

HIGH PERFORMANCE COMPUTING: TOWARDS BETTER PERFORMANCE PREDICTIONS AND EXPERIMENTS

Tom Cornebize

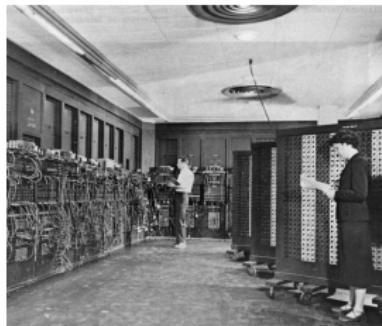
2 June 2021, PhD defense



No SCIENCE WITHOUT COMPUTING



Arithmomètre (1851)



ENIAC (1945)

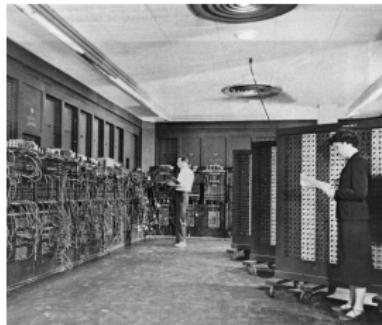


Fugaku (2021)

No SCIENCE WITHOUT COMPUTING



Arithmomètre (1851)



ENIAC (1945)

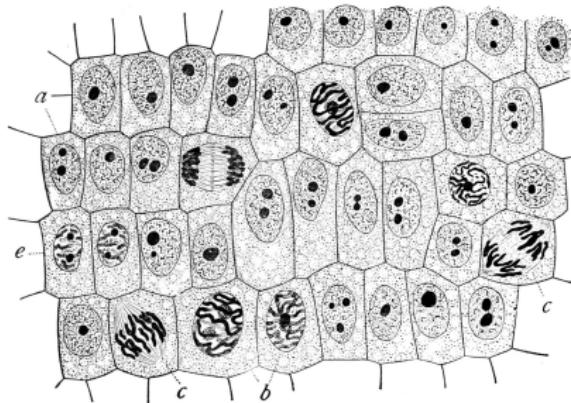


Fugaku (2021)

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



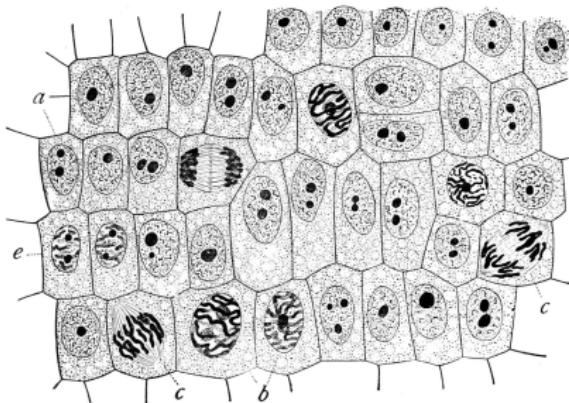
Similar to natural sciences

Complexity ⇒ Variability and Opacity

⇒ No perfect model

⇒ Need for [experiments](#)

EXPERIMENTAL STUDY OF COMPUTER PERFORMANCE



Similar to natural sciences

Complexity ⇒ Variability and Opacity

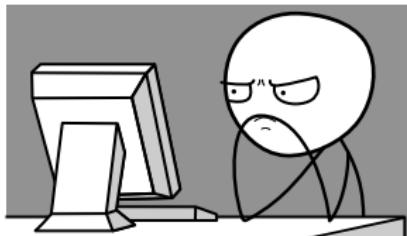
⇒ No perfect model

⇒ Need for experiments

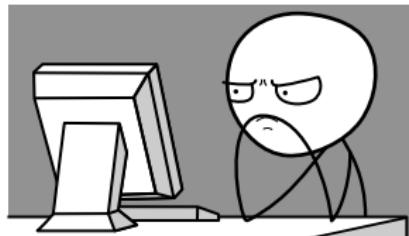
Empirical studies can be carried in reality or in simulation

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

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

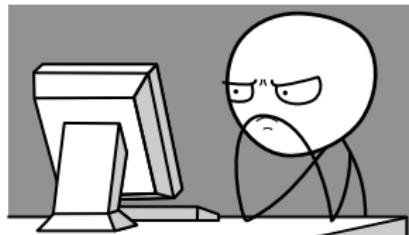


Typical Performance Evaluation Questions (Given my application and a supercomputer)



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

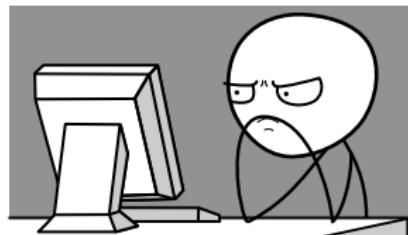
Typical Performance Evaluation Questions (Given my application and a supercomputer)



