HPC and hardware heterogeneity, how to navigate in this environment?

Prof. Dr. Alfredo Goldman

Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

September 2023

Agenda

About myself

The importance of Computer Science

Motivation for using HPC

Current state (yes, it is with heterogeneous hardware)

About be

Assistant professor at IME - USP 1993

MSc (1994) and Ph.D (1999) in Computer Science

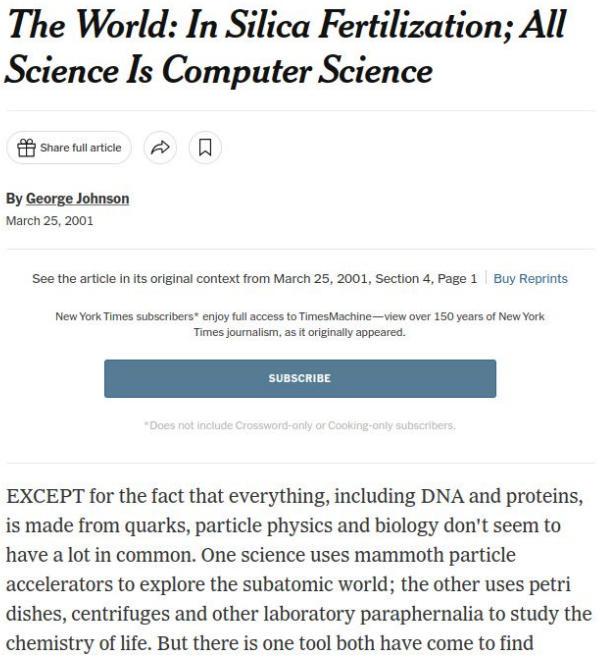
Board of governors in the Brazilian Computer Society

Organizer and co-PC chair of many tracks/conferences

co-Workshop chair of SC 23

The importance of Computer Science

Paper from NYT in 2001



The World: In Silica Fertilization; All Science Is Computer Science

Share full article

By George Johnson
March 25, 2001

EXCEPT for the fact that everything, including DNA and proteins, is made from quarks, particle physics and biology don't seem to have a lot in common. One science uses mammoth particle accelerators to explore the subatomic world; the other uses petri dishes, centrifuges and other laboratory paraphernalia to study the chemistry of life. But there is one tool both have come to find

Facts about the ever growing influence of CS

Changes in scale

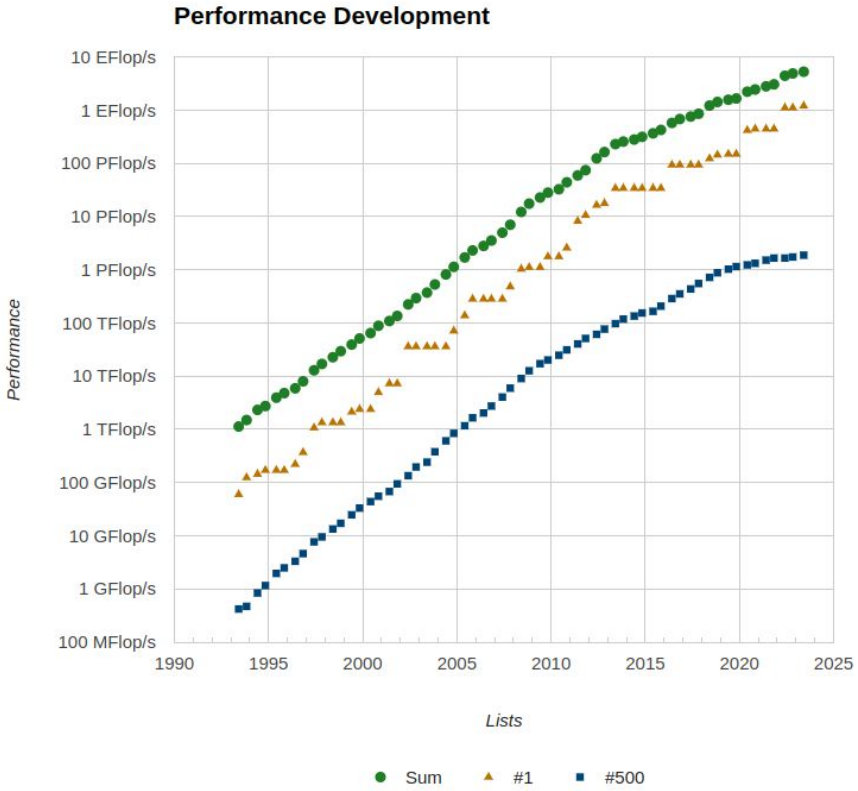
- Around 80's a computer for many people
- Today, many computers by person

Changes in computing power

- A smartwatch has more computing power than the Apollo 11 Computer
- But you should read “Your Smart Toaster Can’t Hold a Candle to the Apollo Computer”

