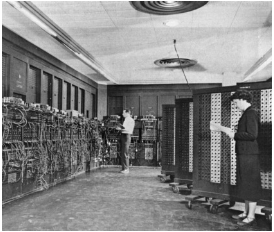
High Performance Computing: Towards Better Performance Predictions and Experiments

Tom Cornebize

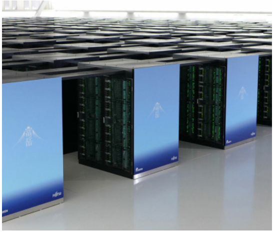
2 June 2021, PhD defense

No science without computing

Arithmomètre (1851)



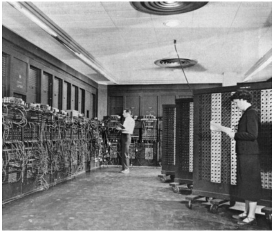
ENIAC (1945)



Fugaku (2021)

No science without computing

Arithmomètre (1851)



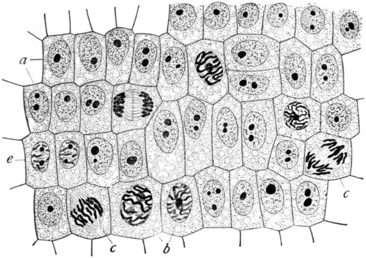
ENIAC (1945)

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

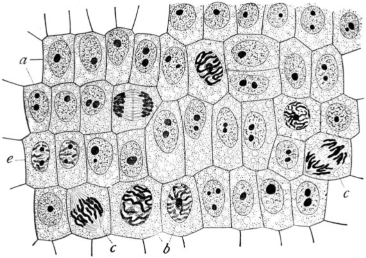
Complexity

⇒ Variability and Opacity

⇒ No perfect model

⇒ Need for experiments

Experimental study of computer performance



Similar to natural sciences

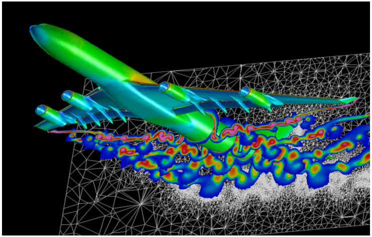
Complexity

$\Rightarrow$ Variability and Opacity

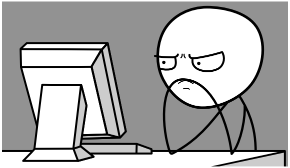
$\Rightarrow$ No perfect model

$\Rightarrow$ Need for experiments

Empirical studies can be carried in reality or in simulation

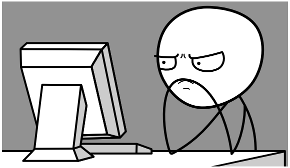


Typical Performance Evaluation Questions (Given my application and a supercomputer)



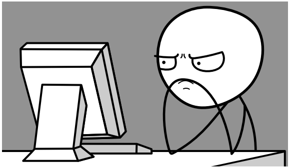
- Before running
  - How many nodes?
  - For how long?
  - Which parameters?

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?

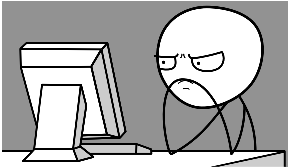
Typical Performance Evaluation Questions (Given my application and a supercomputer)



- **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
- Demonstrate it is possible to predict faithfully the behavior of complex parallel applications
- 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
- Demonstrate it is possible to predict faithfully the behavior of complex parallel applications
- 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
- Demonstrate it is possible to predict faithfully the behavior of complex parallel applications
- 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

Sim(Em)ulation: The SMPI Approach

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

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

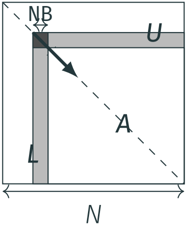
Quick word on HPL

TOP 500
The List.

- Computations and communication overlap (custom collectives)
- More representative of some HPC applications
- Well established, used for the Top500

Quick word on HPL

- Computations and communication overlap (custom collectives)
- More representative of some HPC applications
- Well established, used for the Top500

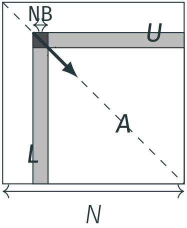


Allocate and initialize A

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

    Allocate the panel
    Factor the panel
    Broadcast the panel
    Update the sub-matrix

- Computations and communication overlap (custom collectives)
- More representative of some HPC applications
- 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

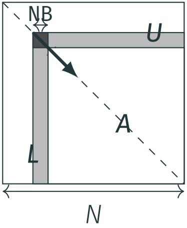
Tuning parameters

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

Hundreds of combinations

Quick word on HPL

- Computations and communication overlap (custom collectives)
- More representative of some HPC applications
- 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

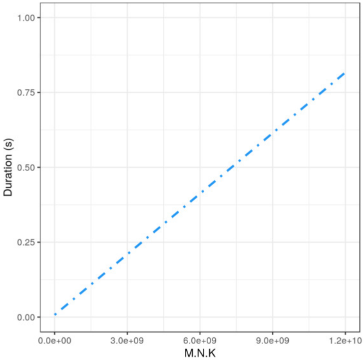
Tuning parameters

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

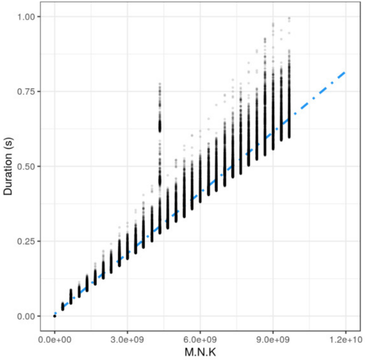
Hundreds of combinations

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

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

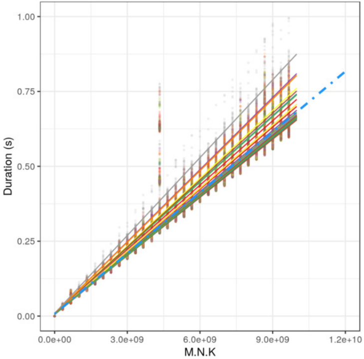


$$\texttt{dgemm}\,(M, N, K) = \alpha.M.N.K$$



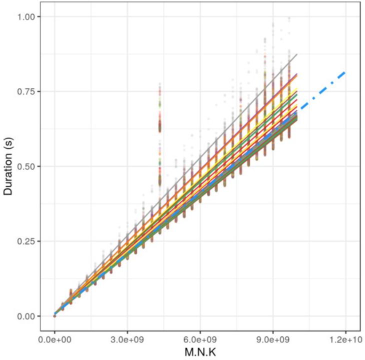
$$\mathtt{dgemm}_i(M, N, K) = \underbrace{\alpha_i.M.N.K}_{\text{per host}}$$