- **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

SIM(EM)ULATION: THE SMPI APPROACH



Full reimplementation of MPI on top of



- 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 

- 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

SIMULATION: THE SMPI APPROACH



Full reimplementation of MPI on top of 

- 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

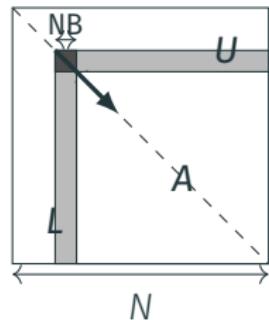
QUICK WORD ON HPL

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
```

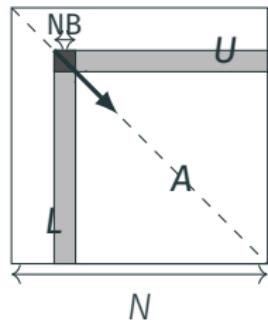
QUICK WORD ON HPL

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

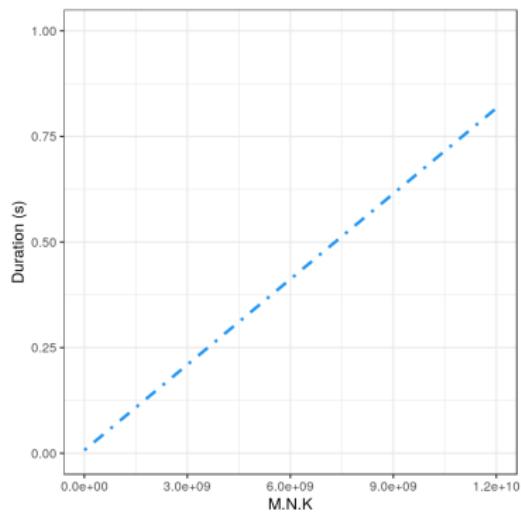
Tuning parameters

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

Hundreds of combinations

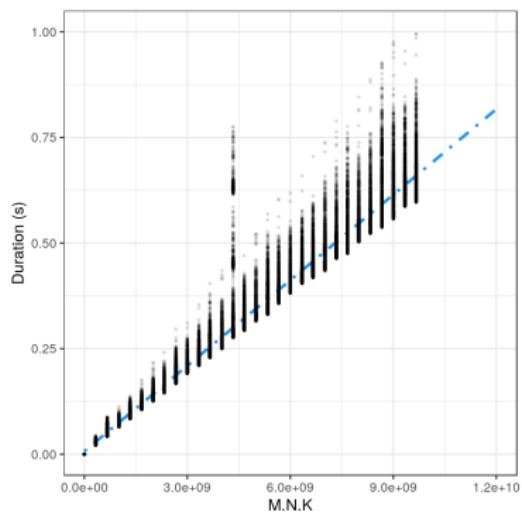
MODELING COMPUTATIONS

$$\text{dgemm } (M, N, K) = \alpha \cdot M \cdot N \cdot K$$



MODELING COMPUTATIONS

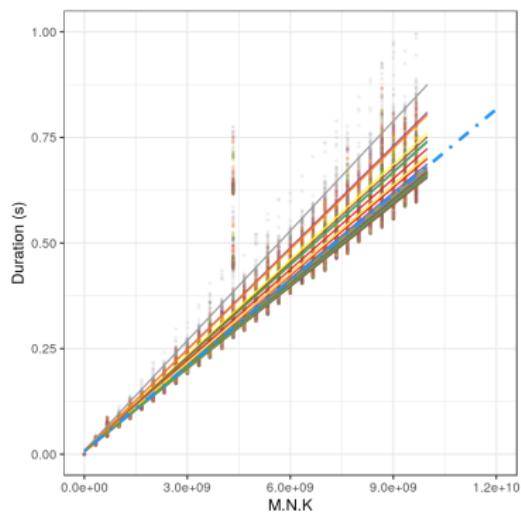
$$\text{dgemm } (M, N, K) = \alpha \cdot M \cdot N \cdot K$$



MODELING COMPUTATIONS

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

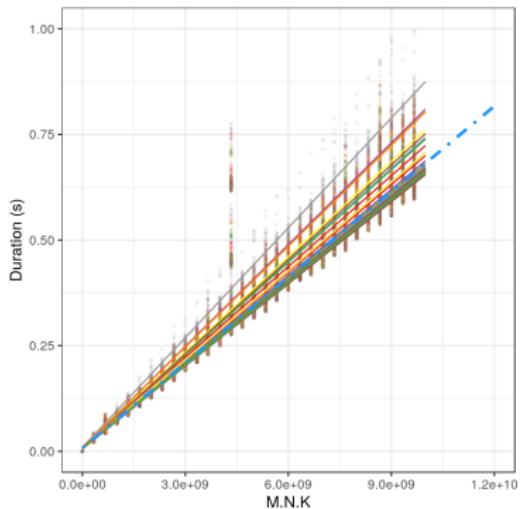