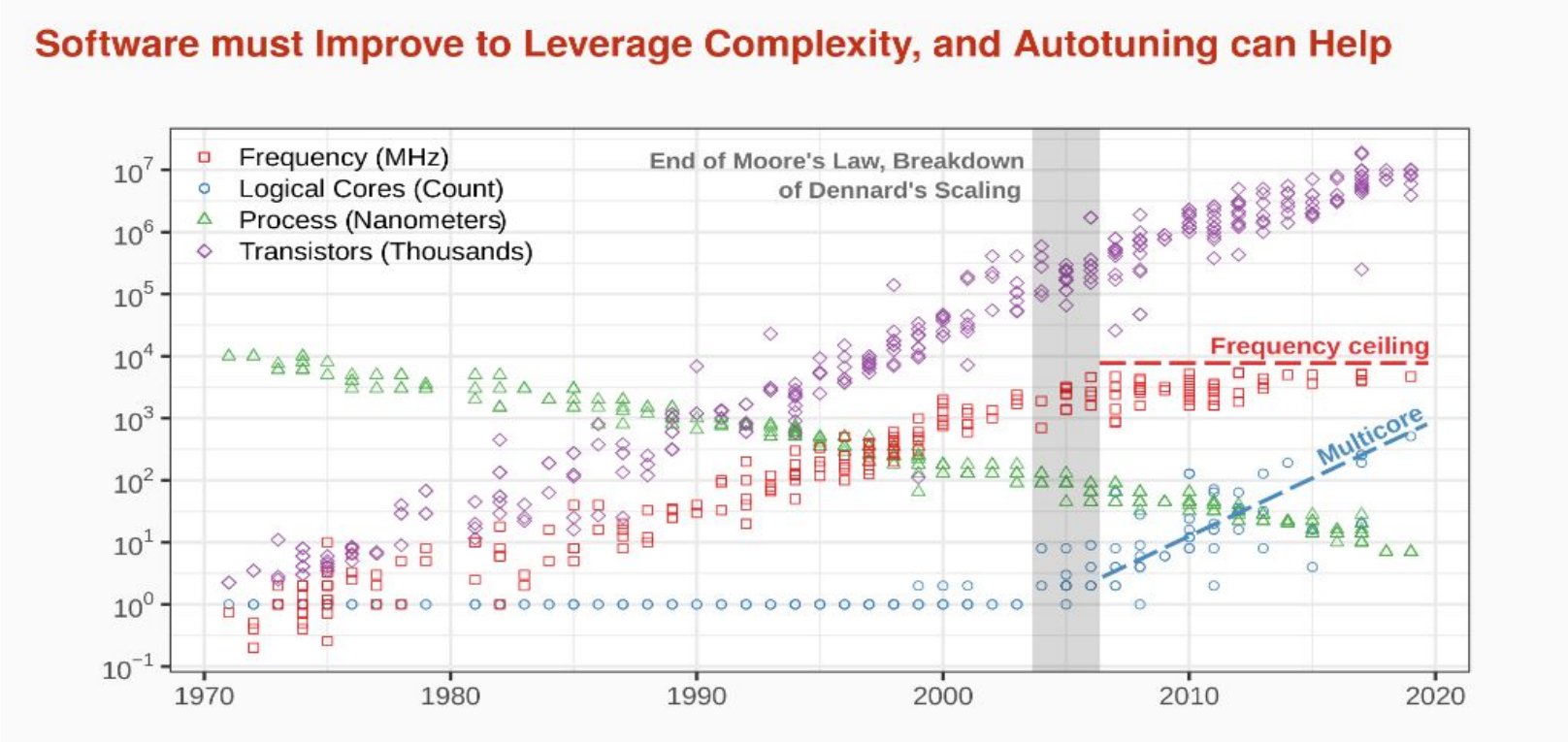
The virtual economy is a reality

- Marketplaces
- Online transactions
- Virtual currencies



Another interesting fact

Performance does not come for free as up to 2005



For some applications it worked fine

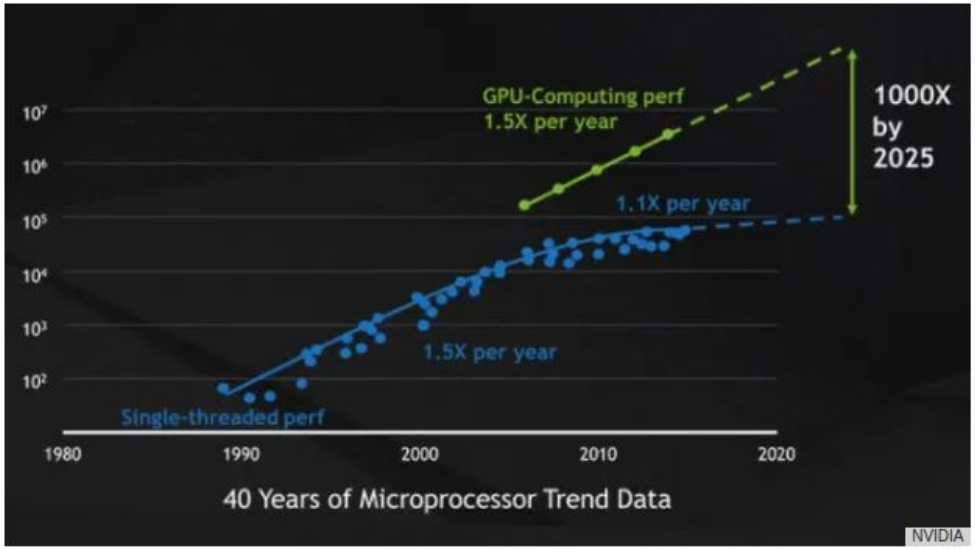
Deep Learning Example

GPUs performance

Tensor operations

Small theoretical advances

Great advances on NLP (chatGPT)



Not so good for some research areas

Paper of Nature

AI is changing the reality for some areas

Areas with the need for pattern recognition that have great amounts of data are killer applications!!!

A top-down view of the human nuclear pore complex, the largest molecular machine in human c

WHAT'S NEXT FOR THE AI PROTEIN-FOLDING REVOLUTION

AlphaFold, software that can predict the 3D shape of proteins, is already changing biology.