Different color $\Rightarrow$ different host

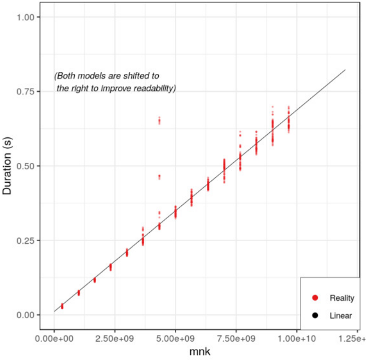


$$\mathtt{dgemm}_i(M,N,K) = \underbrace{\alpha_i.M.N.K}_{\text{per host}}$$

Different color $\Rightarrow$ different host



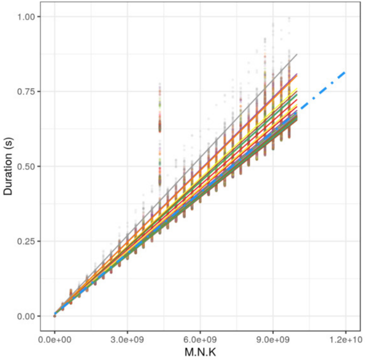
For a particular host



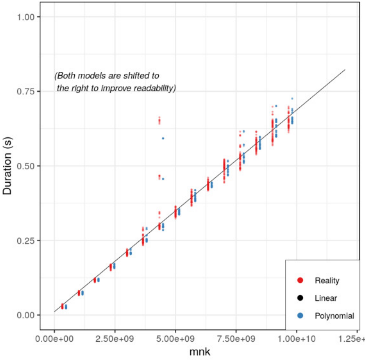
Modeling Commputations

$$\text{dgemm}_i(M,N,K) = \underbrace{\alpha_i.M.N.K}_{\text{per host}} + \underbrace{\beta_i.M.N + \gamma_i.N.K + \ldots}_{\text{polynomial model}}$$

Different color ⇒ different host

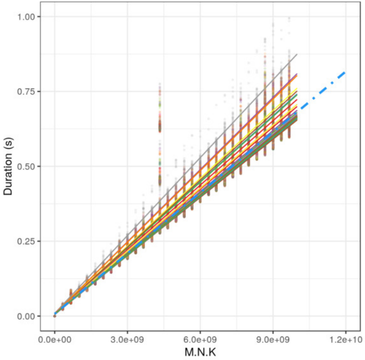


For a particular host

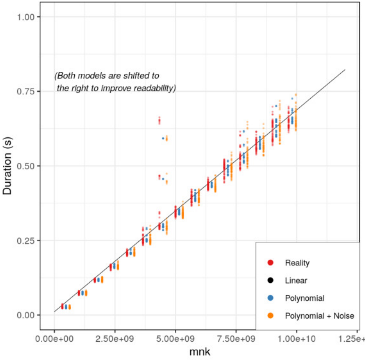


$$\texttt{dgemm}_i(M,N,K) = \underbrace{\alpha_i.M.N.K}_{\text{per host}} + \underbrace{\beta_i.M.N + \gamma_i.N.K + \dots}_{\text{polynomial model}} + \underbrace{\mathcal{N}(0, \alpha'_i.M.N.K + \dots)}_{\text{polynomial noise}}$$

Different color $\Rightarrow$ different host

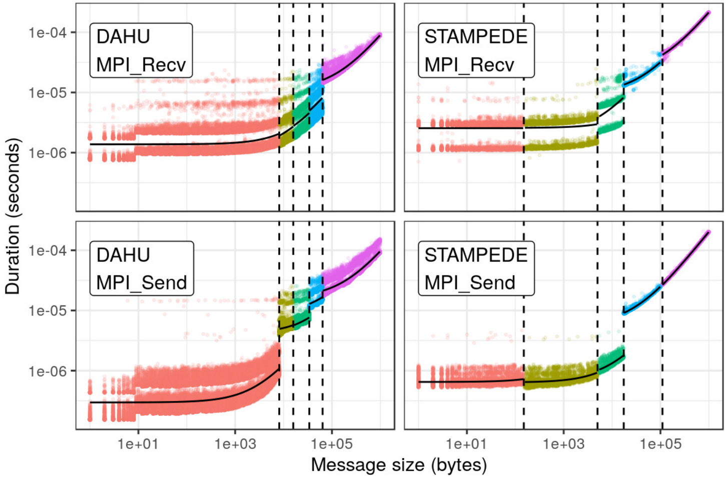


For a particular host



Hand-crafted non-blocking collective operations intertwinned with computations

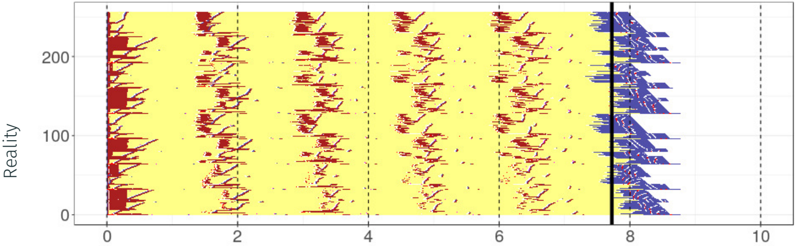
Hand-crafted non-blocking collective operations intertwinned with computations