Different color \Rightarrow different host



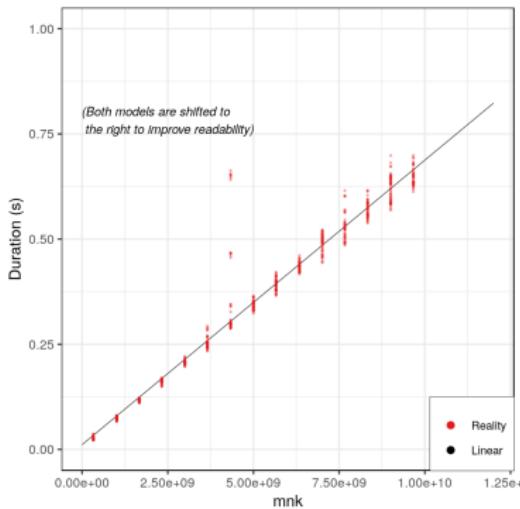
MODELING COMPUTATIONS

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

Different color \Rightarrow different host



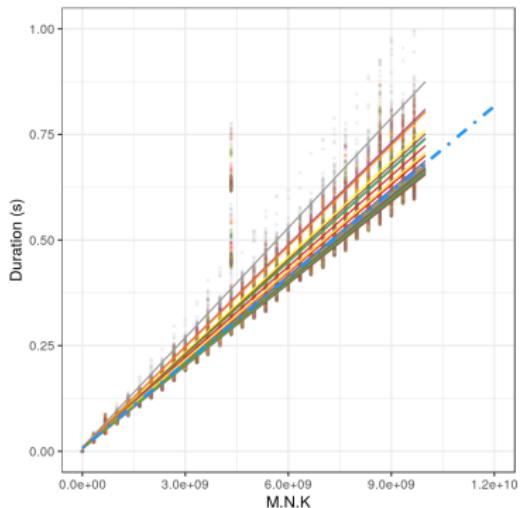
For a particular host



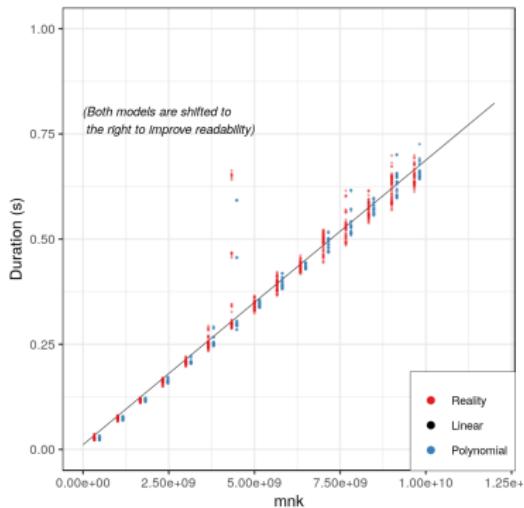
MODELING COMPUTATIONS

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

Different color \Rightarrow different host



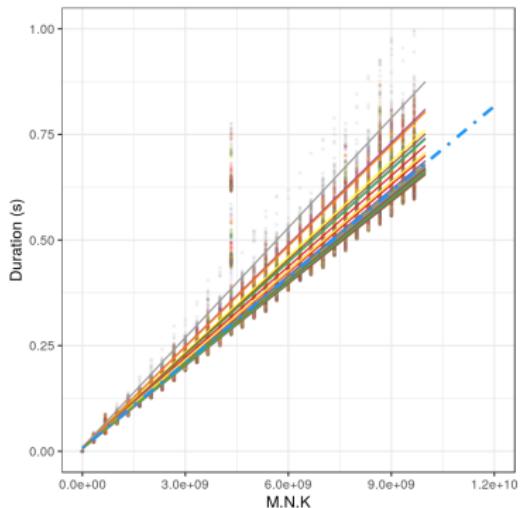
For a particular host



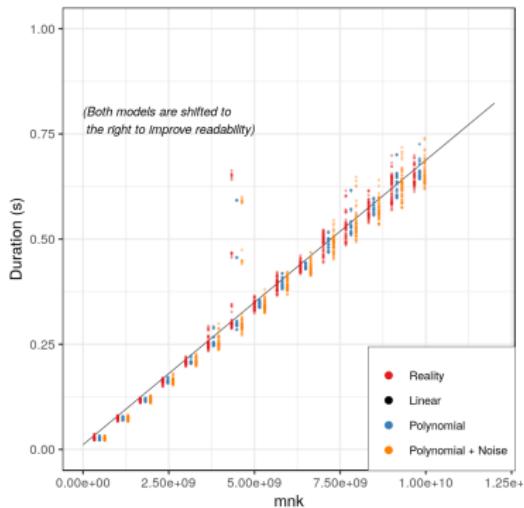
MODELING COMPUTATIONS

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

Different color \Rightarrow different host



For a particular host

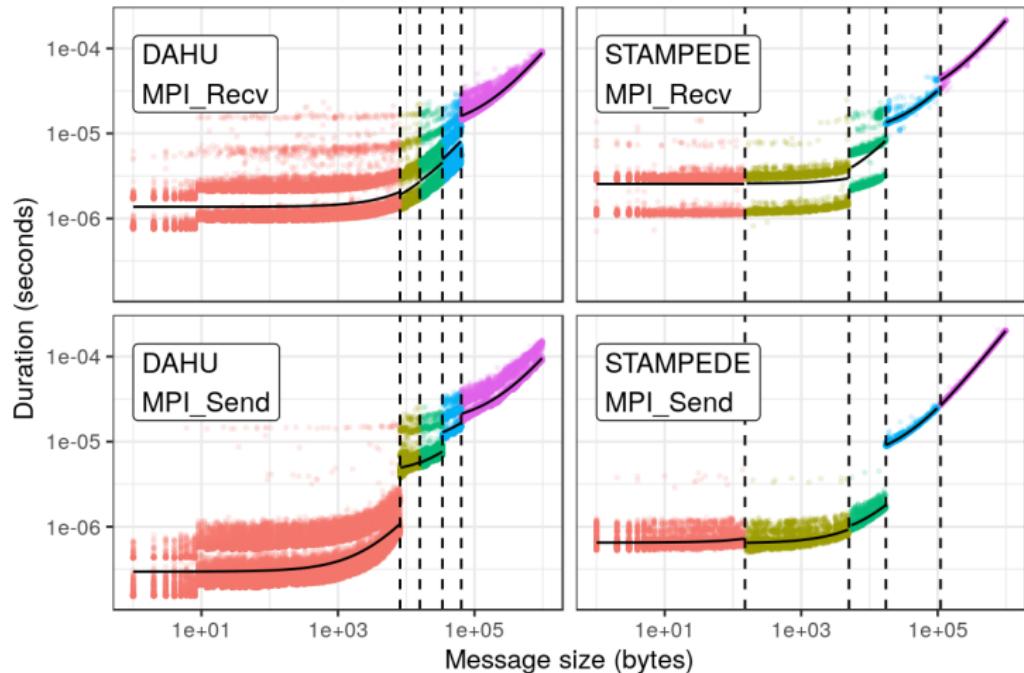


MODELING COMMUNICATIONS

Hand-crafted non-blocking collective operations intertwined with computations

MODELING COMMUNICATIONS

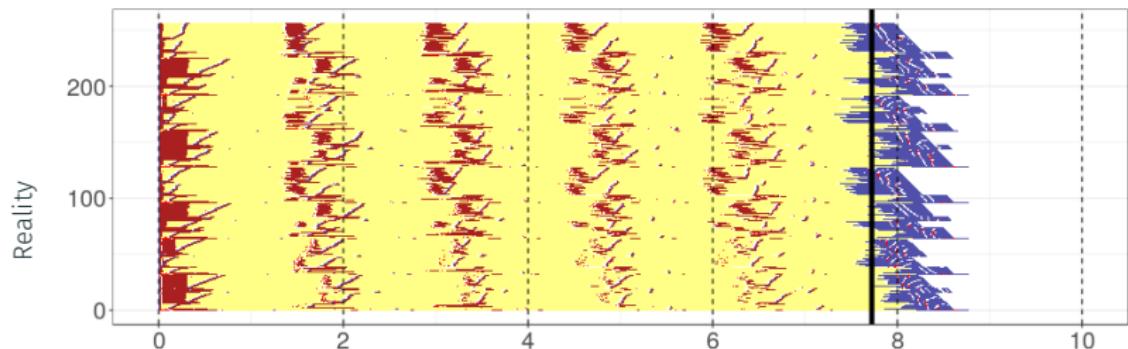
Hand-crafted non-blocking collective operations intertwined with computations



VALIDATING THE PREDICTIONS

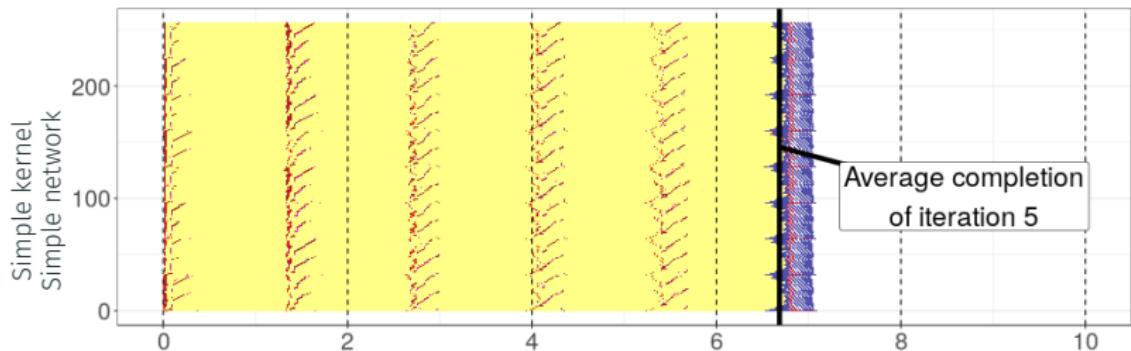
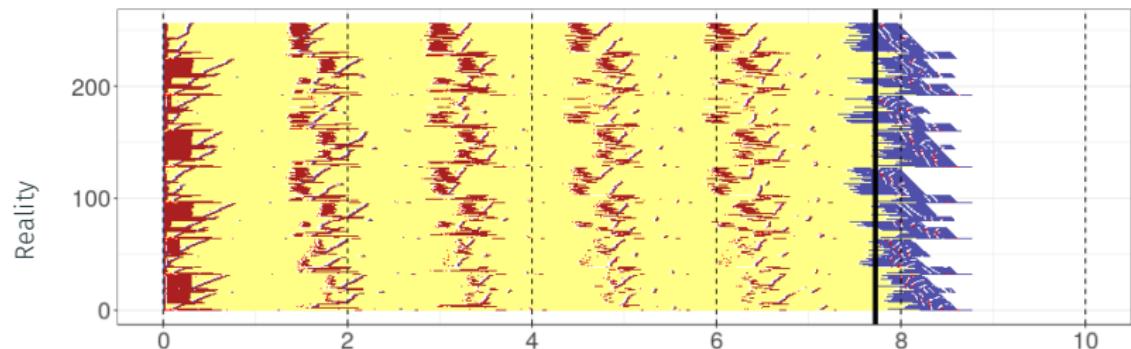
INTERNAL BEHAVIOR OF THE APPLICATION

