

Int  ration Logiciel Architecture

Les performances et leur mesure

Henri-Pierre Charles Henri-Pierre.Charles@cea.fr &
 Fr  d  ric Rousseau Frederic.Rousseau@imag.fr

Metrics How Many ?

Metrics How Many ?

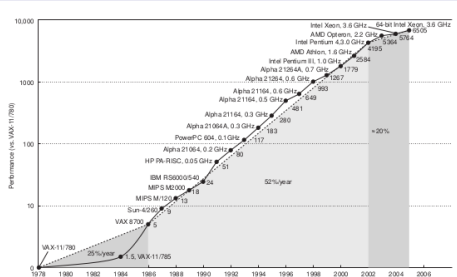
How many

- Computer you own ?
- Processor you own ?
- Computer you use ?
- Computing power ?

Metrics Moore

Metrics Moore

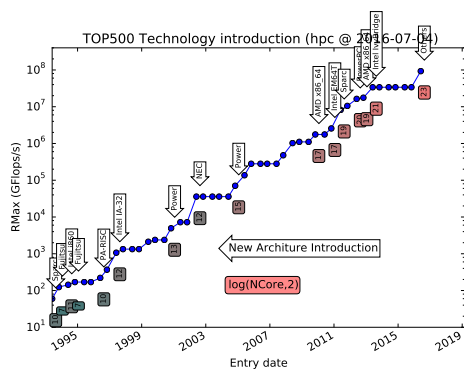
La loi de Moore



Growth in processor performance since the mid-1980s. This chart plots performance relative to the VAX 11/780 as measured by the SPECint benchmarks [Hennessy]

Metrics Top500

Metrics Top500



Lien entre l'introduction d'une nouvelle technologie et son impact sur les performances

1 Motivation

- Metrics How Many ?
- Metrics Moore
- Metrics Top500

2 Why

- Metrics How to Measure Performance ?
- Vocabulaire : Cycle par seconds / Flops
- Vocabulaire Peak / Sustained
- Vocabulaire Lois
- Vocabulaire Speedup
- Amdahl2

- Vocabulaire Amdahl
- Metrics Roofline model
- Metrics Roofline

3 How

- How : Outillage
- How : Static Compilation chain
- How : gprof
- How : PerfCounter
- How : JTAG
- How : gdb
- How : qemu
- How : CrossCompiler
- How : Stats

Metrics How to Measure Performance ?

What scale ?

- Application level (TPS, Frame/s, .../)
- Run to completion
- Method level
- Instruction level

Tools

- Wall clock
- gettimeofday
- Performance counters

Full application or function call ?

- Cold start / warmup
- Statistic / multiple calls
- How many calls

Vocabulaire : Cycle par seconds / Flops

Units

- Mips** Million operation per second
- Flops** Floating point operation per second
<http://www.top500.org>
- Flops/Watt** Floating point operation per watt per second
<http://www.green500.org>
- IPC** : Instructions per Cycle

How to mesure

- Analytique (wall clock)
- Instrumentation
 - "Portable" (gettimeofday())
 - Hardware (hardware performance counter)

Vocabulaire Peak / Sustained

Notions

- Peak** performance : maximal theoretical performance, assuming no bubble
- Sustain** performance : real acheived performance, on a real benchmark

Cost

- What is the percentage you're ready to lose ? 90% 95% ?
- How many are you ready to pay (time, money) to minimise this loss ?

<http://www.top500.org>

Vocabulaire Lois

Obey the law !

- Moore http://en.wikipedia.org/wiki/Moore's_law
- Amdahl http://en.wikipedia.org/wiki/Amdahl's_law
- Memory bound http://en.wikipedia.org/wiki/IO_bound
- CPU bound http://en.wikipedia.org/wiki/CPU_bound

Vocabulaire Speedup

Notion

Speedup $S = 100 * \frac{T_{seq}}{T_{opt}}$
Can be between

- 1 and N processor
- 1 and vectorized
- non optimized versus optimized version

Mesure Quality what are the execution conditions : data set, computer workload, reproducibly, ...

Computer science has to use “human science” tools & methodology

Amdahl2

Argumentation

Assume that a task has two independent parts, A and B. B takes roughly 25% of the time of the whole computation. By working very hard, one may be able to make this part 5 times faster, but this only reduces the time for the whole computation by a little. In contrast, one may need to perform less work to make part A be twice as fast. This will make the computation much faster than by optimizing part B, even though B's speed-up is greater by ratio, (5x versus 2x)