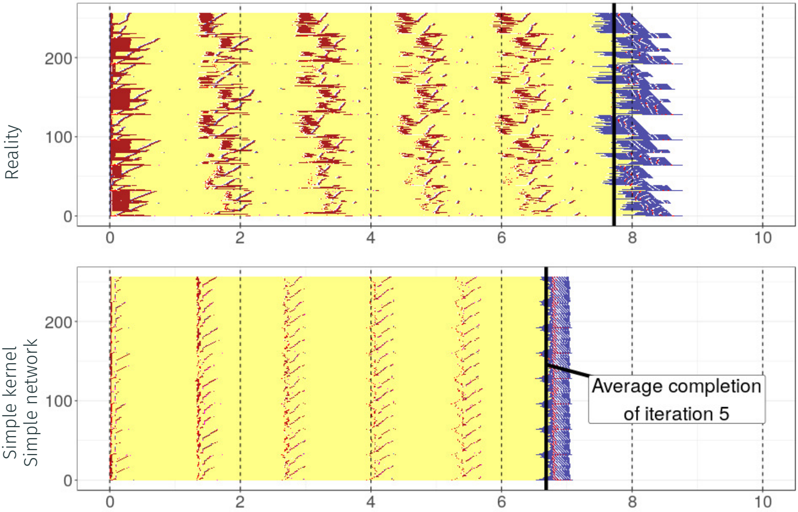
Validating the predictions

Internal behavior of the application

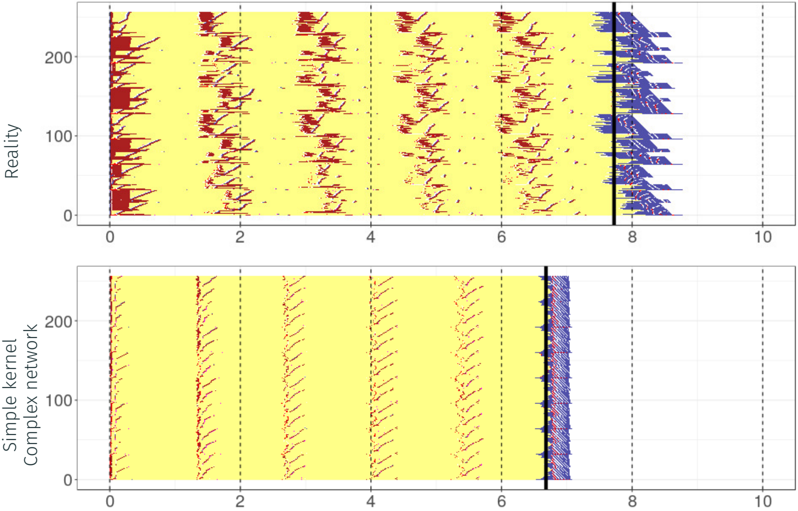
256 MPI processes, interrupted after the 5th iteration



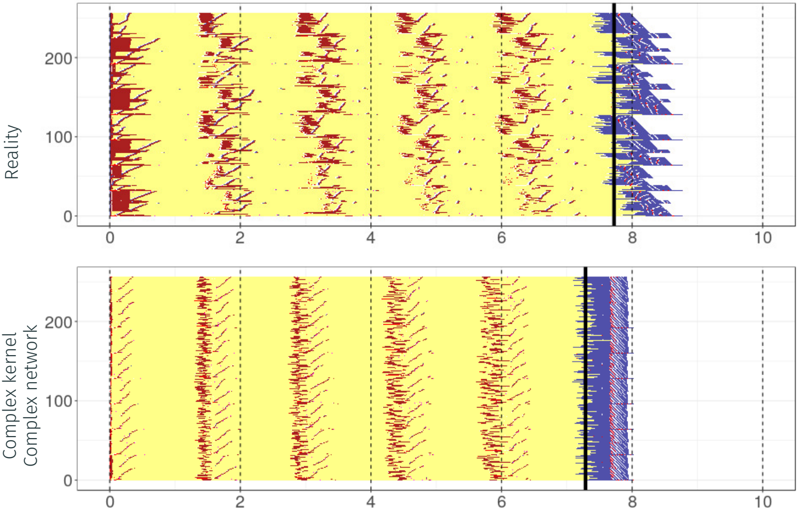
256 MPI processes, interrupted after the 5th iteration



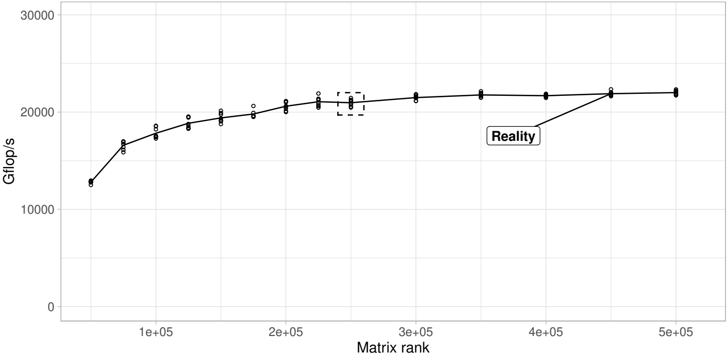
256 MPI processes, interrupted after the 5^{th} iteration



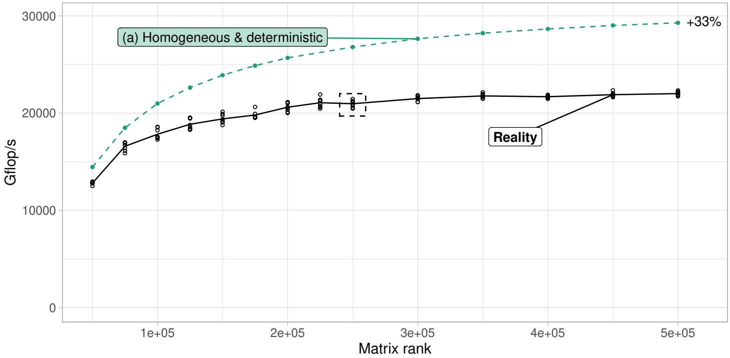
256 MPI processes, interrupted after the 5th iteration



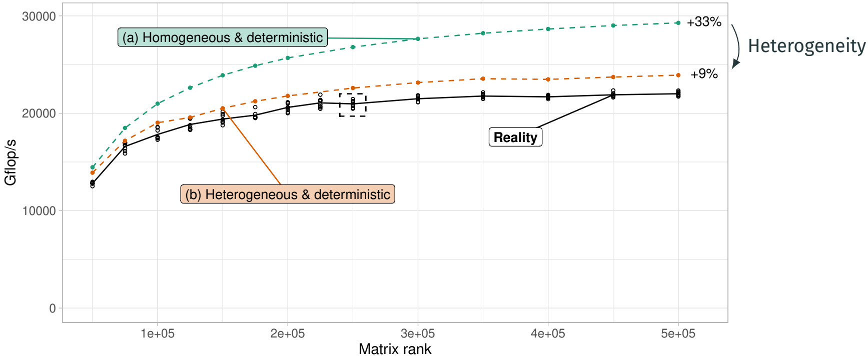
Now the complete run, with 1024 MPI processes



