High Performance Co$^{mputing}_{ding}$ (HPC)

Alberto Dassatti - 2024

Who

- Professor: Alberto Dassatti
- Office: A11 (Cheseaux)
- E-mail: alberto.dassatti@heig-vd.ch
- Telephone: +41 24 557 61 60

2002: Master in Electronics Engineering, Politecnico di Torino

2003: Interim at Advanced System Technologies STMicroelectronics, Geneva

2004-2008 : Ph.D. in Telecomunications and Electronics (reconfigurable systems), Politecnico di Torino,

2008 : Visiting Ph.D. student UNSW Sydney, Australia

2008-2009: Post-Doc Researcher at VLSI Lab, Politecnico di Torino

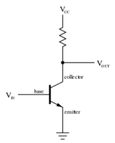
2003-2010: Entrepreneur fondateur of MicroC s.n.c., a Consultancy firm

2010-2012: ingénieur at NATO Undersea Research Centre

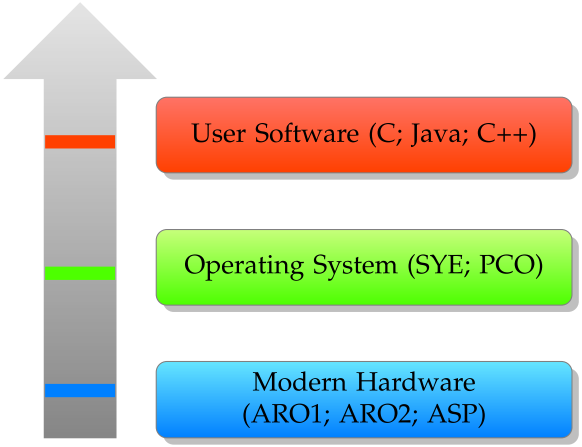
2012-2013: ingénieur au REDS

2013-: Professeur HES HEIG-VD

2018-: REDS' Director



What you know


User Software (C; Java; C++)
Operating System (SYE; PCO)
Modern Hardware (ARO1; ARO2; ASP)

What is HPC goal?

Make programs run faster!

What is HPC about

1. Defining *performances*

What is HPC about

1. Defining *performances*

2. Measure *performances*

What is HPC about

1. Defining *performances*
2. Measure *performances*
3. Some modern Hardware (Super Scalar CPU, Multi Cores, Caches, SIMD, GPU)

What is HPC about

1. Defining *performances*
2. Measure *performances*
3. Some modern Hardware (Super Scalar CPU, Multi Cores, Caches, SIMD, GPU)
4. Something about Compilers (LLVM, gcc)

What is HPC about

1. Defining *performances*
2. Measure *performances*
3. Some modern Hardware (Super Scalar CPU, Multi Cores, Caches, SIMD, GPU)
4. Something about Compilers (LLVM, gcc)
5. Something about libraries and languages (CUDA, OpenCL, BLAS)

What is HPC about

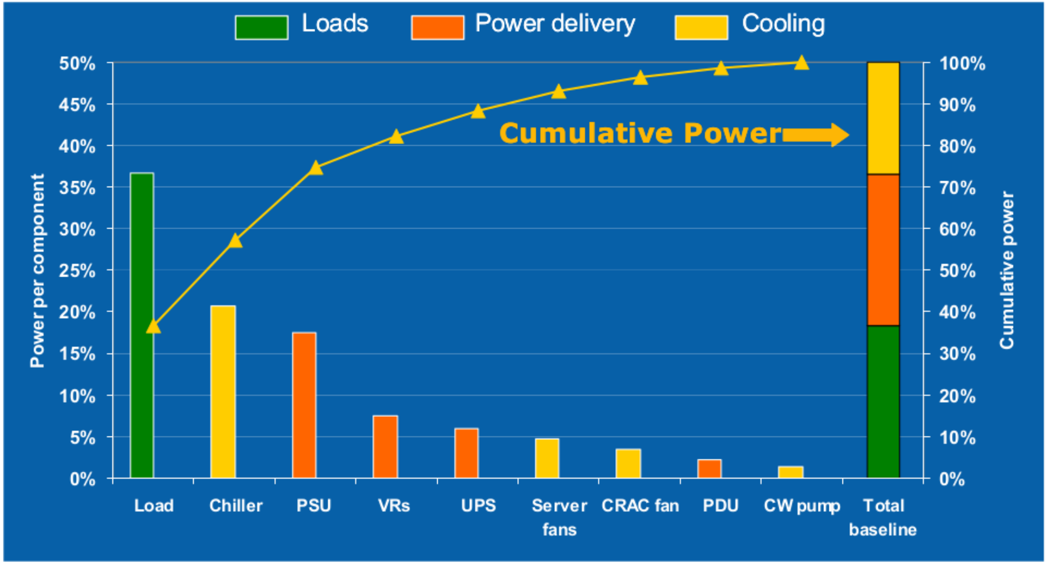
1. Defining *performances*
2. Measure *performances*
3. Some modern Hardware (Super Scalar CPU, Multi Cores, Caches, SIMD, GPU)
4. Something about Compilers (LLVM, gcc)
5. Something about libraries and languages (CUDA, OpenCL, BLAS)
6. Become a better programmer