By Ewen Callaway

Start of the bad news

The Internet was supposed to be a free territory

About 10 years ago, the beginning of the bad news

- There was only one non commercial page among the top 10 on the web

Now, it is even worse, the regular user stays connected only to social networks

- Not transparent
- Addicting
- Have bad characteristics
  - Killer example: The Terraplanists

After the good news let's see the other side (½)

The advances of hardware does not imply in advances for software

- Large problems on many systems

- Bugs all over the world
  - No industry accept the software failure rates
  - This is know as software crisis (term coined in 1967!)
  - Many advances in software, but there is no “silver bullet”

- Python and JavaScript are the top programming languages

After the good news let's see the other side (2/2)

MPI is still the de facto standard for distributed programming

OpenMP appeared to ease the programing task

- But, it has still some difficulties
- “OpenMP is Not as Easy as It Appears” paper from 2016 :)

So, the machines evolved, but the way to use them not so much

Counterexamples:

- Virtualization -> Cloud
- Agile Methods

Before continuing

A question for the audience:

Who depends on software for the research?

How many years of experience of software do you have?

Bachelor's in Computer Science – Curriculum

- Duration: 4 years
- 23 compulsory courses
  - of these, 5 in mathematics and 1 in statistics
- 21 elective courses
  - of these, 1 for science and 1 for statistics
  - of these, 6 from any USP unit
- 240 hours of complementary academic activities (undergraduate teaching assistant, research projects, seminars, etc.)
- In terms of hours:
  - 2565 class hours; 720 work hours; 240 hours of complementary academic activities
- Total of 3525 hours of dedication

How to improve?

Large efforts is software engineering

- Automated tests are a most
- Modern languages to help the parallel developper
  - (Rust, GO, Julia, etc)
- Everything in the cloud
- Microservices
- HPC as a service

Large efforts to have strong curricula in CS courses

- Concepts are essential

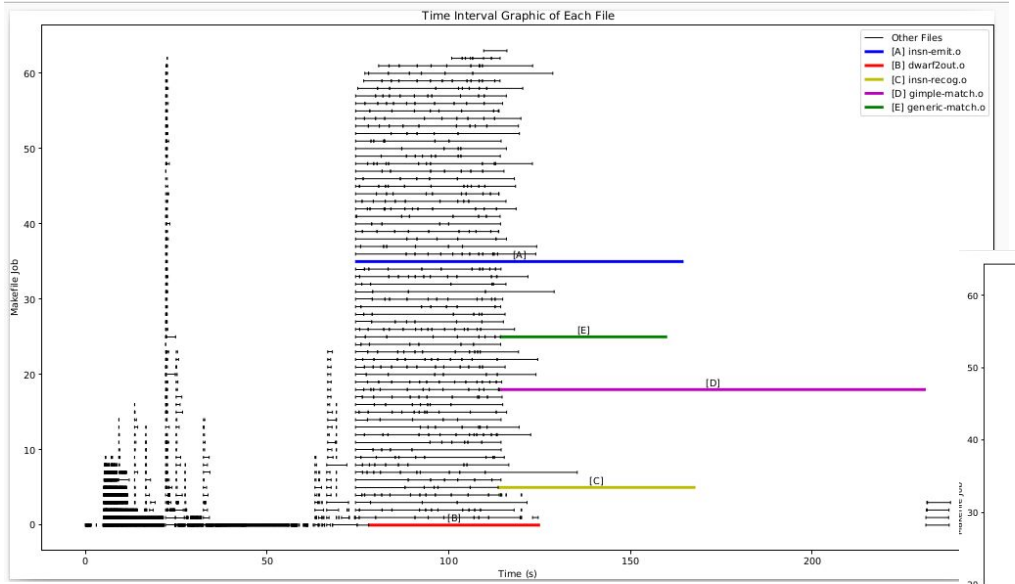
Efforts to show the importance of

Research Software Engineering

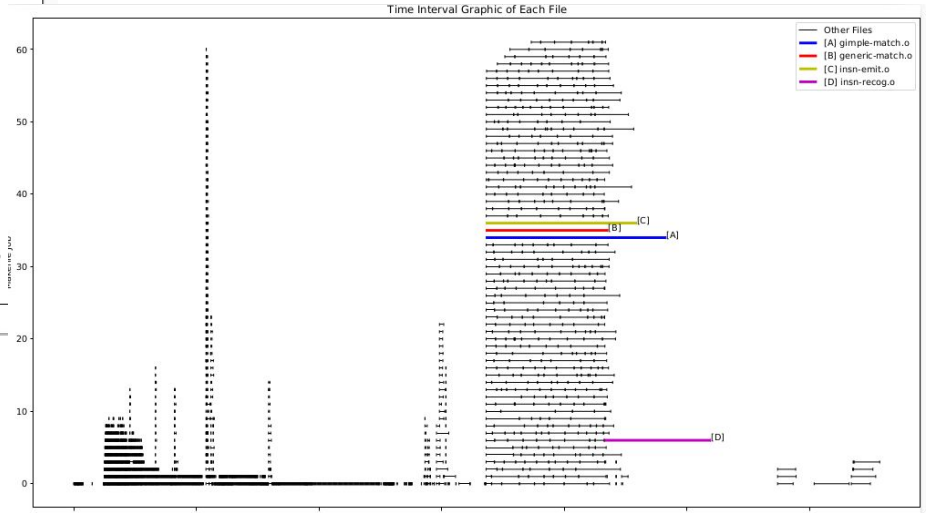
- See the talk from Daniel Katz at IME - USP

Examples of possible improvements - 1

Parallelization of GCC one of the best well known compilers!