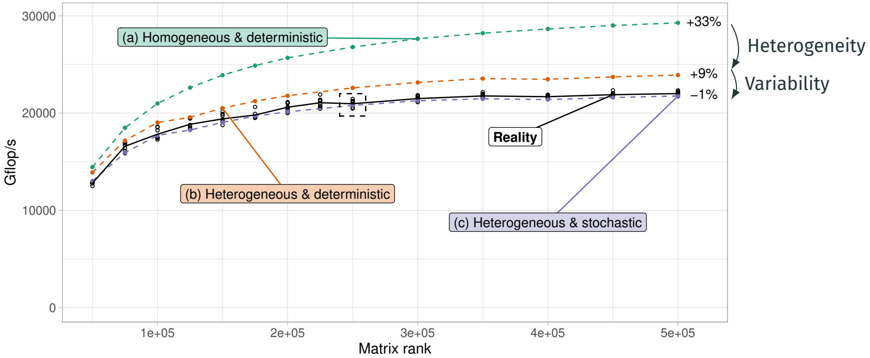
Now the complete run, with 1024 MPI processes



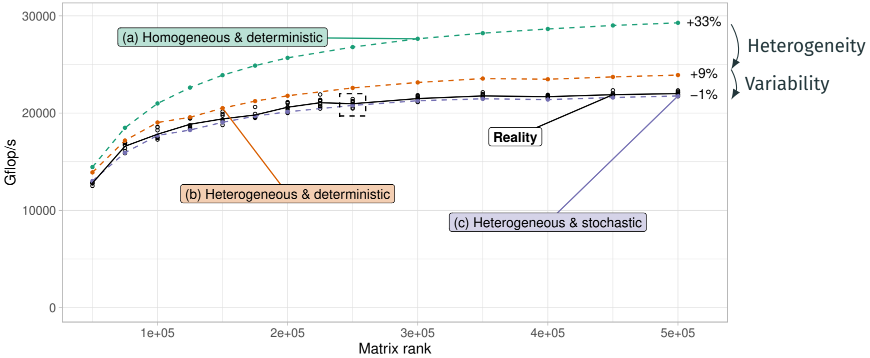
Now the complete run, with 1024 MPI processes



Now the complete run, with 1024 MPI processes



Now the complete run, with 1024 MPI processes

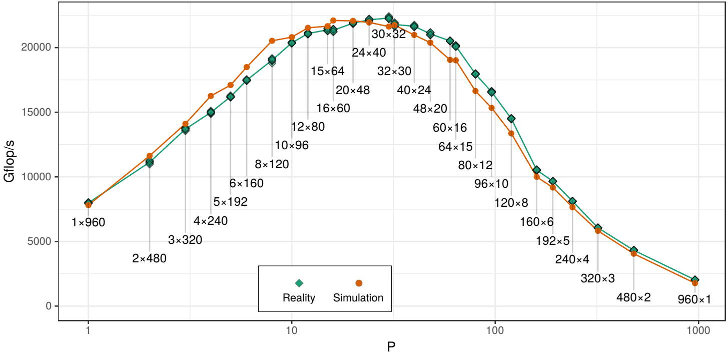


Take-Away Message: accurate prediction

Modeling both spatial and temporal computation variability is essential

Influence of the geometry

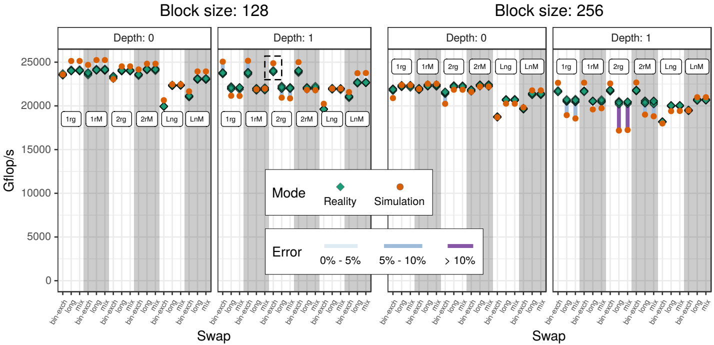
$P \times Q$ MPI processes, organized in a 2D grid



Perspective: geometry tuning in simulation

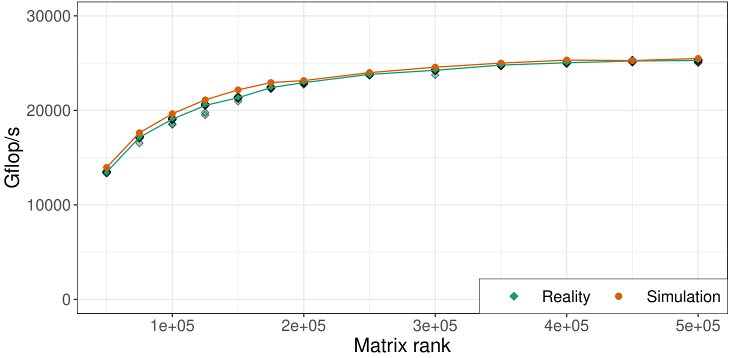
Influence of the other parameters

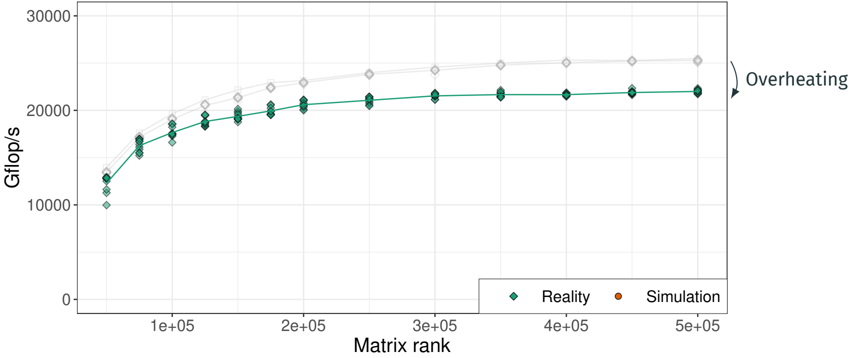
Tested the 72 combinations of the remaining parameters



