High Performance Computing: Towards Better Performance Predictions and Experiments

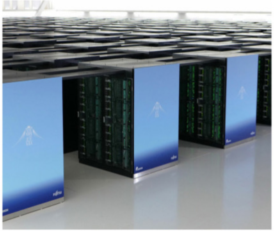
Tom Cornebize

2 June 2021, PhD defense

No science without computing

Arithmomètre (1851)

ENIAC (1945)

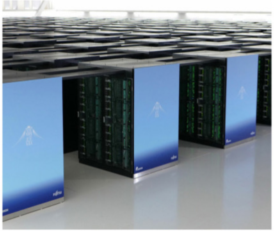


Fugaku (2021)

No science without computing

Arithmomètre (1851)

ENIAC (1945)

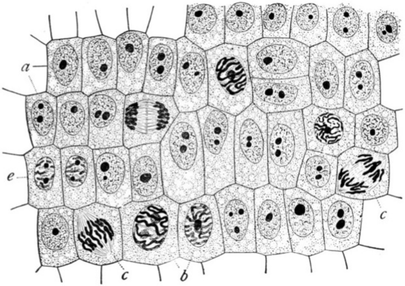


Fugaku (2021)

Last decades:

- Exponential performance improvements (e.g. sequencing an entire human genome costed $100,000,000 in 2001, $1000 now)
- At the price of complexity (both software and hardware)

Experimental study of computer performance



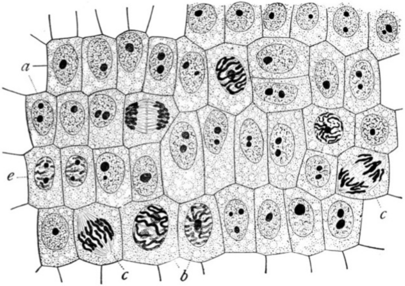
Similar to natural sciences

Complexity $\Rightarrow$ Variability and Opacity

$\Rightarrow$ No perfect model

$\Rightarrow$ Need for experiments

Experimental study of computer performance



Similar to natural sciences

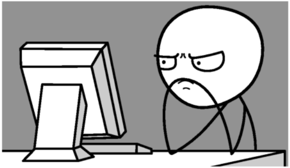
Complexity ⇒ Variability and Opacity

⇒ No perfect model

⇒ Need for experiments

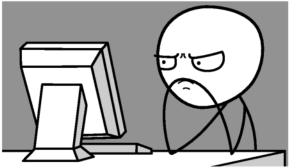
Empirical studies can be carried in reality or in simulation

Typical Performance Evaluation Questions (Given my application and a supercomputer)



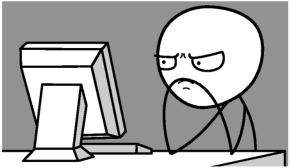
- **Before** running
  - How many nodes?
  - For how long?
  - Which parameters?

Typical Performance Evaluation Questions (Given my application and a supercomputer)



- **Before** running
  - How many nodes?
  - For how long?
  - Which parameters?
- **After** running
  - Performance as "expected"?
  - Problem in the app or the platform?

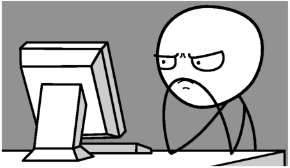
Typical Performance Evaluation Questions (Given my application and a supercomputer)



- **Before** running
  - How many nodes?
  - For how long?
  - Which parameters?
- **After** running
  - Performance as “expected”?
  - Problem in the app or the platform?

So many large-scale runs, solely to tune performance?!?

Typical Performance Evaluation Questions (Given my application and a supercomputer)



- **Before** running
  - How many nodes?
  - For how long?
  - Which parameters?
- **After** running
  - Performance as "expected"?
  - Problem in the app or the platform?

So many large-scale runs, solely to tune performance?!?