Compiling Files in Parallel: A Study with GCC



Examples of possible improvements - 2

Parallelization of GIT, both grep and checkout

Supported by AWS

Very important for projects as Chromium

Parallelizing Git Checkout: a Case Study of I/O Parallelism
Running code on GIT :)

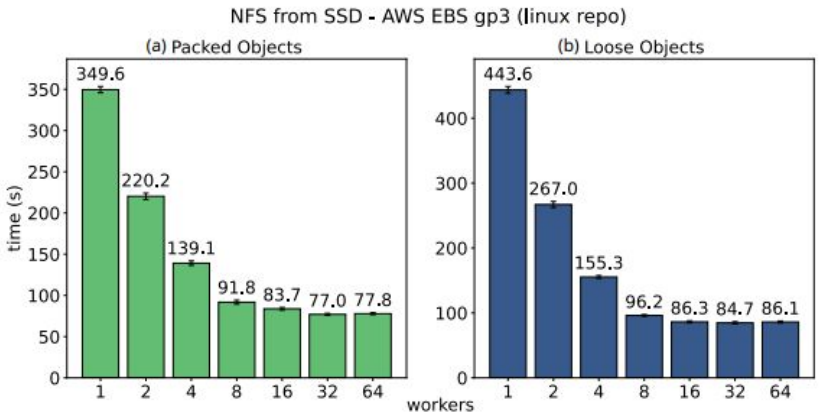


Figure 7.7: *Checkout benchmark on NFS - AWS EBS gp3 (SSD)*

Current state of heterogeneous HW

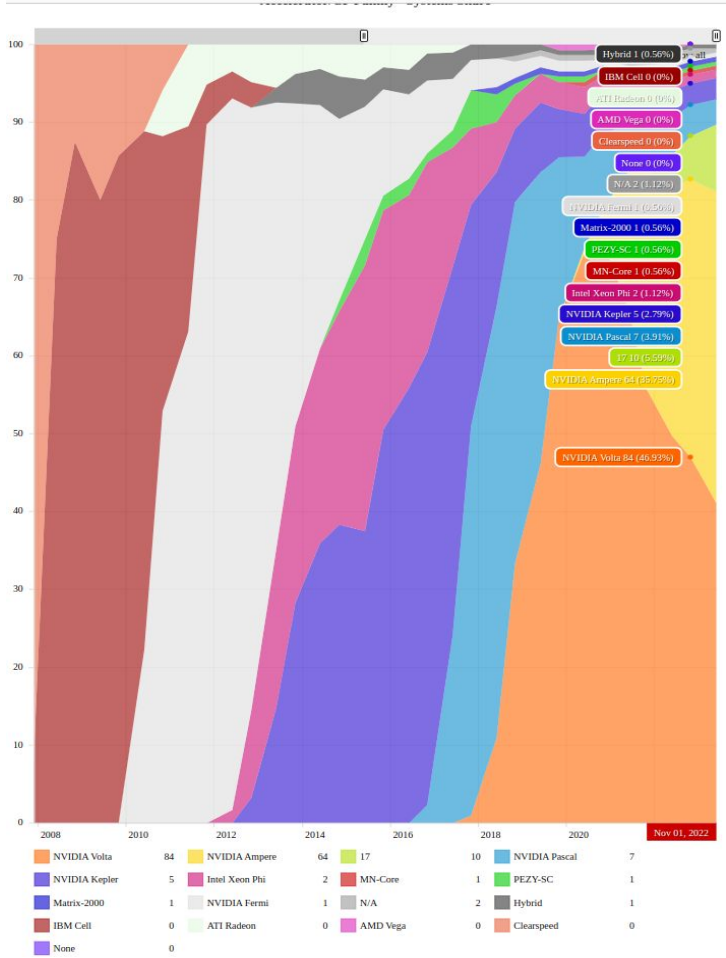
GPUs are already mainstream

GPUs are perfect for some applications

CUDA programming was a hype on the previous years

Now, there are many libraries with most common algorithms

There is a Lab on GPU programming running



There are other accelerators

TPU - Tensor Processing Units

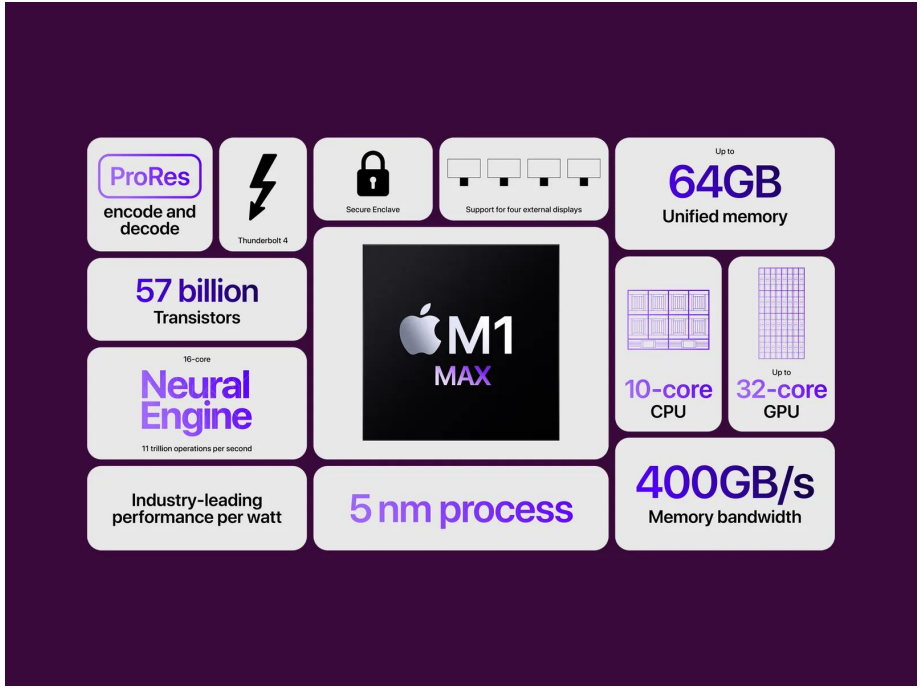
In memory processing

Persistent Memory

Other type of accelerators (FFT, ASICs)

An example of good use of accelerators

The Chip M1 from Apple



And now?