Why

NEWS

Data centers are the new polluters

report



Source: Intel Corp.

Why

Does Anyone really cares?

CppCon 2014: Andrei Alexandrescu

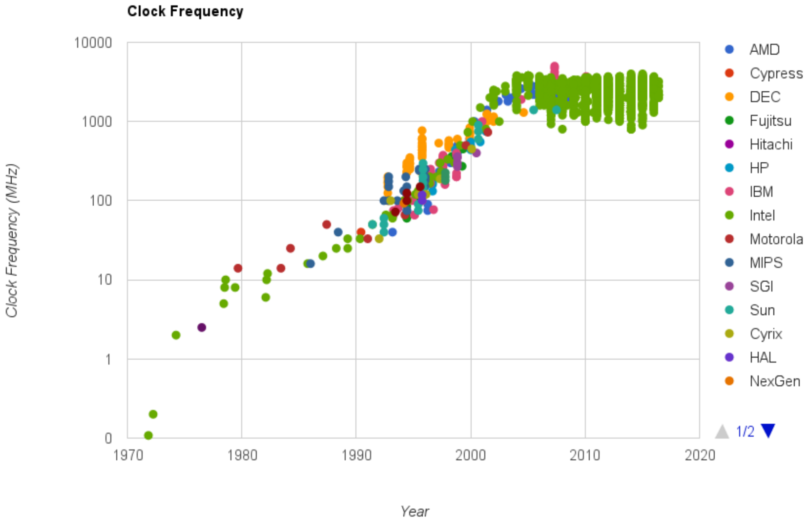
Performance Summary

The BIG Picture

$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

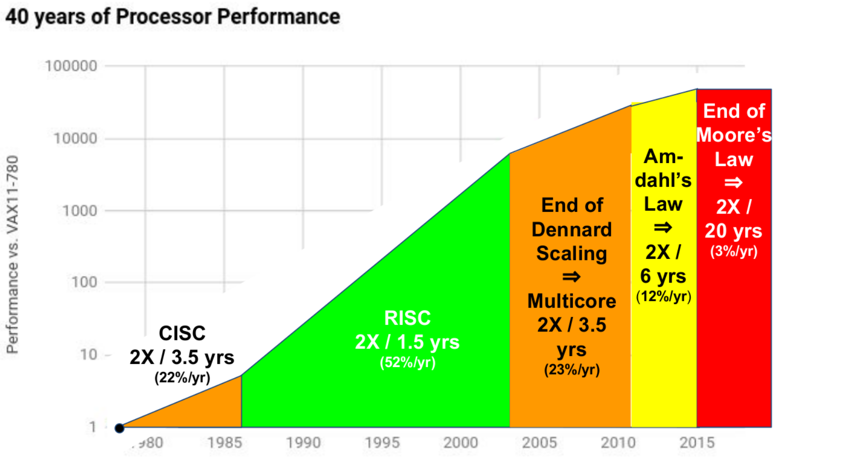
- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, T_c

Deeper reasoning



[source]

End of Growth of Performance?



Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

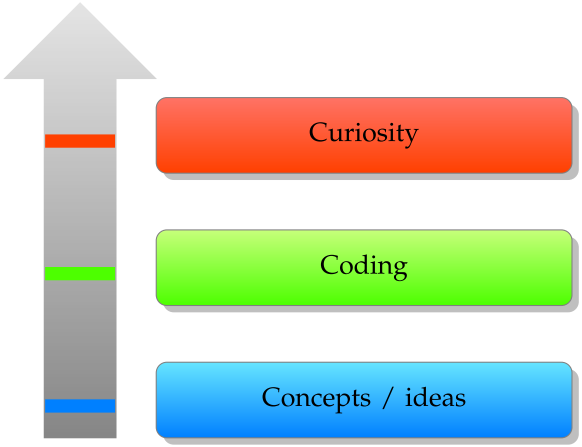
[Hennessy, Patterson 2018.]

