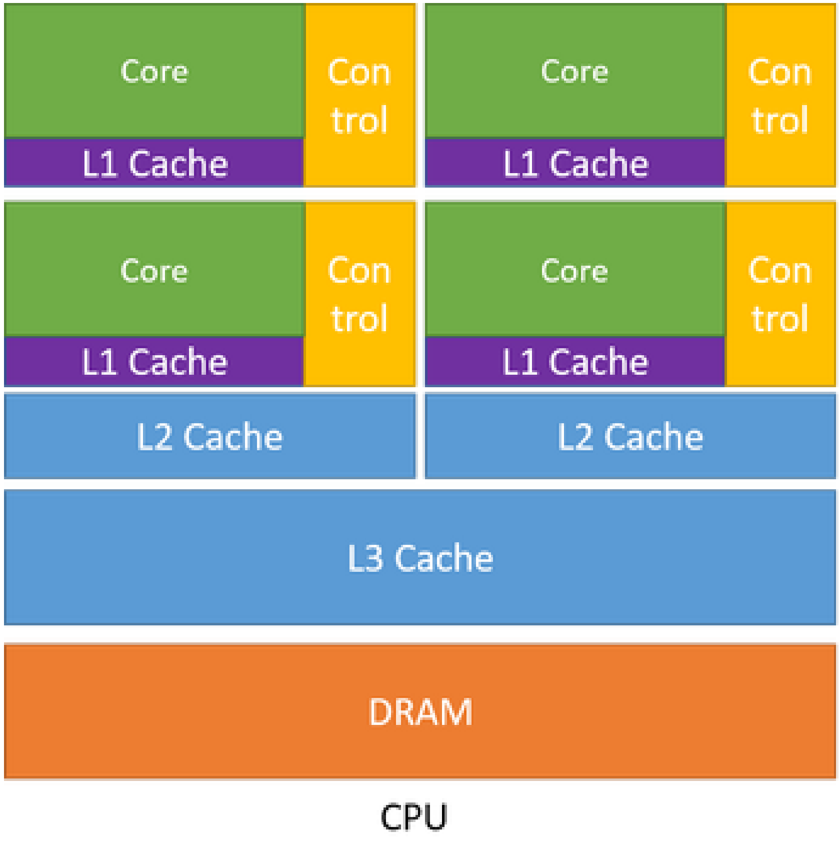
- *Graphics Processing Units (GPU)* were originally designed to accelerate the large number of multiply and add computations performed in graphics rendering
- The simulation of engineering and scientific problems is very closely related to the type of computation performed for graphic rendering.
  - Both perform a large number of floating point multiply-add computations.
  - GPUs are now designed specifically for the engineering and scientific marketplace
  - New GPU cards supporting up to 8 GPUs
- **GPUs are used to speedup highly-parallel computing tasks**
  - Machine learning, Image/Video Processing
- *Using GPUs for computing general purpose tasks is also known as GPGPU* (General Purpose GPU)

What's the Difference Between a CPU and a GPU?

- GPUs are best suited for repetitive and highly-parallel computing tasks.
  - Machine learning, Image/Video Processing
- You can find an interesting article in
- https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/