How to find a tradeoff between the hardware performance and the flexibility of software?

Creating new ways to do HW with SW :)

FPGAs!

What is an FPGA?

Basic Logic Circuits

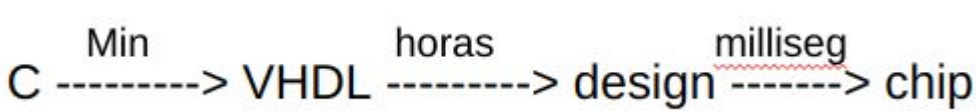
- **Ports**
- **Memory**
- **Auxiliary Processor**

This basic circuits can be connected to reproduce HW

Can be programmed using VHDL/Verilog

- **(HW description languages)**

Or using High Level Synthesis, from languages to circuits



Verilog

```
//NAND gate using data flow modeling
module nand_gate_d(a,b,y);
input a,b;
output y;

assign y = ~(a & b);

endmodule
```

The two sides

Advantages

- **Fast deploy**
  - **If the circuit is ready**
- **Big players (Xilinx & Altera)**
  - **(AMD & Intel)**
- **Available on the cloud**

Disadvantages

- **Slow Clock (hundreds of MHz)**
- **Learning Courbe**
  - **HW ou SW?**
- **Lack of standardization**
- **Lack of “circuits”**

Research opportunities

To implement algorithms of part of them in FPGAs

A CPU-FPGA heterogeneous approach for biological sequence comparison using high-level synthesis (Jorge et al.)

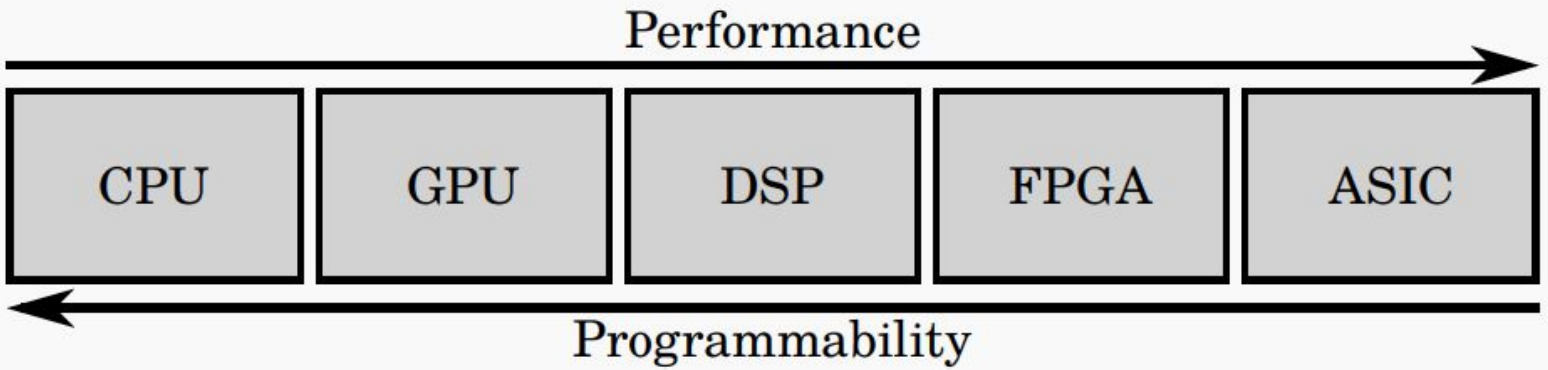
To change processors behavior

Enabling HW-based Task Scheduling in Large Multicore Architectures (Moras et al.)

Which circuits have to be on the FPGA?