How

How

- Theory presentations: 2h/w
- Lab work: 2h/w
- Your research and ideas
- May be some expert seminars
- Your personal effort (reading and watch video)
- Bruno and me are here to support and help (ask a meeting by email)
- It's all up to you

How

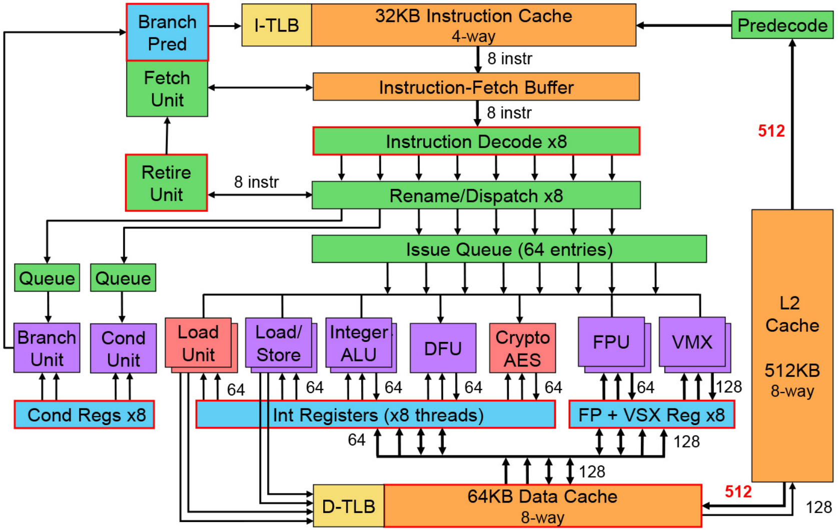


Create a mental model

- Before looking at code
- Model the principal components
- Set clear expectations:
  - Back of the envelop calculations https://github.com/sirupsen/napkin-math
  - Look for similar systems
- Refine your model with all available information (bayesian reasoning)
- Don't hate statistics...

FROM C ABSTRACT MACHINE TO HARDWARE

```
i += 3;
```



What you need

to know

- System Architecture (the basics)
- Operating System (SYE)++
- Programming in C (very well)
- A little of assembly is useful
- Linux
- version control (git, ...)

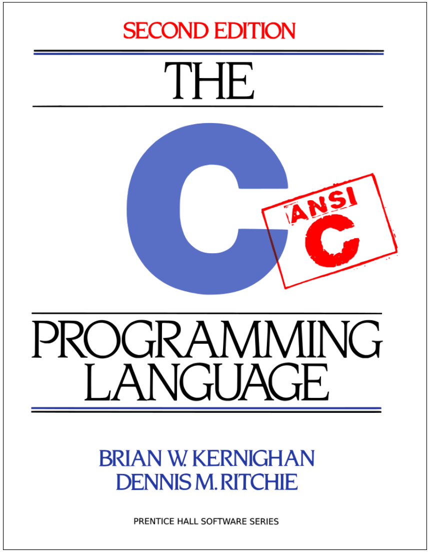
and...

- Curiosity
- Adapt your coding style
- Investigate and relentless improving
- read, read, test, read, ask, test, read again
- a little statistics ...

It is hard

- Very complex situations (each system is different)
- Hard to work in isolatation, reproduce
- Custom tools
- Concurrency (SW and HW/SW)
- Few methodologies

Unusual Disk Latency


SECOND EDITION

THE C PROGRAMMING LANGUAGE

ANSI C

BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

PRENTICE HALL SOFTWARE SERIES

Why C?