Perspective: parameter tuning in simulation

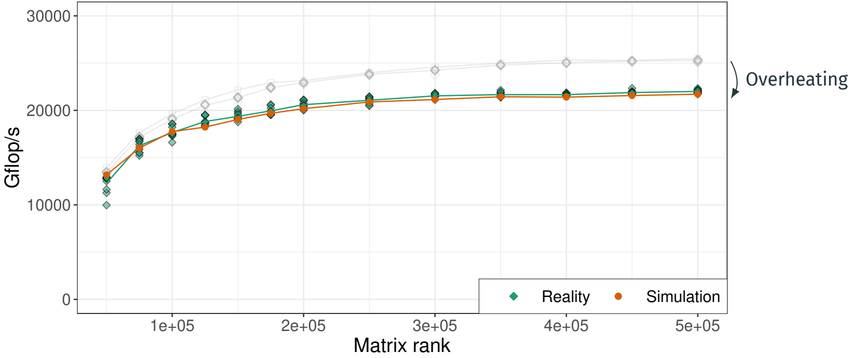
Influence of a platform change



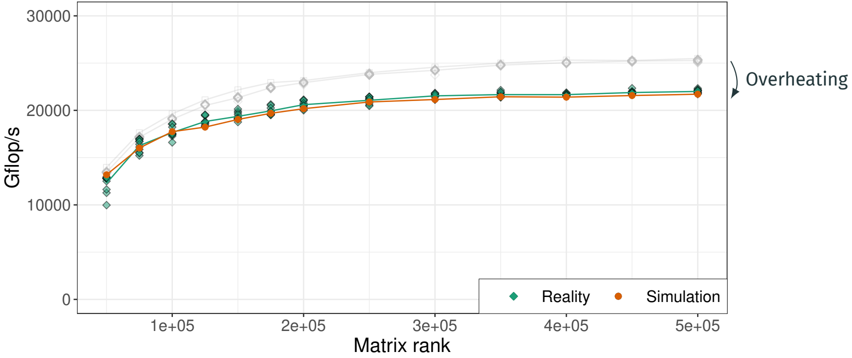


On four nodes, the cooling system malfunctionned for several weeks

Influence of a platform change



On four nodes, the cooling system malfunctionned for several weeks

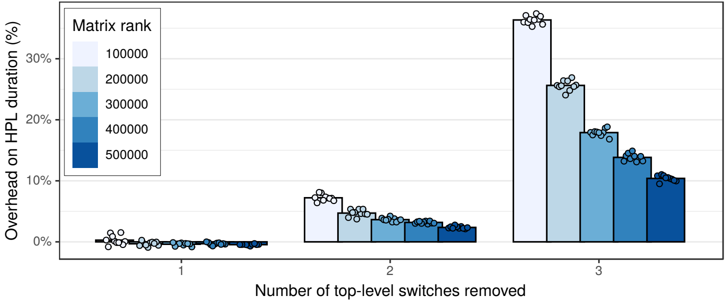


On four nodes, the cooling system malfunctionned for several weeks

Take-Away Message: Re-measuring `dgemm` durations to generate a new model was enough to account for the platform change

What if the network topology of my cluster was different?

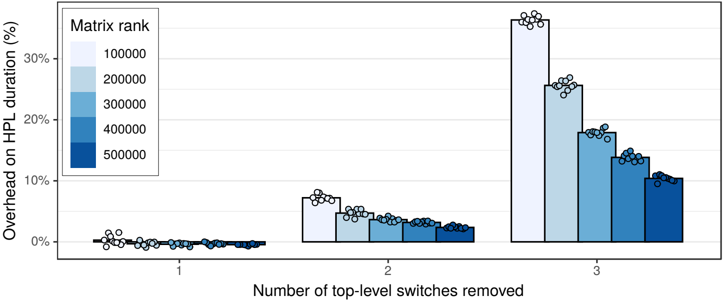
Study: take a 2-level fat tree with 4 top-level switches, remove them one by one



Use case: sensibility analysis

What if the network topology of my cluster was different?

Study: take a 2-level fat tree with 4 top-level switches, remove them one by one



Faithful surrogate $\Rightarrow$ Empirical studies of hypothetical platforms

$\Rightarrow$ Extrapolation of existing platforms

Goal: performance prediction ✓

Goal: performance prediction ✓

Main difficulties:

- Experimentation/calibration
- Platform changes (e.g., the cooling issue)

Experimental biases when measuring `dgemm` or MPI durations

Effect on durations, but also other metrics (e.g. CPU frequency)

On the difficulties of experimentation

Experimental biases when measuring `dgemm` or MPI durations

Effect on durations, but also other metrics (e.g. CPU frequency)

- Sampling method for generating the sequence of calls
  $\sim$ 10% systematic performance change

Experimental biases when measuring `dgemm` or MPI durations

Effect on durations, but also other metrics (e.g. CPU frequency)

- Sampling method for generating the sequence of calls
  $\sim$ 10% systematic performance change
- Interferences between computations and communications
  $\sim$ 50% performance change in extreme configurations

On the difficulties of experimentation

Experimental biases when measuring `dgemm` or MPI durations

Effect on durations, but also other metrics (e.g. CPU frequency)

- Sampling method for generating the sequence of calls
  $\sim$ 10% systematic performance change
- Interferences between computations and communications
  $\sim$ 50% performance change in extreme configurations
- Content of the matrices used by `dgemm`
  $\sim$ 5% systematic performance change

On the difficulties of experimentation

Experimental biases when measuring `dgemm` or MPI durations

Effect on durations, but also other metrics (e.g. CPU frequency)

- Sampling method for generating the sequence of calls
  $\sim$ 10% systematic performance change
- Interferences between computations and communications
  $\sim$ 50% performance change in extreme configurations
- Content of the matrices used by `dgemm`
  $\sim$ 5% systematic performance change

Take-Away Message:

- These biases could only be identified with a solid experimental methodology with heavy use of randomization
- Bias may be desirable, to increase prediction accuracy