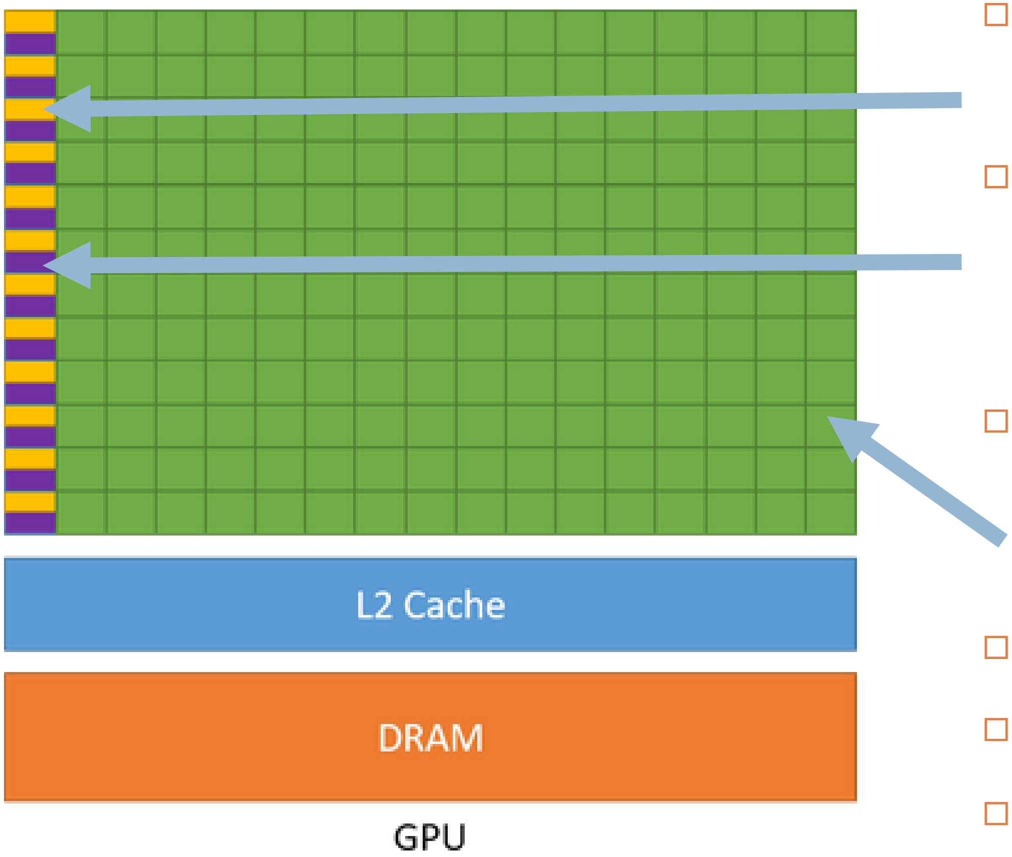
CPU	***GPU***
Several cores	*Many cores*
Low latency	*High throughput*
Good for serial processing	*Good for parallel processing*
Can do a handful of operations at once	*Can do thousands of operations at once*

CPU Design



- Powerful Cores
  - Reduced operation latency
- Large and slower caches
- Sophisticated control
  - Branch prediction
  - Data forwarding
- High CPU frequencies
- Each thread executes many-many instructions, a few threads
- The thread creation and thread switch overhead is high
- Threads can execute different code blocks

GPU Design



- Small caches
  - To boost memory throughput
- Simple control
  - No branch prediction
  - No data forwarding
- Many simple cores
  - Many, long latency but heavily pipelined for high throughput
- Massive number of threads
- Lower operational frequencies
- Lightweight threads, execute just a few instructions
- Threads execute the same code block

GPU Parallel Programming Frameworks

- The main GPU parallel programming frameworks are:
  - **CUDA (Compute Unified Device Architecture)**
    - Only for Nvidia GPUs - Nvidia Corporation proprietary
    - By far Best performance for Nvidia GPUs
  - **OpenCL (Open Computing Language)**
    - Open, maintaned by the Khronos Group
    - Programming is not than different from CUDA
    - Portable - CPUs, GPUs, other coprocessors
  - **OpenMP (Open Multi-Processing) – code annotation**
    - Very easy to use
    - Portable - CPUs, GPUs, other coprocessors
  - **OpenACC (open accelerators) – code annotation**
    - Very easy to use
    - Portable – CPUs, GPUs, other coprocessors

Vector Addition Example using OpenMP and OpenACC

- ***See how easy it is to write GPU code using OpenMP or OpenACC …***
  - **But not that fast as CUDA for Nvidia GPUs …**
  - Why?
    - CUDA is designed just for Nvidia GPUs, CUDA code is at a lower level, better control of the hardware resources, allows for code optimizations…