256 MPI ranks, interrupted after the 5th iteration



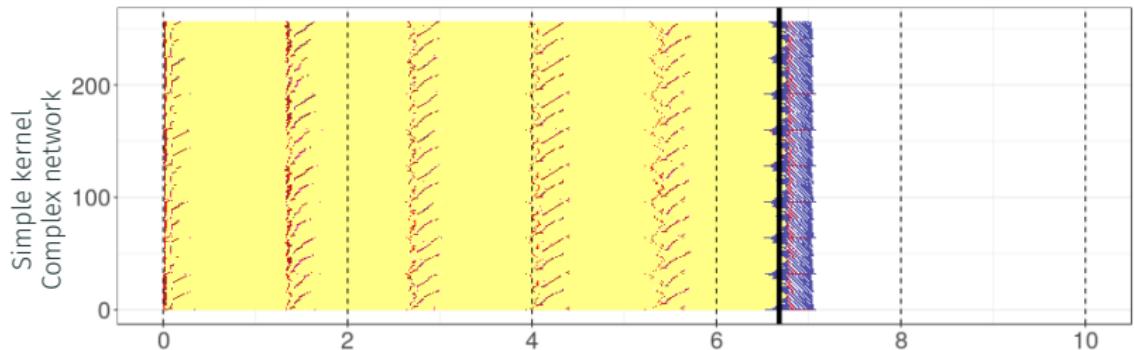
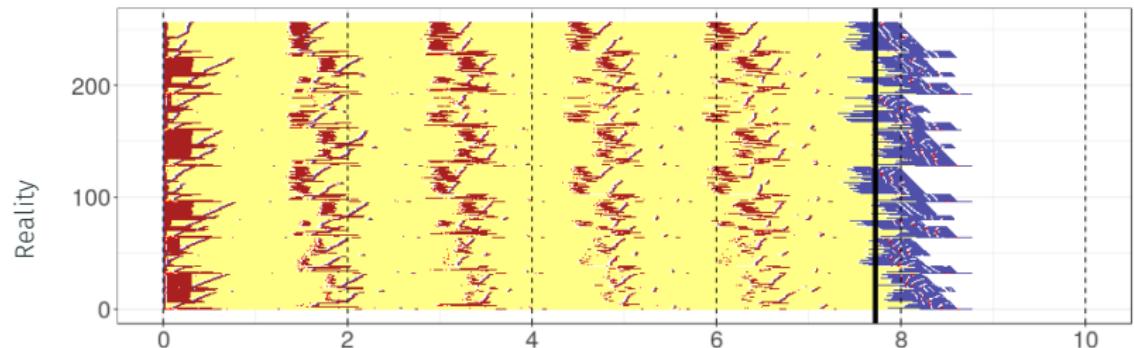
INTERNAL BEHAVIOR OF THE APPLICATION

256 MPI ranks, interrupted after the 5th iteration



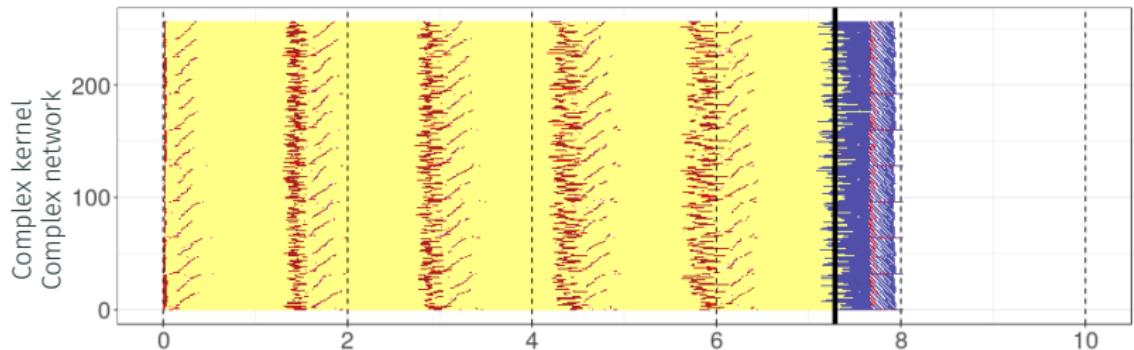
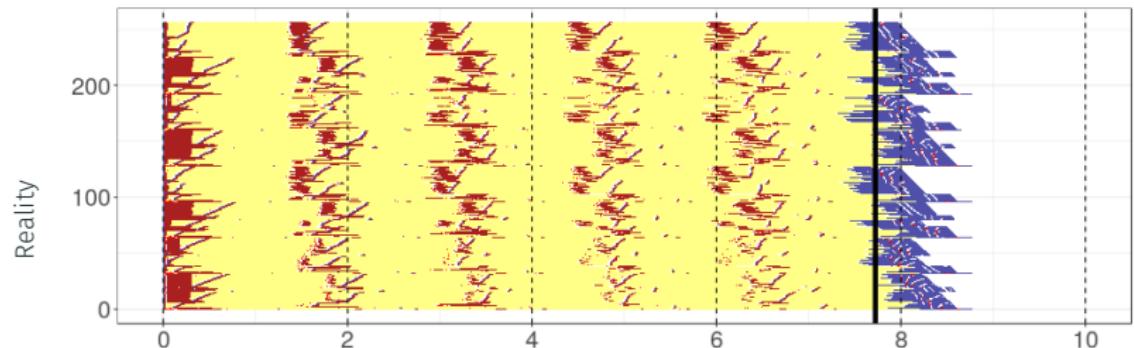
INTERNAL BEHAVIOR OF THE APPLICATION

256 MPI ranks, interrupted after the 5th iteration



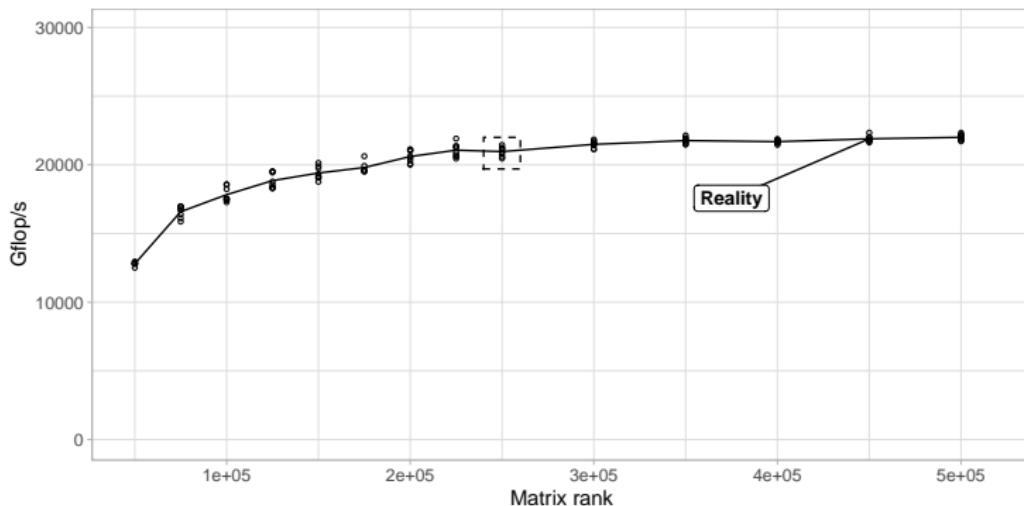
INTERNAL BEHAVIOR OF THE APPLICATION

256 MPI ranks, interrupted after the 5th iteration



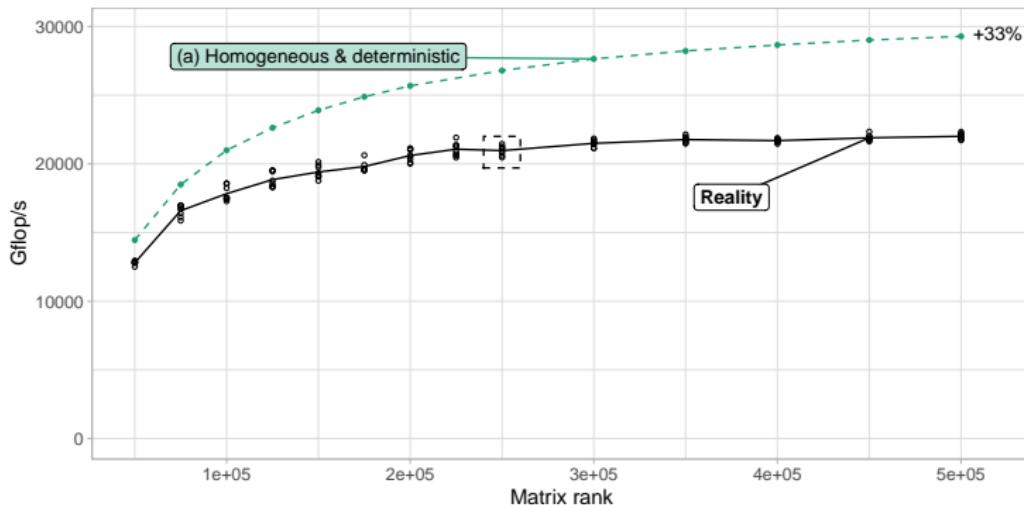
INFLUENCE OF THE PROBLEM SIZE

Now the complete run, with 1024 MPI ranks



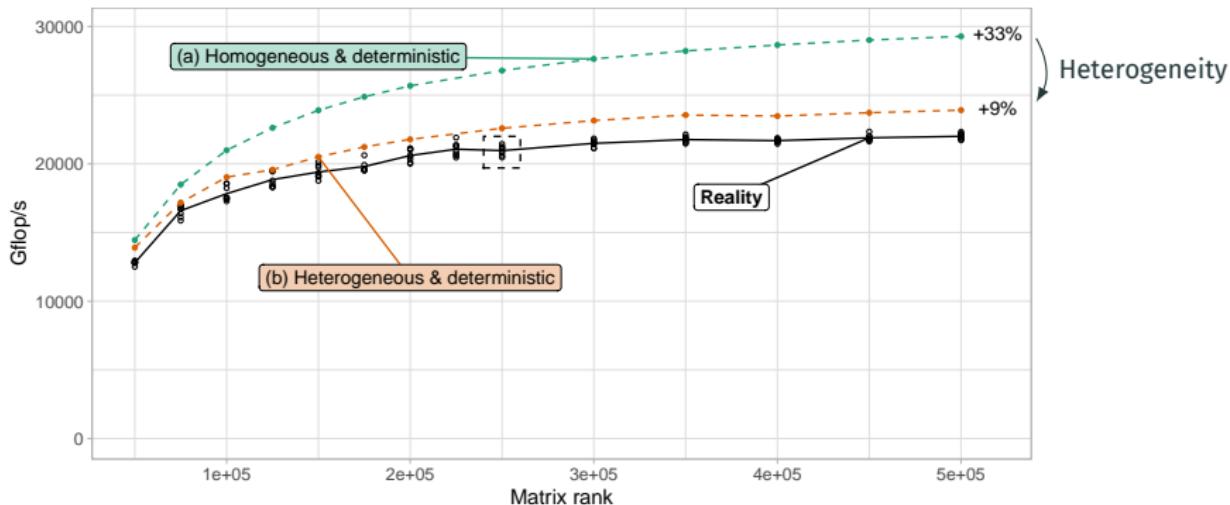
INFLUENCE OF THE PROBLEM SIZE

Now the complete run, with 1024 MPI ranks



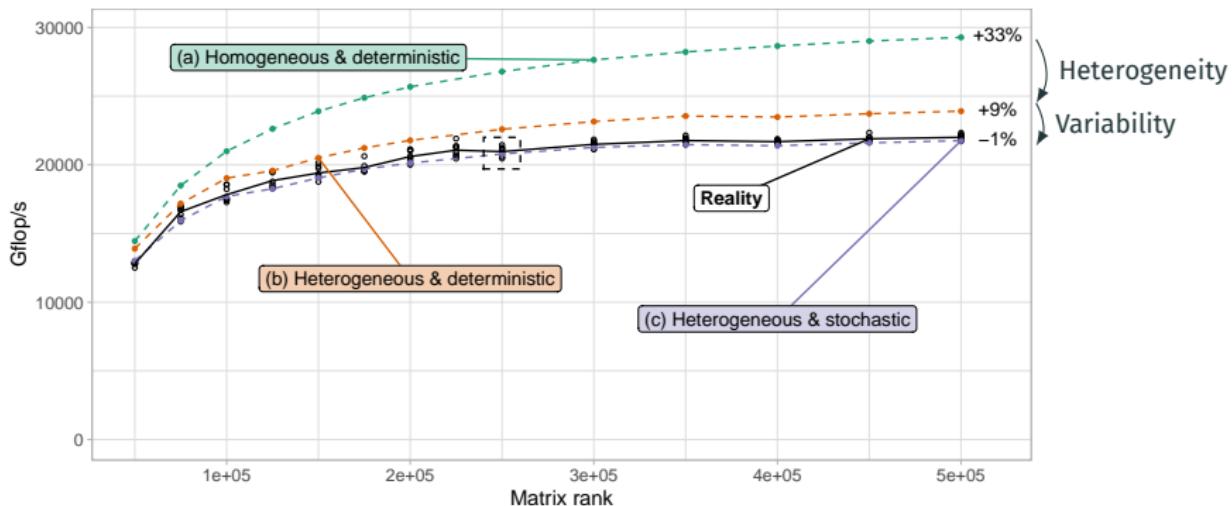
INFLUENCE OF THE PROBLEM SIZE

Now the complete run, with 1024 MPI ranks



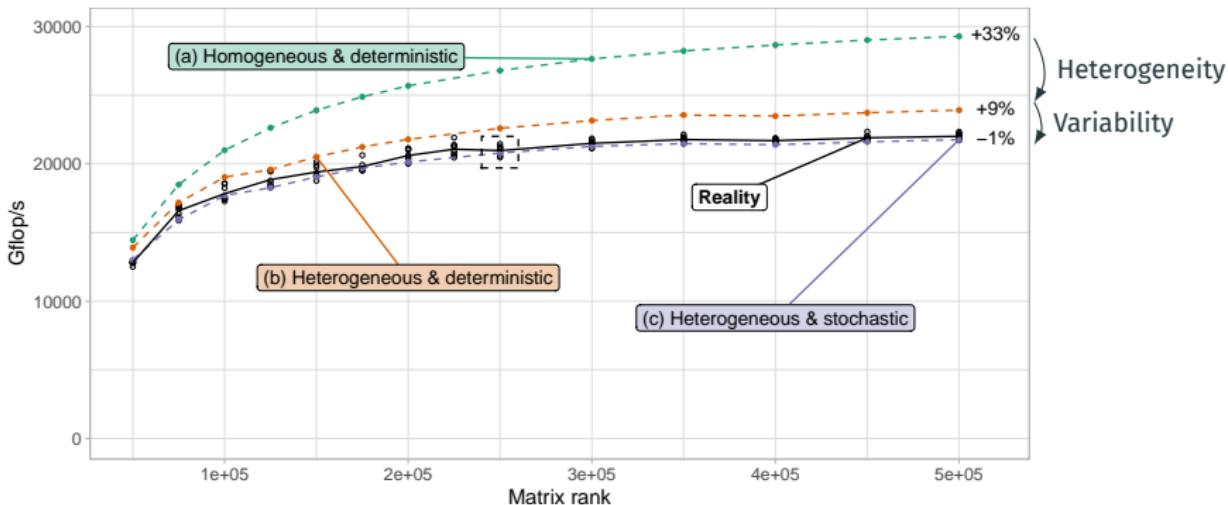
INFLUENCE OF THE PROBLEM SIZE

Now the complete run, with 1024 MPI ranks



INFLUENCE OF THE PROBLEM SIZE

Now the complete run, with 1024 MPI ranks

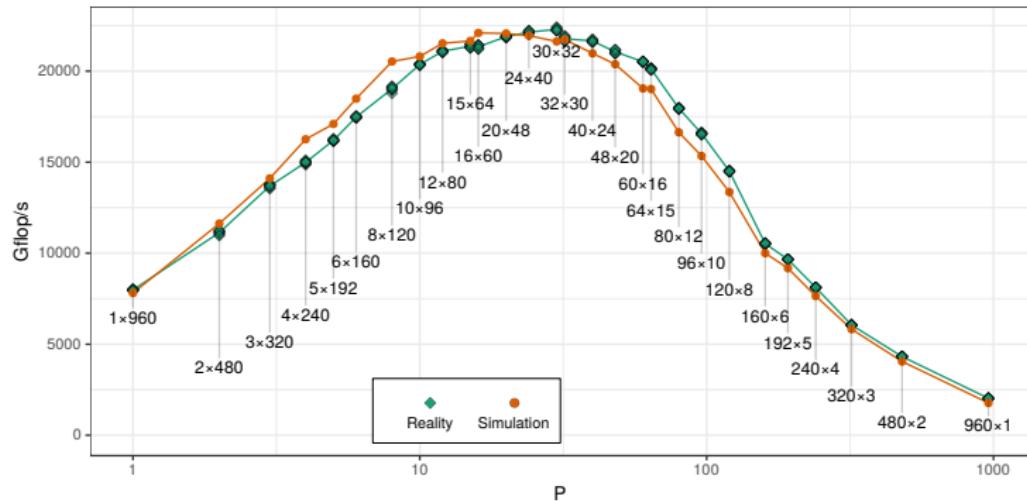


Take-Away Message: accurate prediction