Continuous platform modeling

On a near-daily basis, run the `dgemm` calibration code on Grid'5000

454 nodes (792 CPU) from 12 clusters, $\sim$ 30 min. jobs

On a near-daily basis, run the `dgemm` calibration code on Grid'5000

454 nodes (792 CPU) from 12 clusters, $\sim$ 30 min. jobs

For each CPU, collect:

- average `dgemm` performance
- `dgemm` coefficients of regression (i.e. the model for simulation)

Regular measures

On a near-daily basis, run the `dgemm` calibration code on Grid'5000

454 nodes (792 CPU) from 12 clusters, $\sim$ 30 min. jobs

For each CPU, collect:

- average `dgemm` performance
- `dgemm` coefficients of regression (i.e. the model for simulation)
- average CPU frequency
- average CPU power consumption
- average DRAM power consumption
- average temperature

On a near-daily basis, run the `dgemm` calibration code on Grid'5000

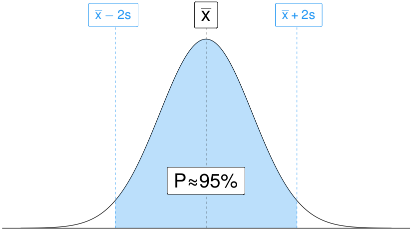
454 nodes (792 CPU) from 12 clusters, $\sim$ 30 min. jobs

For each CPU, collect:

- average `dgemm` performance
- `dgemm` coefficients of regression (i.e. the model for simulation)
- average CPU frequency
- average CPU power consumption
- average DRAM power consumption
- average temperature

If the platform did not change, then each parameter is normally distributed (thanks to CLT)

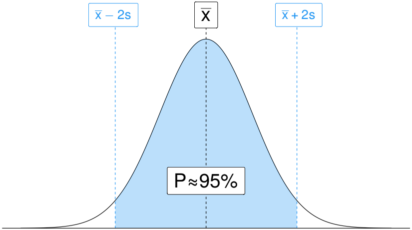
Given a sequence of old observations $x_1, \ldots, x_n$ and a new observation x_{n+1}, how likely was it to observe x_{n+1}?



Take the sample mean $\bar{x}$ and sample standard deviation s of the old observations

$$\mathbb{P}\left(x_{n+1} \in [\bar{x} - 2s; \bar{x} + 2s]\right) \approx 95\%$$

Given a sequence of old observations $x_1, \ldots, x_n$ and a new observation x_{n+1}, how likely was it to observe x_{n+1}?



Take the sample mean $\bar{x}$ and sample standard deviation s of the old observations

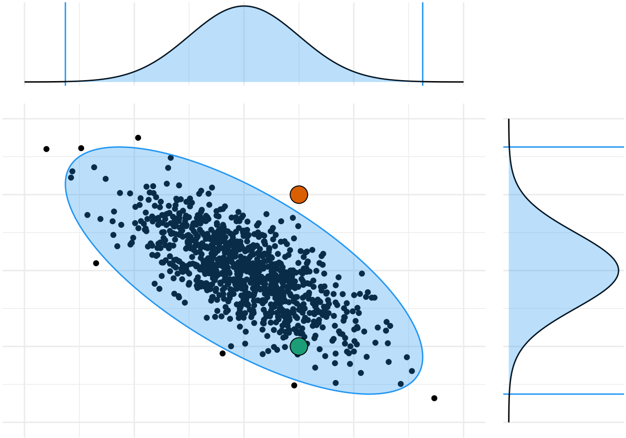
$$\mathbb{P}\left(x_{n+1} \in [\bar{x} - 2s; \bar{x} + 2s]\right) \approx 95\%$$

Note: using the F distribution instead of the normal distribution (the true mean and standard deviation are unknown)

Fluctuation interval for several variables

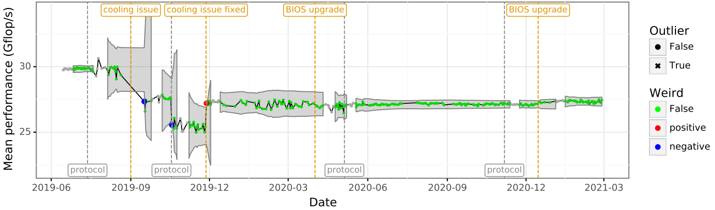
With several variables, use their covariance matrix

Example in dimension 2, with $\mathbb{P}(x_{n+1} \in \text{interval}) \approx 99.5\%$



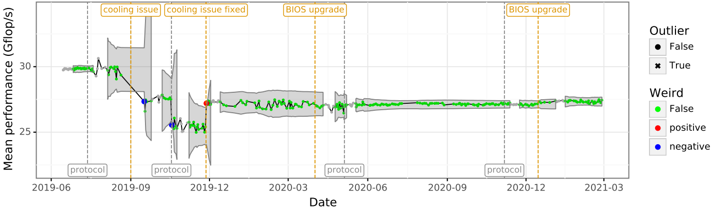
Result: performance fluctuation

Performance fluctuation of the node dahu-14

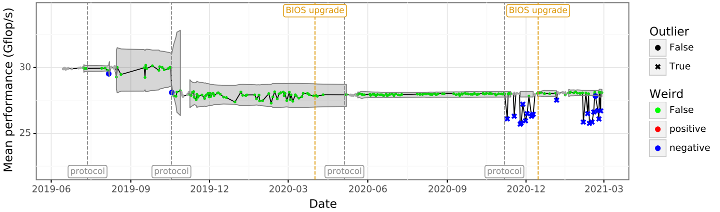


Result: performance fluctuation

Performance fluctuation of the node dahu-14



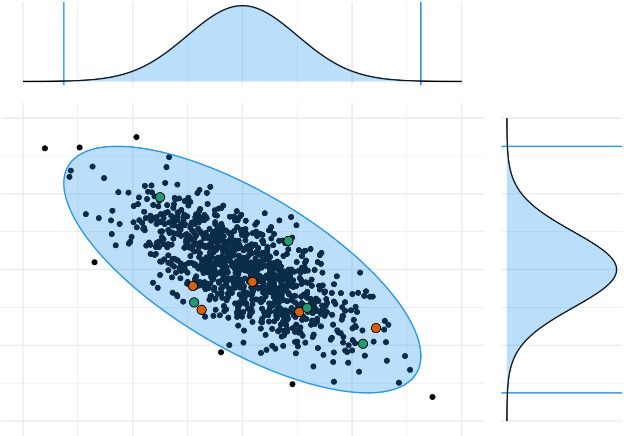
Performance fluctuation of the node dahu-32