- Portable
- You are the master
- A lot of tools
- A lot of libraries
- Other languages can call high performance C libs

and

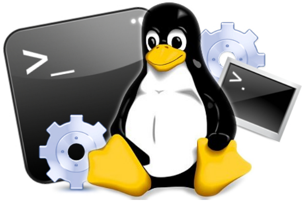
- OSes are written in C and a little assembler (yes, Windows and OSX too)
- GitHub
- Popularity

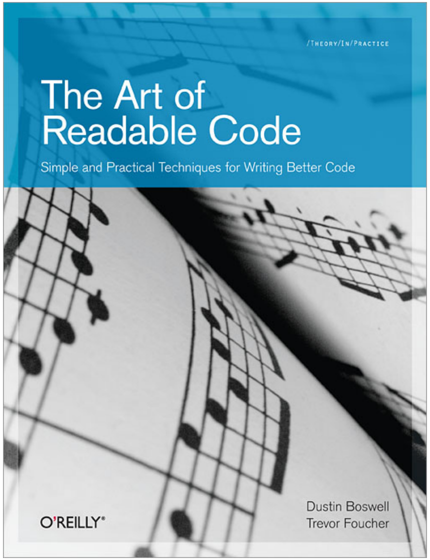
C can surprise you in many ways

- It’s very influencial
- It’s complex
- if you look at performances can shock you
- Cay Horstmann presentation is instructive

Why Linux?

- It's documented
- It's open
- Tons of tools
- Tons of libraries
- It's very common




/THEORY/IN/PRACTICE
The Art of Readable Code
Simple and Practical Techniques for Writing Better Code
O'REILLY
Dustin Boswell
Trevor Foucher

Please, please and please

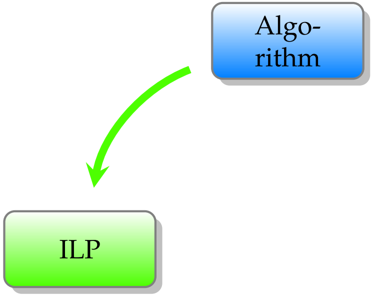
Write readable, litterate, clear and meaningful code!

Looking for performance cannot be used as an excuse for bad written code.

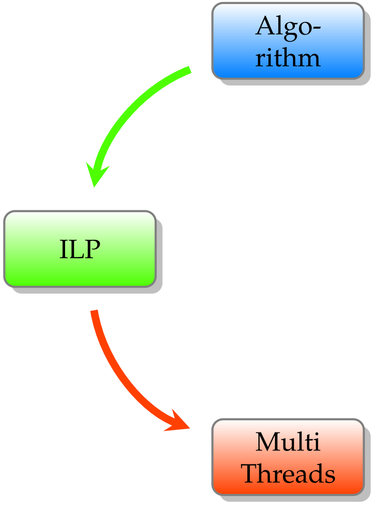
Some software engineering Best Practices can be derogated.

Where we optimize

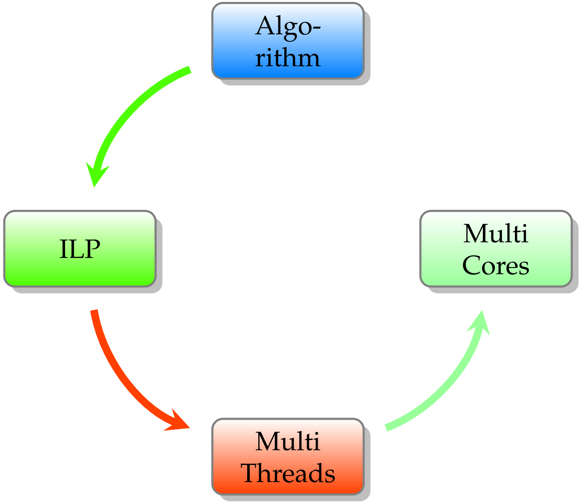
Where we optimize



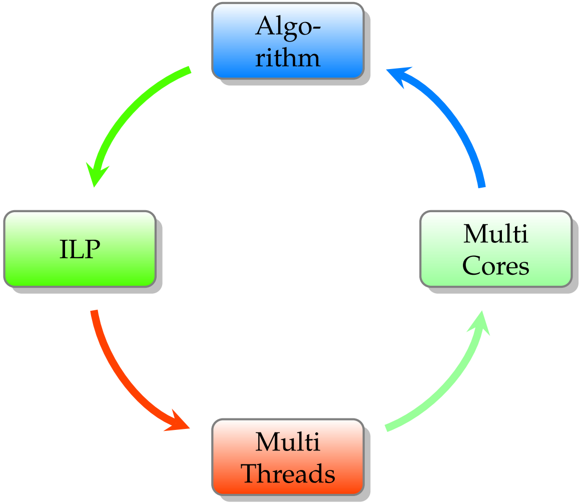
Where we optimize



WHERE WE OPTIMIZE



Where we optimize



References

https://datacenters.lbl.gov/resources/dc-power-improved-data-center-efficiency

Questions

?