Holy Grail: Predictive Simulation on a "Laptop"

Capture the **whole application** and **platform complexity**

Initial goal: predict the performance of a parallel application

Initial goal: predict the performance of a parallel application

Thesis contributions (towards this goal)

- Case study: High Performance Linpack (HPL)
- Extensive (in)validation, comparing simulations with reality
- Modeling correctly the platform variability is key

Initial goal: predict the performance of a parallel application

Thesis contributions (towards this goal)

- Case study: High Performance Linpack (HPL)
- Extensive (in)validation, comparing simulations with reality
- Modeling correctly the platform variability is key

Thesis contributions (made on the way)

- Automation (of experiments, statistical analyzes, etc.)
- Experiment methodology, to bias or not to bias
- Performance tests, to detect eventual platform changes

Initial goal: predict the performance of a parallel application

Thesis contributions (towards this goal)

- Case study: High Performance Linpack (HPL)
- Extensive (in)validation, comparing simulations with reality
- Modeling correctly the platform variability is key

Thesis contributions (made on the way)

- Automation (of experiments, statistical analyzes, etc.)
- Experiment methodology, to bias or not to bias
- Performance tests, to detect eventual platform changes

Performance prediction through simulation

Full reimplementation of MPI on top of SIMGRID

- C/C++/F77/F90 codes run unmodified out of the box
- Simply replace mpicc/mpirun by smpicc/smpirun

Full reimplementation of MPI on top of SIMGRID

- C/C++/F77/F90 codes run unmodified out of the box
- Simply replace mpicc/mpirun by smpicc/smpirun

Emulation: how?

- Application runs for real on a laptop
- Communications are faked, good fluid network models
- Performance model for the target platform

Full reimplementation of MPI on top of SIMGRID

- C/C++/F77/F90 codes run unmodified out of the box
- Simply replace mpicc/mpirun by smpicc/smpirun

Emulation: how?

- Application runs for real on a laptop
- Communications are faked, good fluid network models
- Performance model for the target platform

Contribution: Skip the expensive computations (mostly `dgemm`) and replace them by performance models

Quick word on HPL

Validations of SMPI before this thesis: simple applications without any high performance tricks

Quick word on HPL

Validations of SMPI before this thesis: simple applications without any high performance tricks

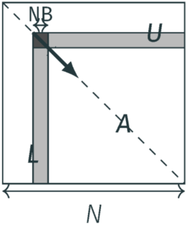
Contribution: predict accurately the performance of HPL

- Computations and communication overlap
- More representative of some HPC workloads
- Well established, used for the Top500

Validations of SMPI before this thesis: simple applications without any high performance tricks

Contribution: predict accurately the performance of HPL

- Computations and communication overlap
- More representative of some HPC workloads
- Well established, used for the Top500



Allocate and initialize A

for $k = N$ ***to*** 0 ***step*** NB **do**

  Allocate the panel

  Factor the panel

  Broadcast the panel

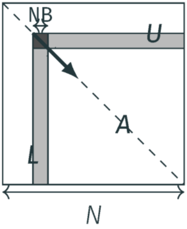
  Update the sub-matrix

Quick word on HPL

Validations of SMPI before this thesis: simple applications without any high performance tricks

Contribution: predict accurately the performance of HPL

- Computations and communication overlap
- More representative of some HPC workloads
- Well established, used for the Top500



Allocate and initialize A
for $k = N$ ***to*** 0 ***step*** NB **do**
    Allocate the panel
    Factor the panel
    Broadcast the panel
    Update the sub-matrix

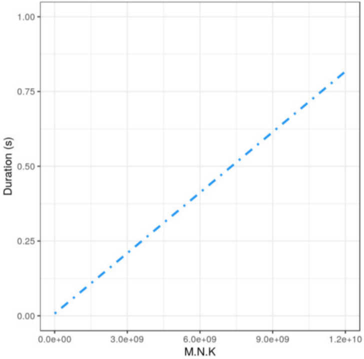
Tuning parameters

- Broadcast algorithm
- Block size
- Process grid
- etc.

Hundreds of combinations

MODELING COMMPUTATIONS

$$\texttt{dgemm}(M, N, K) = \alpha.M.N.K$$



Modeling Commputations

$\texttt{dgemm}\,(M, N, K) = \alpha.M.N.K$