How to detect more subtle changes? Take several consecutive measures $x_{n+1}, \ldots, x_{n+k}$, use their average and shrink the interval accordingly

How to detect more subtle changes? Take several consecutive measures $x_{n+1}, \ldots, x_{n+k}$, use their average and shrink the interval accordingly

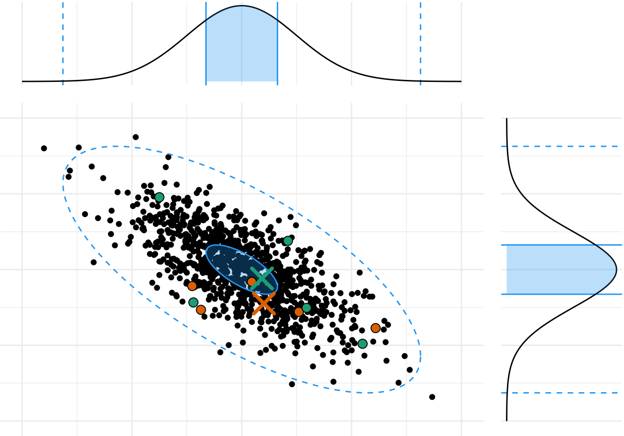
Example with 5 measures



Fluctuation interval for several measures

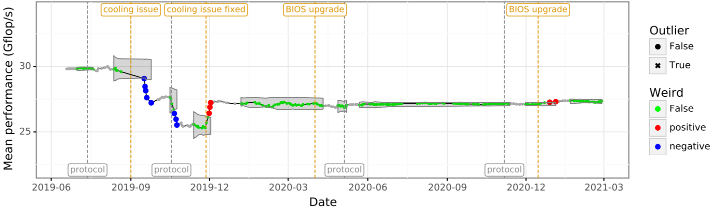
How to detect more subtle changes? Take several consecutive measures $x_{n+1}, \ldots, x_{n+k}$, use their average and shrink the interval accordingly

Example with 5 measures (averages represented by crosses)

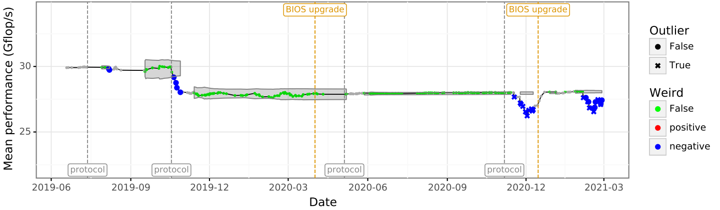


Result: performance fluctuation

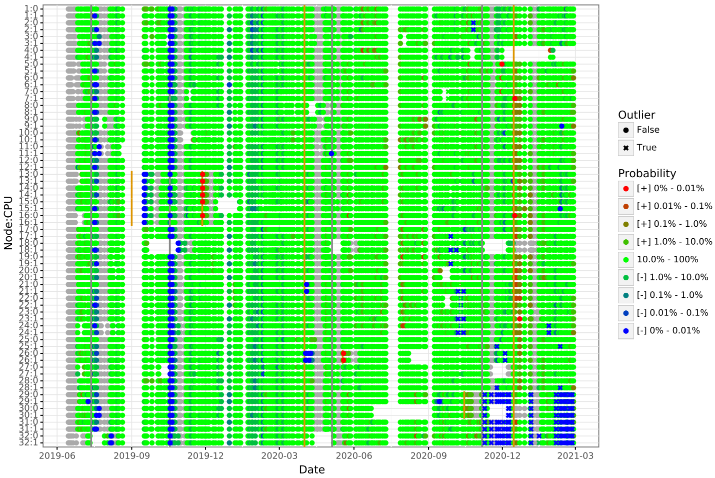
Performance fluctuation of the node dahu-14 (5-day window)



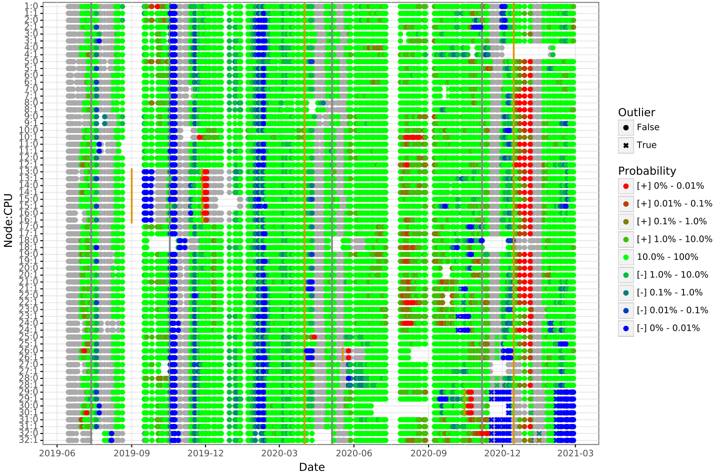
Performance fluctuation of the node dahu-32 (5-day window)



Overview of the performance on cluster dahu



Overview of the performance on cluster dahu (5-day window)



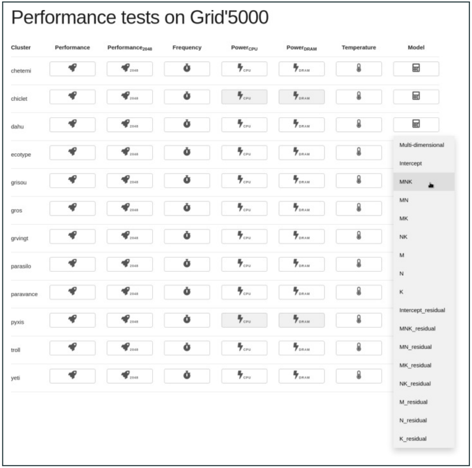
Performance tests: wraping up

Multi-variable test also implemented, on all the model coefficients

Performance tests: wraping up

Multi-variable test also implemented, on all the model coefficients

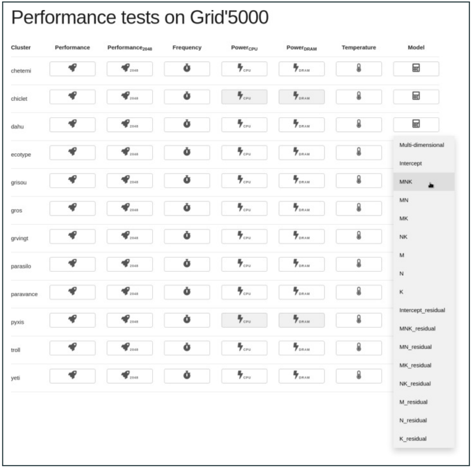
Results available at https://cornebize.net/g5k_test