Modeling both spatial and temporal computation variability is essential

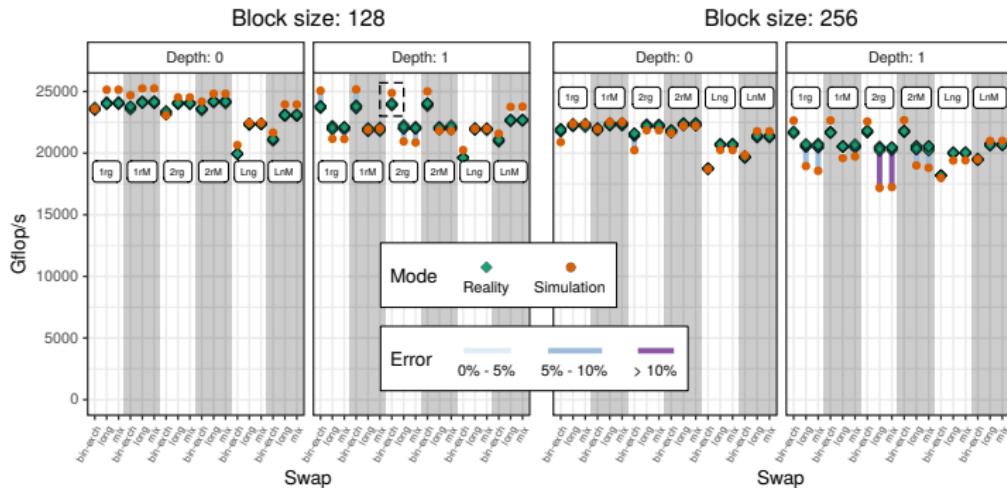
INFLUENCE OF THE GEOMETRY

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

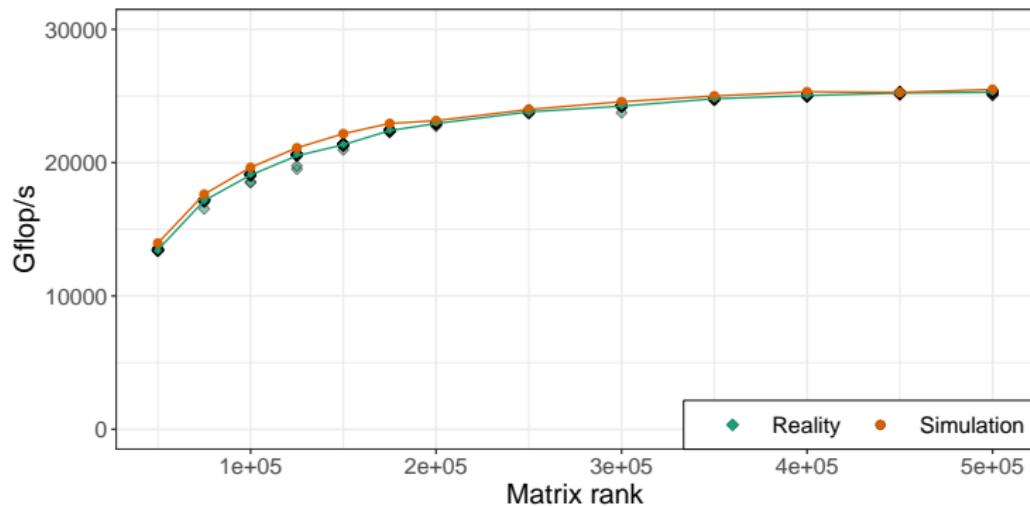


INFLUENCE OF THE OTHER PARAMETERS

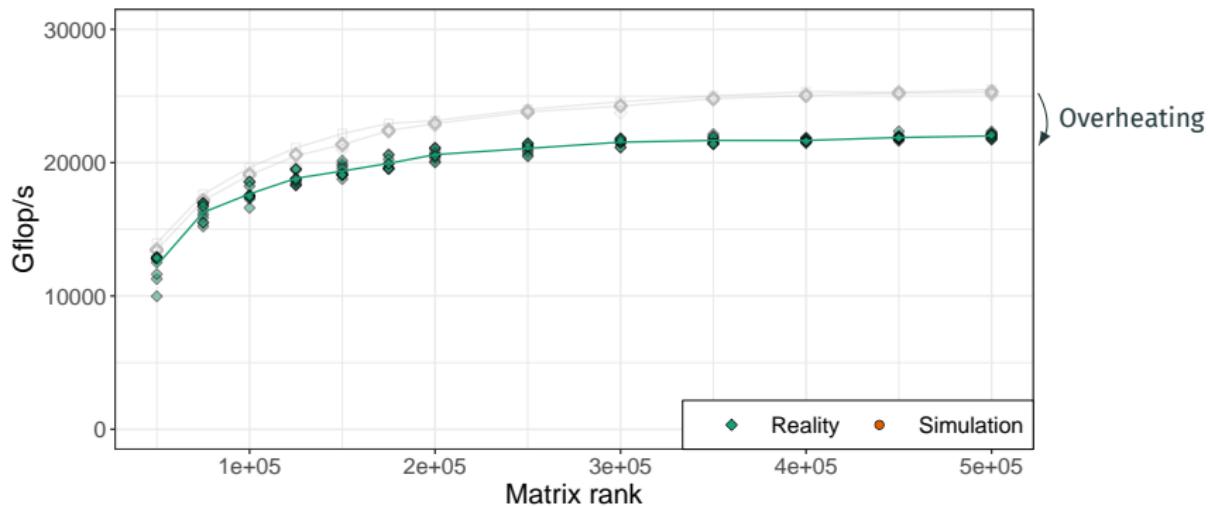
Tested the 72 combinations of the remaining parameters



INFLUENCE OF A PLATFORM CHANGE

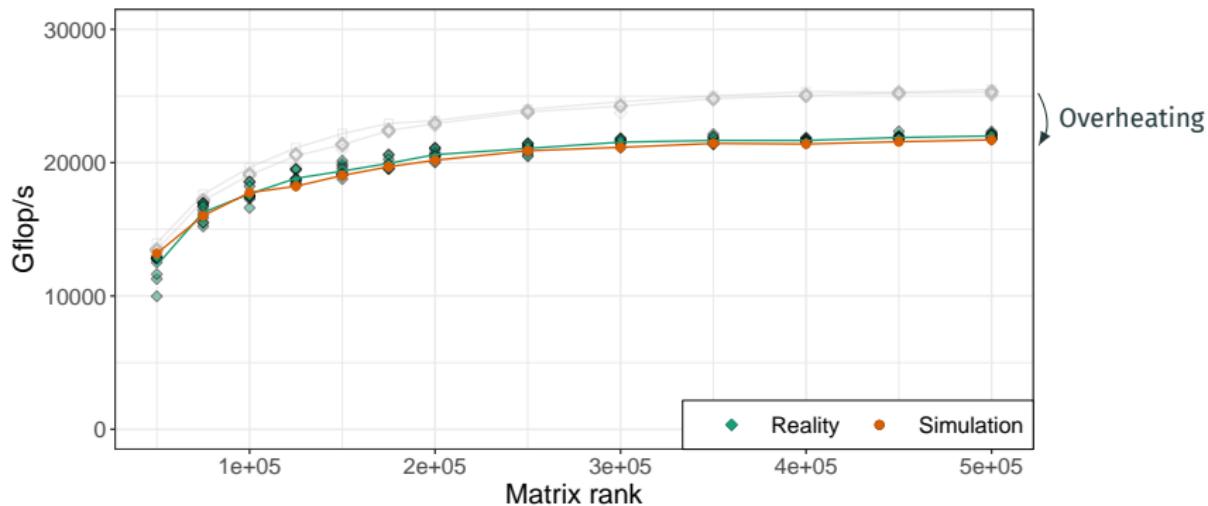


INFLUENCE OF A PLATFORM CHANGE



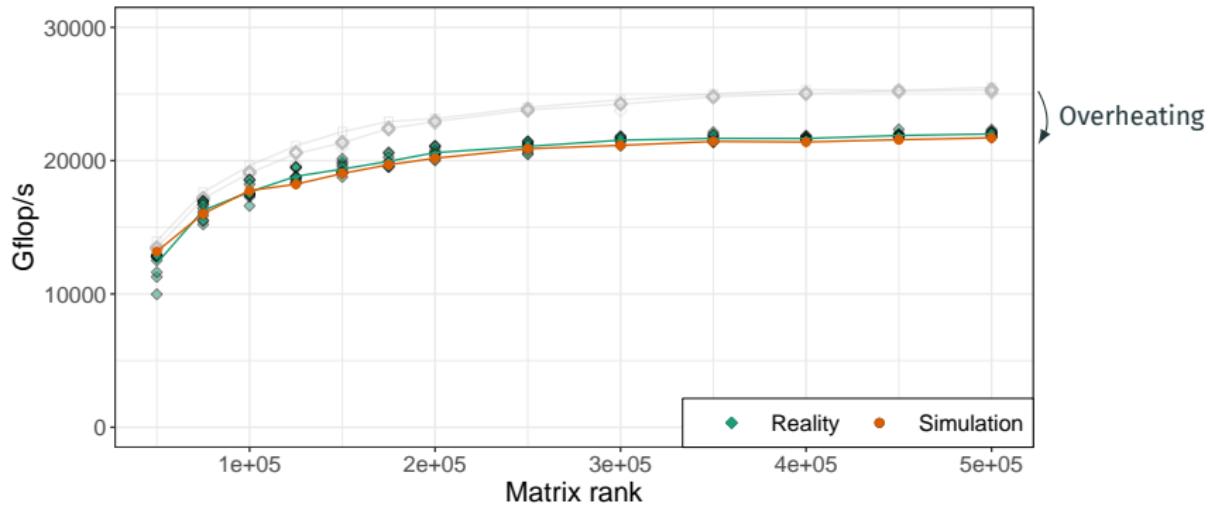
On four nodes, the cooling system malfunctionned for several weeks

INFLUENCE OF A PLATFORM CHANGE



On four nodes, the cooling system malfunctionned for several weeks

INFLUENCE OF A PLATFORM CHANGE

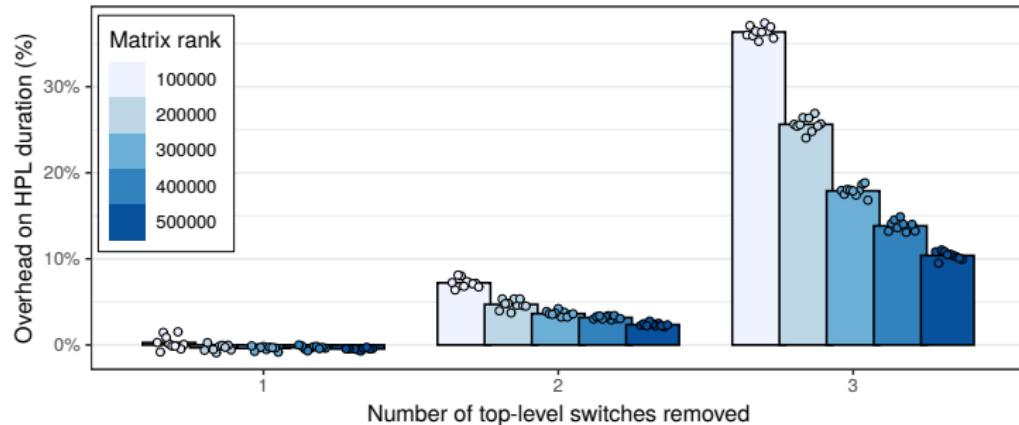


On four nodes, the cooling system malfunctionned for several weeks

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

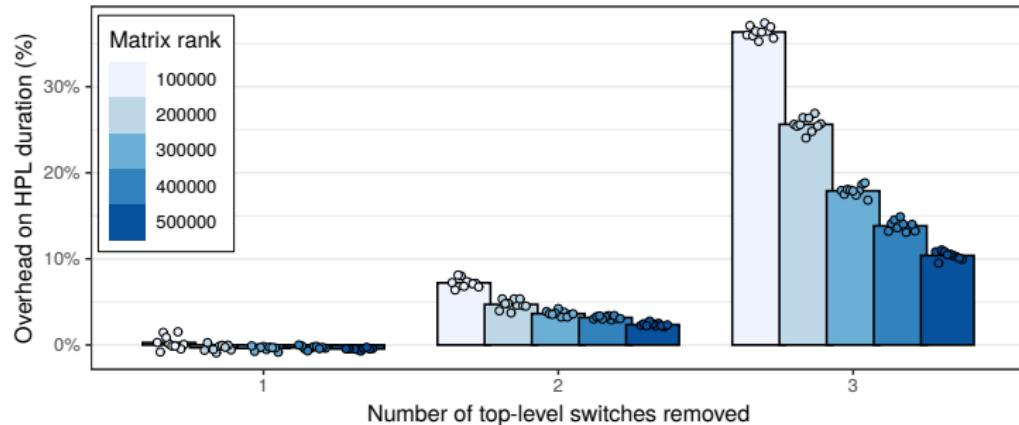
USE CASE: SENSIBILITY ANALYSIS

What if the network topology of my cluster was different?



USE CASE: SENSIBILITY ANALYSIS

What if the network topology of my cluster was different?