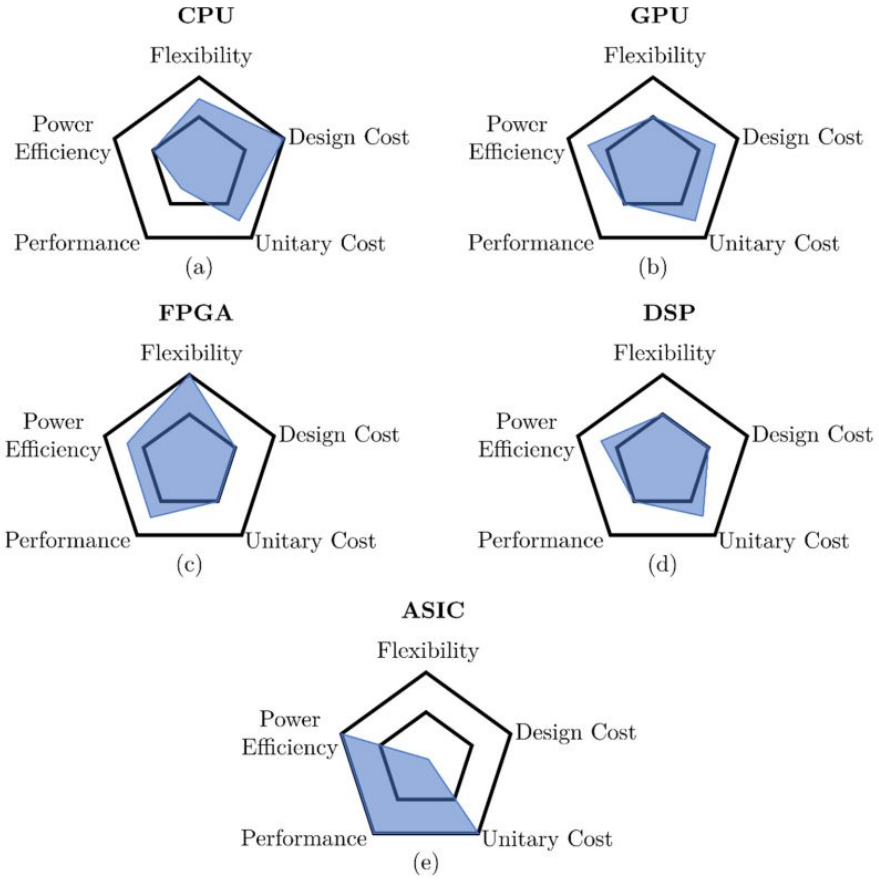
Are we going to have processors with integrated FPGAs?

Where is the place of FPGAs (1/2)



Accelerators and FPGAs (2/2)?

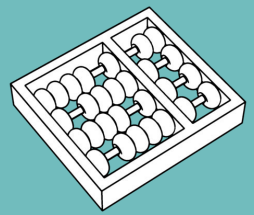
Taken from Hardware Architectures for Real-Time Medical Imaging, Alcaín et al. 2021



OpenMP Cluster (OMPC)

Cluster Programming with OpenMP only

University of Campinas (Unicamp)
Institute of Computing (IC)
Computing Systems Laboratory (LSC)



Overview

- Introduction
- Programming model
- Execution flow
- FPGA Integration

Introduction

The Project


UNICAMP


UFABC


FAPESP
FUNDAÇÃO DE AMPARO À PESQUISA
DO ESTADO DE SÃO PAULO


USP
UNIVERSIDADE
DE SÃO PAULO

OMPC: Cluster Programming Made Easy

- Cluster programming with OpenMP
  - Mixing classic and target OpenMP directives
- Extends the LLVM OpenMP runtime for distributed architectures
  - Automatic task mapping and scheduling
  - Uses MPI for interprocess communication
- Offers fault tolerance mechanisms

OMPC users can program MPI processes without writing any MPI code

Where can I find details?

The OpenMP Cluster Programming Model

Hervé Yviquel*
University of Campinas – UNICAMP
Brazil

Marcio Pereira
University of Campinas – UNICAMP
Brazil

Emílio Francesquini
Federal University of ABC – UFABC
Brazil

Guilherme Valarini
University of Campinas – UNICAMP
Brazil

Gustavo Leite
University of Campinas – UNICAMP
Brazil

Pedro Rosso
University of Campinas – UNICAMP
Brazil

Rodrigo Ceccato
University of Campinas – UNICAMP
Brazil

Carla Cusihualpa
University of Campinas – UNICAMP
Brazil

Vitoria Dias
University of Campinas – UNICAMP
Brazil

Sandro Rigo
University of Campinas – UNICAMP
Brazil

Alan Sousa
Petrobras

Guido Araujo†
University of Campinas – UNICAMP

Programming Model

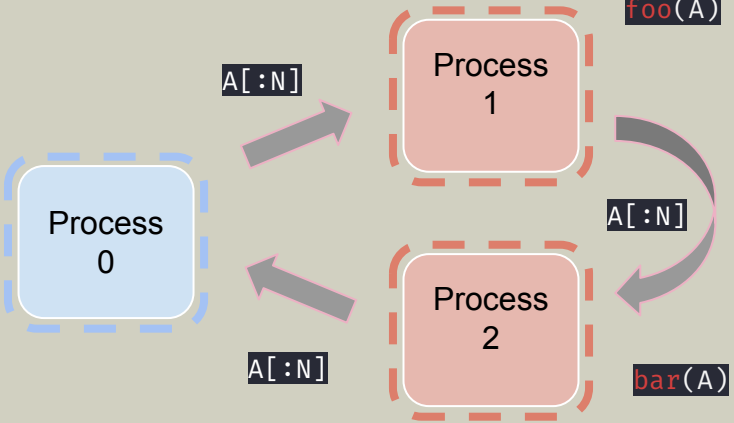
OMPC Task

```
#pragma omp target nowait
printf(“Hello World\n”)
```

- A target task is an independent piece of work to program accelerators
  - They are standard OpenMP directives
  - The concept was expanded by OMPC to program clusters

OMPC Data Management

```
#pragma omp target enter data map(to: A[:N]) nowait depend(out: *A)
#pragma omp target nowait depend(inout: *A)
foo(A)
#pragma omp target nowait depend(inout: *A)
bar(A)
#pragma omp target exit data map(from: A[:N]) nowait depend(inout: *A)
```



- Data mapping for multiple target tasks
- **Data dependencies** must be specified in the target tasks
- Allows **direct communication** between workers