Performance tests: wraping up

Multi-variable test also implemented, on all the model coefficients

Results available at `https://cornebize.net/g5k_test`

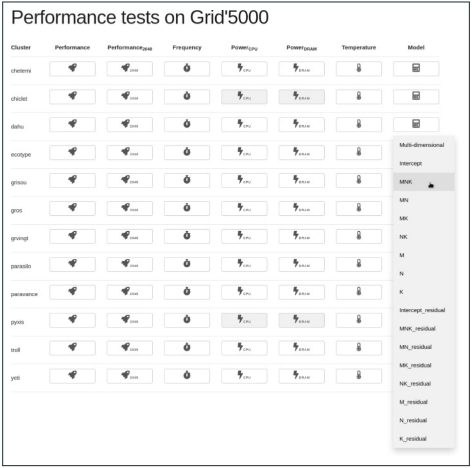


Detected events

- BIOS upgrades
- Cooling issue
- Faulty memory
- Power instability

Multi-variable test also implemented, on all the model coefficients

Results available at `https://cornebize.net/g5k_test`



Detected events

- BIOS upgrades
- Cooling issue
- Faulty memory
- Power instability

All went unnoticed by both Grid'5000 staff and users, despite significant effects

$\Rightarrow$ Great help potential

Concluding thoughts

How to know if our predictions are faithful?

There is no *correctness proof*, a model can be validated only by trying to break it

Can we trust our predictions?

How to know if our predictions are faithful?

There is no *correctness proof*, a model can be validated only by trying to break it

$\Rightarrow$ Often broke the simulations during the last few years

Can we trust our predictions?

How to know if our predictions are faithful?

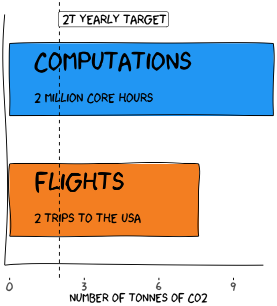
There is no *correctness proof*, a model can be validated only by trying to break it

$\Rightarrow$ Often broke the simulations during the last few years

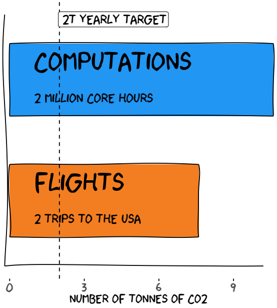
Repeated the whole study from scratch on a new cluster: ✓

Where to stop? Try all the Grid'5000 clusters? Other applications?

About 18 t of CO2eq were emitted for this thesis



About 18 t of CO2eq were emitted for this thesis



Do we really *need* to attend conferences in person?

What about computations?

Why so many computations?

More than half the total core hours were used for performance tests

Why so many computations?

More than half the total core hours were used for performance tests

How to reduce them?

- Change the experiment procedure (e.g. no full node reinstallation)
- Test less frequently (e.g. only once a week)
- Use a cheaper test (e.g. shorter warmup, less extensive coverage)

Why so many computations?

More than half the total core hours were used for performance tests

How to reduce them?

- Change the experiment procedure (e.g. no full node reinstallation)
- Test less frequently (e.g. only once a week)
- Use a cheaper test (e.g. shorter warmup, less extensive coverage)

Who should be responsible of tests?

- Platform staff? But what should they test?
- Researchers? Isn't it redundant?

Applying our approach on the whole life cycle of supercomputers:

Design Constructing the best machine for a given budget, using co-design

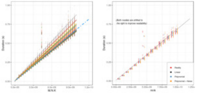
Development Debugging and improving software performance

Maintenance Ensuring that routine upgrades keep the performance as expected

Modeling Commputations

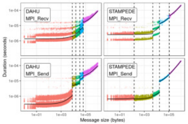
$$\text{dgemm}_i(M,N,K) = \underbrace{\alpha_i.M.N.K}_{\text{per host}} + \underbrace{\beta_i.M.N + \gamma_i.N.K + \ldots}_{\text{polynomial model}} + \underbrace{\mathcal{N}(0, \alpha'_i.M.N.K + \ldots)}_{\text{polynomial noise}}$$

Different color ⇒ different host For a particular host



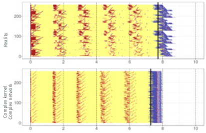
Modeling Communications

Hand-crafted non-blocking collective operations intertwinned with computations



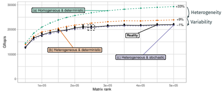
Internal behavior of the application

256 MPI processes, interrupted after the 5[th] iteration



Influence of the problem size

Now the complete run, with 1024 MPI processes

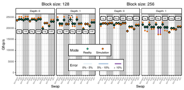


Take-Away Message: accurate prediction

Modeling both spatial and temporal computation variability is essential

Influence of the other parameters

Tested the 72 combinations of the remaining parameters

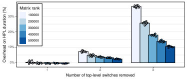


Perspective: parameter tuning in simulation

Use case: sensibility analysis

What if the network topology of my cluster was different?

Study: take a 2-level fat tree with 4 top-level switches, remove them one by one



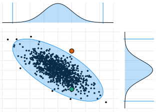
Faithful surrogate ⇒ Empirical studies of hypothetical platforms

⇒ Extrapolation of existing platforms

Fluctuation interval for several variables

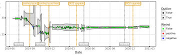
With several variables, use their covariance matrix

Example in dimension 2, with $\mathbb{P}(x_{n+1} \in \text{interval}) \approx 99.5\%$

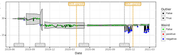


Result: performance fluctuation

Performance fluctuation of the node dahu-14

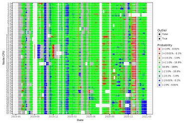


Performance fluctuation of the node dahu-32



Result: performance overview

Overview of the performance on cluster dahu (5-day window)



Thank you all!