Faithful surrogate \Rightarrow Empirical studies of hypothetical platforms
 \Rightarrow Extrapolation of existing platforms
 \Rightarrow Accounting for spatial and temporal variability

Goal: performance prediction ✓

Goal: performance prediction ✓

Main difficulties:

- Realistic experimental conditions
- Platform changes (e.g., the cooling issue)

PARENTHESIS: ON THE DIFFICULTIES OF EXPERIMENTATION

Experimental biases when measuring `dgemm` or MPI durations

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

PARENTHESIS: 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

PARENTHESIS: 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
- Sample itself (for a given sampling method)

PARENTHESIS: 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
- Sample itself (for a given sampling method)
- Interferences between computations and communications

PARENTHESIS: 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
- Sample itself (for a given sampling method)
- Interferences between computations and communications
- Content of the matrices used by `dgemm`

PARENTHESIS: 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
- Sample itself (for a given sampling method)
- Interferences between computations and communications
- Content of the matrices used by `dgemm`

Bias may be desirable in some situations

PERFORMANCE TESTS

REGULAR MEASURES

On a near-daily basis, run the `dgemm` calibration code on
454 nodes (792 CPU) from 12 clusters



REGULAR MEASURES

On a near-daily basis, run the `dgemm` calibration code on
454 nodes (792 CPU) from 12 clusters



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
454 nodes (792 CPU) from 12 clusters



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

REGULAR MEASURES

On a near-daily basis, run the `dgemm` calibration code on
454 nodes (792 CPU) from 12 clusters



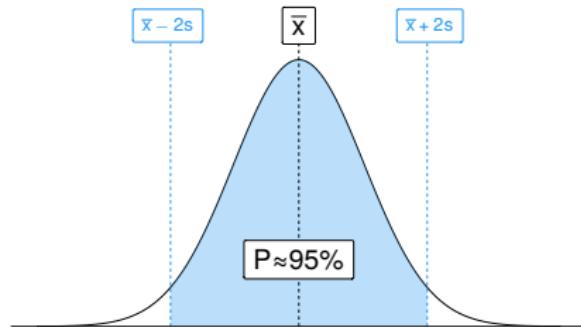
For each CPU, collect:

- 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

Each parameter is **normally distributed** (thanks to CLT)

FLUCTUATION INTERVAL

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

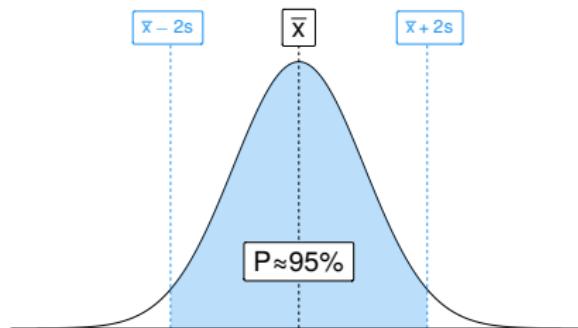


Take the sample mean \bar{x} and standard deviation s of the old observations

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

FLUCTUATION INTERVAL