Execution Flow

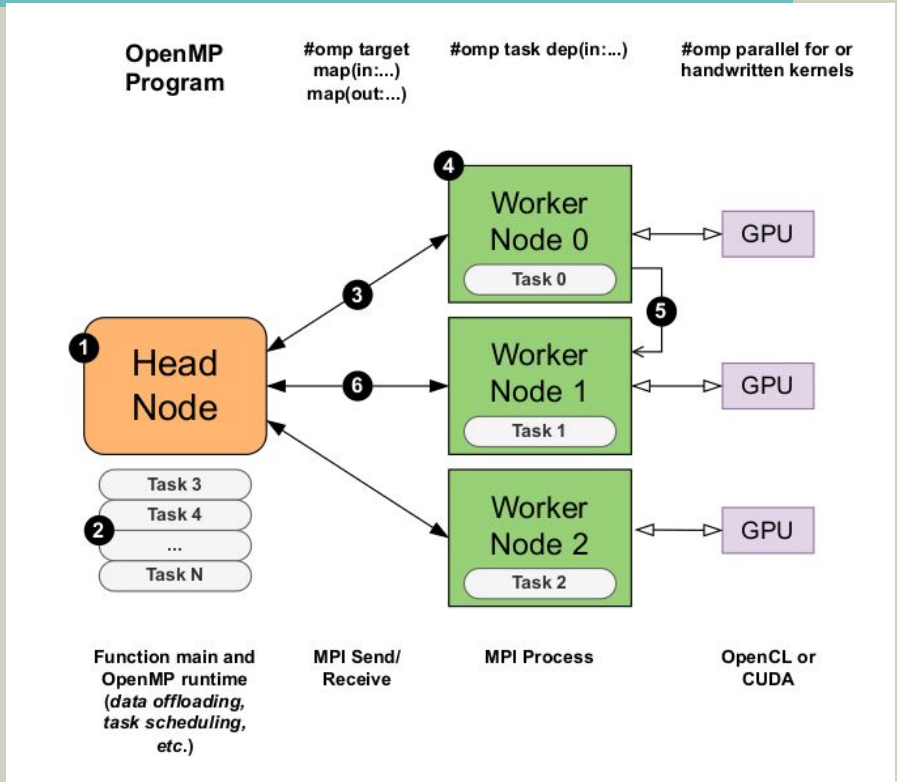
Programmer vs Runtime

- The **programmer** specifies through OpenMP directives
  - Target tasks with *target nowait* constructs
  - Data dependencies with the *depend* clause
  - Data mappings with *map* clauses

- The **OMPC runtime system** takes care of
  - Mapping tasks and data across the cluster to minimize communication
  - Maintaining data coherence between tasks and nodes
  - Providing fault tolerance mechanisms

OMPC Runtime Workflow

- Distributed architecture
  - Head node manages the tasks (load-balancing, etc)
  - Worker nodes execute the tasks
- 6-step execution
  1. Program execution
  2. Task generation
  3. Task distribution
  4. Task execution
  5. Inter-nodes communication
  6. Retrieve result

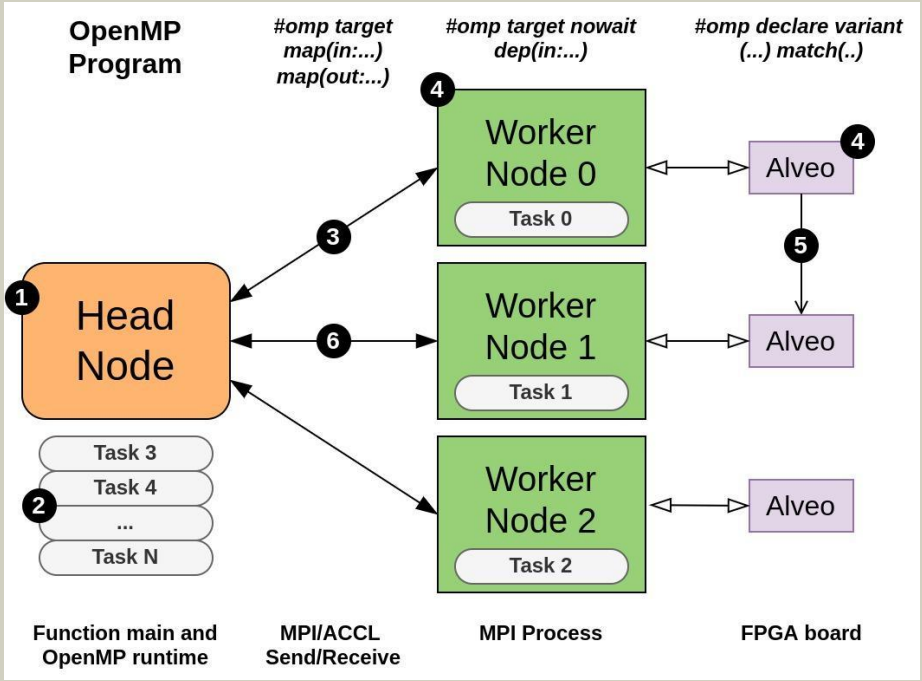


FPGA Integration

Execution Overview

Runtime Workflow with FPGAs

- 6-step execution
  1. Program execution
  2. Task generation
  3. Task distribution
  4. Task execution (CPU/FPGA)
  5. Inter-FPGA communication
  6. Retrieve result
- Communication
- Execution