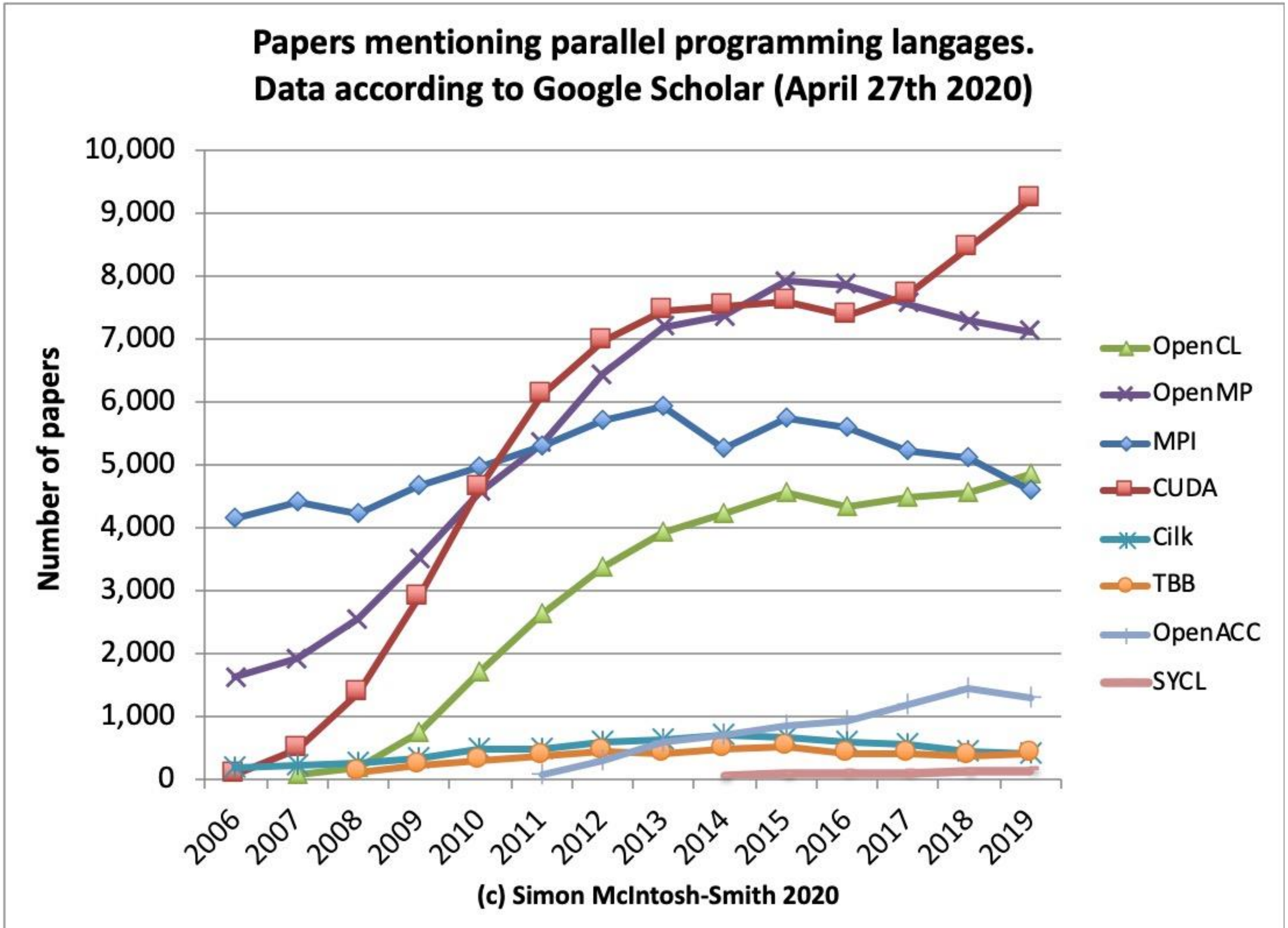
```
 // OpenACC code that runs on the GPU
#pragma acc kernels copyout(c[0:n]) copyin(a[0:n], b[0:n])
  for (i=0; i<n; i++) {
    c[i] = a[i] + b[i];
  }
```

```
// OpenMP code that run on the GPU
#pragma omp target map(to: a[0:N], b[:N]) map(from: c[0:N])
#pragma omp parallel for
for (int i = 0; i < N; i++) {
      c[i] = a[i] + b[i];
    }
```

How a CUDA program looks like?

```
void sin_serial(const float* in, float* out) {
int i;

for (i = 0; i < N; i++)
out[i] = sinf(in[i]);
}



__global__ void sin_parallel (const float* in, float* out) {

int g_id = threadIdx.x + blockIdx.x * blockDim.x;

if (g_id < N) {
out[g_id] = sinf(in[g_id]);
 }
}
```

Parallel Programming Languages Popularity in Research (not industry)



(c) Simon McIntosh-Smith 2020

New ExaScale hardware architectures have been announced

- ***Exascale computing*** is expected to revolutionize computational science and engineering by providing ***1000x the capabilities*** of currently available computing systems, while having a ***similar power footprin***t.
- The new exascale hardware architectures are heterogeneous
  - CPUs+GPUs (Aurora)
  - CPUs+FPGAs (Arm EPI)
- Although, the exascale supercomputers are currently being developed, only a few HPC applications are so far able to fully exploit the capabilities of the current **petascale** systems, mainly because of their limited **scalability.**
- Therefore, efforts for preparing HPC applications for Exascale are needed
  - People with such expertise get highly payed jobs

Questions?