Illustration

Two independent parts **A** **B**

Original process 

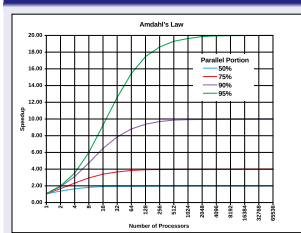
Make **B** 5x faster 

Vocabulaire Amdahl

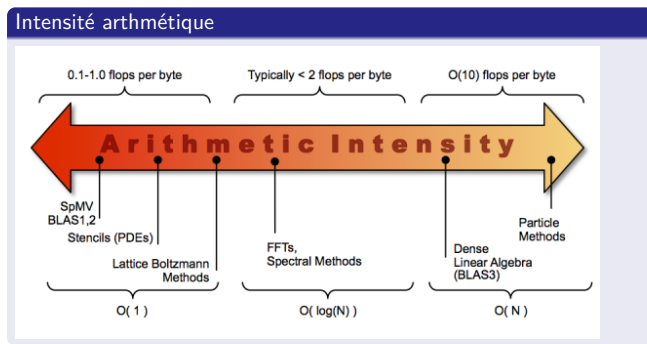
Argumentation

The speedup of a program using multiple processors in parallel computing is limited by the sequential fraction of the program. For example, if 95% of the program can be parallelized, the theoretical maximum speedup using parallel computing would be 20x as shown in the diagram, no matter how many processors are used.

Illustration



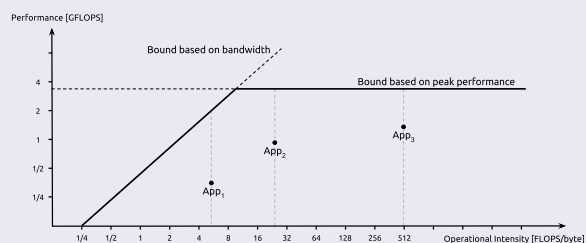
Metrics Roofline model



(From <https://crd.lbl.gov/departments/computer-science/PAR/research/roofline/>)

Metrics Roofline

Arithmetic intensity versus performance



https://en.wikipedia.org/wiki/Roofline_model

How : Outillage

Outillage pour la performance

- HW related
 - gprof
 - gdb
- Simulation
 - Performance counters
 - JTAG
- Analytical
 - Statical analysis

How : Static Compilation chain

Static compilation (on C language) :

- | | |
|---|-------|
| ① Preprocessor (all # stuff : rewriting) | cc -E |
| ② Compilation (from C to textual assembly) | cc -S |
| ③ Assembly (from textual asm to binary asm) | cc -c |
| ④ Executable (binary + dynamic library) | |

Optional

- Profiling : Compile (use -pg) produce File ; Run File ; Use gprof
- cc -da dump all intermediate representation

(Use gcc -v to see all the steps)

Don't stop at static time (Operating system + processor) : Load in memory, dynamic linking ; Branch resolution ; Cache warmup

Argumentation

- Compile using -pg option
gcc -pg -o prog
prog.c, add information
at the begin / end of each
function
- Run the code ./prog
(create the gmon.out file)
- Run gprof gprof prog
- Read report !

Pitfalls

- Warning with -O3
interaction !
- Use -O0 as first pass
- Use -O3 for real

Limitations

- Not fine grain
- Measure perturbation

Performance counter from HW

- Simple one : cycle
counter
- More complex :
RAT_STALLS Counts the
number of cycles during
which execution stalled
due to several reason
- L2, L3 cache access, hit,
miss ...
- Miss prediction, etc

Tools

- Intel tools :
<http://www.intel.com/software/pcm>
- Papi : <http://icl.cs.utk.edu/papi>
- Tiptop from Inria :
<http://tiptop.gforge.inria.fr/>

Limitations

- Complicated to
understand / analyse
- No real portable library

Question

- Comment calculate-t-on

How : JTAG

Description

- Hardware connector
- Able to get processor
internal information
- Embedded domain

Drawback

- Need HW access
- Impossible to get large
configuration



How : gdb

Debugger

- Source level debugger
- Compile with -g (gcc -g
-o p p.c)
- Use gdb program (or ddd
or emacs or ...)
- Step execution : next,
previous line, next,
previous insn
- Inspect live data
- Set breakpoint, on line,
on insn, on data access

Other usage

- .gdbinit : automatize
test
- Embed python into gdb :
program information
collection

Question

- Why -O3 could perturb
gdb ?
- How could gdb stop an
execution ?

How : qemu

Qemu

- Instruction level simulator
- Emulate processor and peripheral
- Modes System & User

Usage

- Functional simulation
- Trace information

<http://qemu.org>

How : CrossCompiler

What

- Allow to generate binary code for a machine A on a machine B
- How to build ?
-

Limitation

- crosstool-ng, buildroot
- llvm
- FreeBSD, NetBSD

How : Stats

Are execution reproducible?

- Reproducibility !
- Why ?

Methodology

- Statistical approach
- Best run, multiples run (how many)
- Extrema elimination
- Means
- Cold start, warmnup