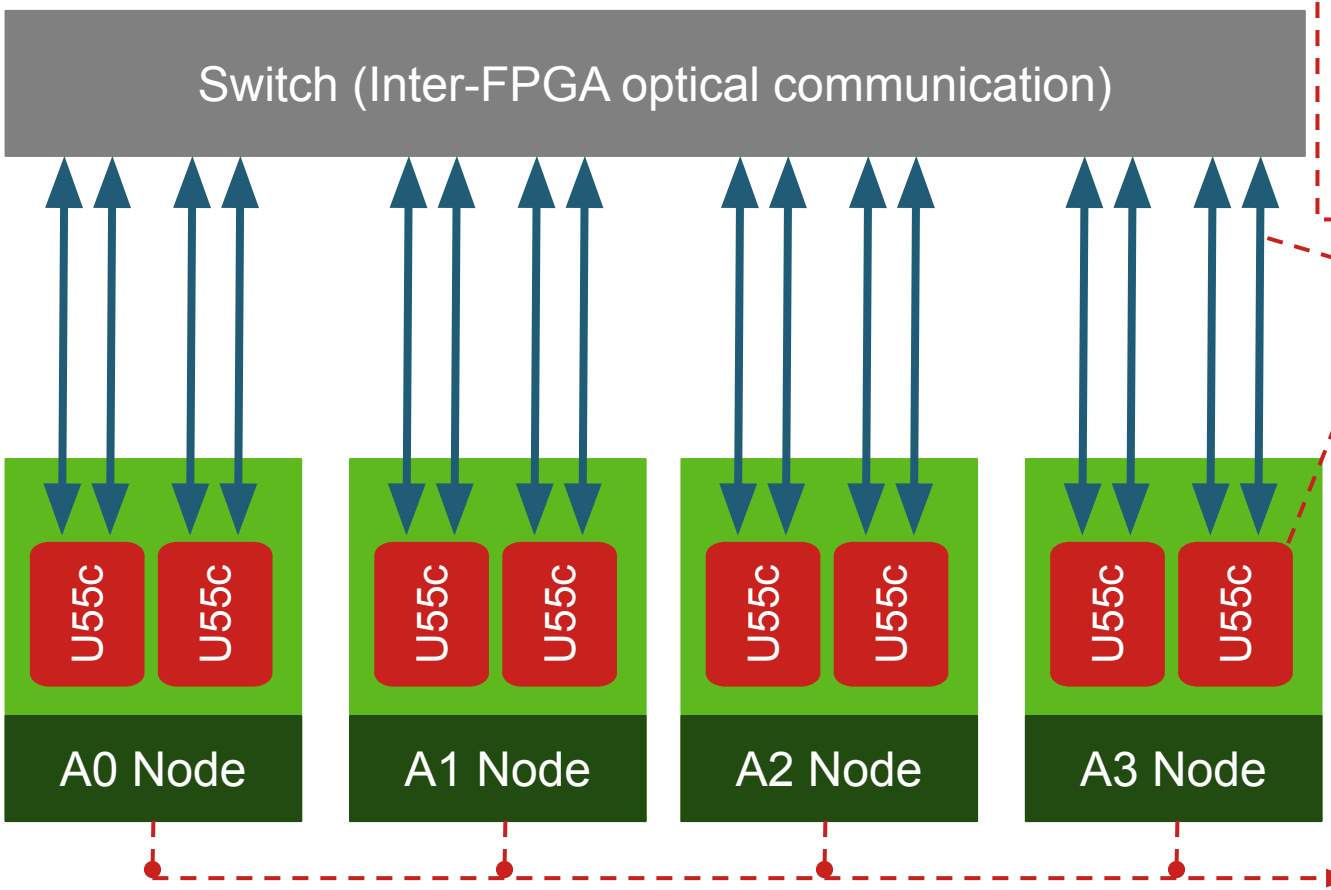
OMPC FPGA Code

```
void vadd_hw(int *in1, int *in2, int *out, unsigned int num); // FPGA implementation prototype
#pragma omp declare variant( vadd_hw ) match( device={arch(alveo)} )
void vadd_sfw(int *in1, int *in2, int *out, unsigned int num); // CPU impl. prototype

...

int OpenMPVadd(int *in1, int *in2, int *out, int BS, int NB) {
  for(int i = 0; i < NB; ++i) {
    int *A  = &in1[BS * i], *B  = &in2[BS * i], *C  = &out[BS * i];
    #pragma omp target depend( in: A[0], B[0] ) depend ( out: C[0])              \
                      map(tofrom: A[:BS], B[:BS], C[:BS] ) nowait
    vadd_sfw(A, B, C, block_size);
  }
}
```

The A-Machine

- A partnership USP, UNICAMP and UFABC sponsored by FAPESP
- 4 nodes having 2 state-of-the-art Alveo U55C per node (8 U55C total) interconnected using fiber optic cables
- One of the few in the World!

The A-Machine



Current Switch model: DELL - S4128F-ON (It has only 2 QSFP28 interfaces, but we need 16). Thus, we ordered a new switch having more QSFP28 (>16) interfaces

QSFP28 Cables

Alveo U55c

A0, A1, A2, A3: 2x Intel Xeon Silver 4210R 10Cores/20Threads, Base Frequency: 2.4 GHz, Max Frequency: 3.2GHz, Cache: 13.75MB.
A0, A1: 192GB DDR4-2666 RDIMM ECC RAM, 1x 2TB 7.2K RPM SATA HDD.
A2, A3: 128GB DDR4-2666 RDIMM ECC RAM, 1x 1TB, ST1000DM010-2EP102 (CC46), 1x 1TB Samsung SSD 970 EVO Plus

Final Message

The Hardware is available and it is heterogeneous

It depends on us to use the hardware in a very effective way

Alfredo Goldman (gold@ime.usp.br)