Given a sequence of old observations x_1, \dots, 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 standard deviation s of the old observations

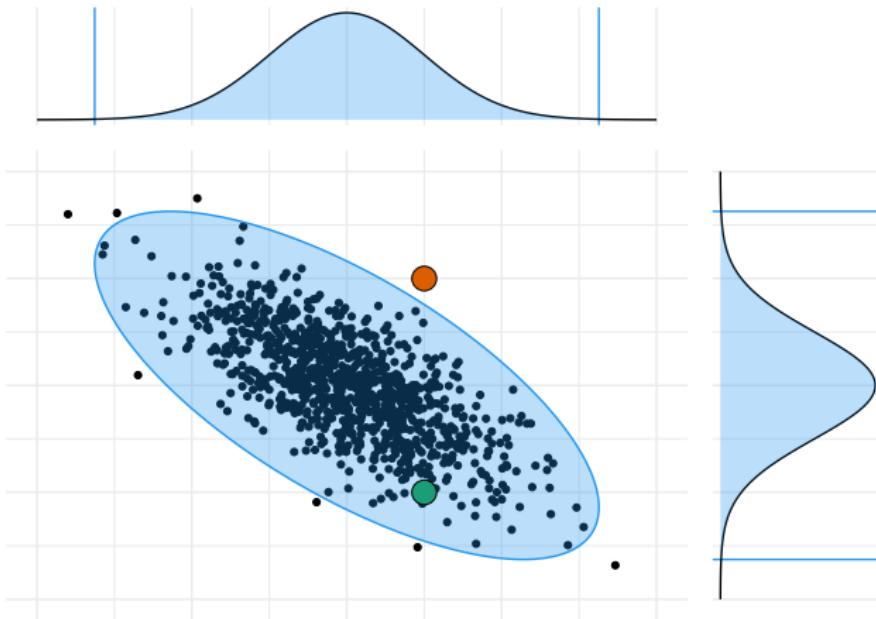
$$\mathbb{P}(x_{n+1} \in [\bar{x} - 2s; \bar{x} + 2s]) \approx 95\%$$

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, using their [covariance matrix](#)

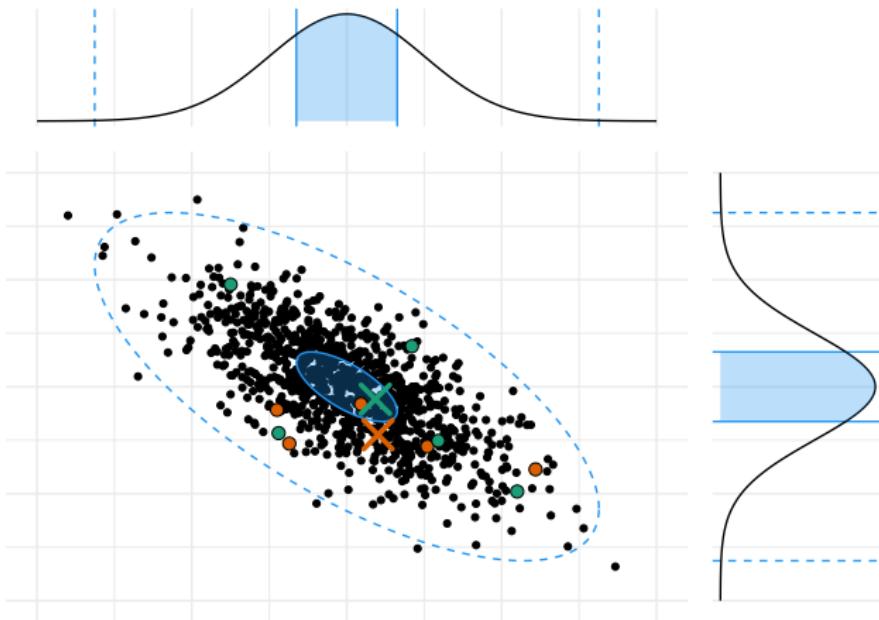
Example in dimension 2, with $\mathbb{P}(x_{n+1} \in \text{interval}) \approx 99.5\%$



FLUCTUATION INTERVAL FOR SEVERAL MEASURES

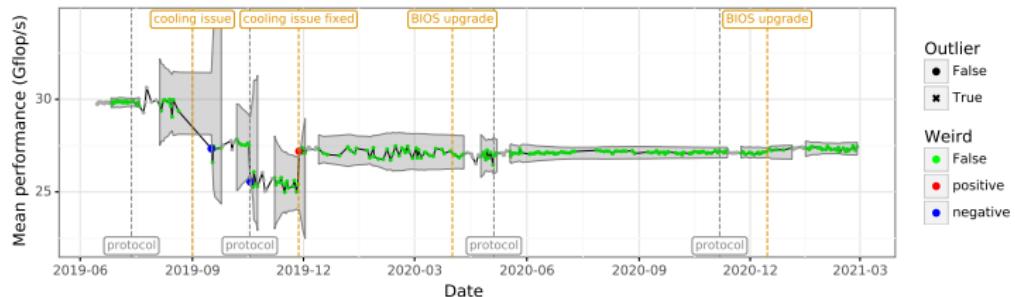
With several measures, using their **average** and shrinking the interval

Example with 5 measures (averages represented by crosses)



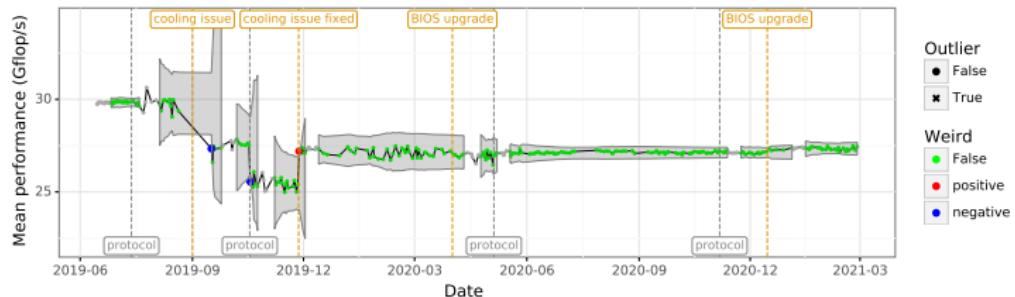
RESULT: PERFORMANCE FLUCTUATION

Performance fluctuation of the node dahu-14

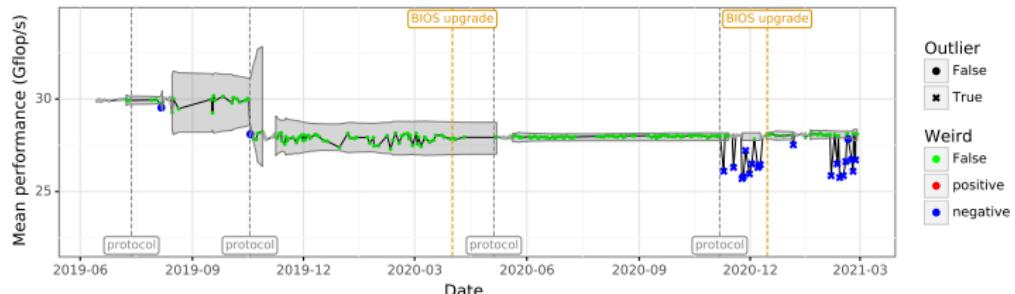


RESULT: PERFORMANCE FLUCTUATION

Performance fluctuation of the node dahu-14

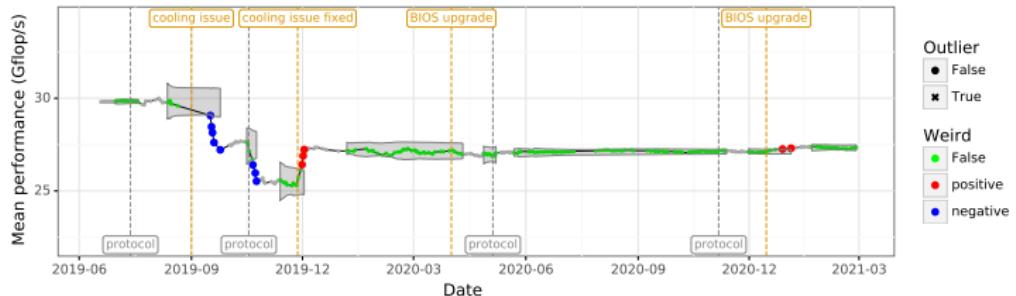


Performance fluctuation of the node dahu-32

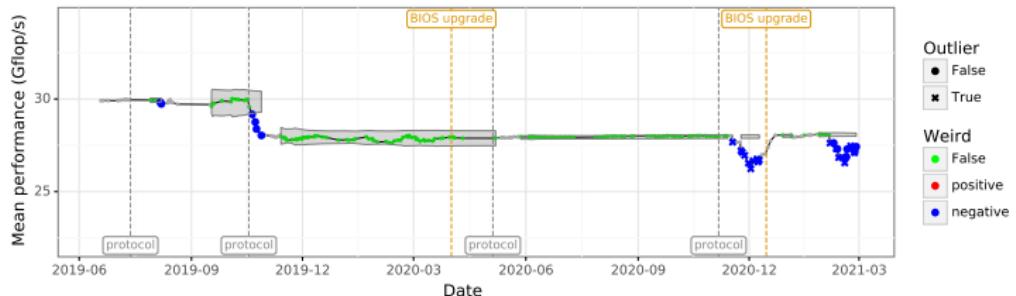


RESULT: PERFORMANCE FLUCTUATION

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

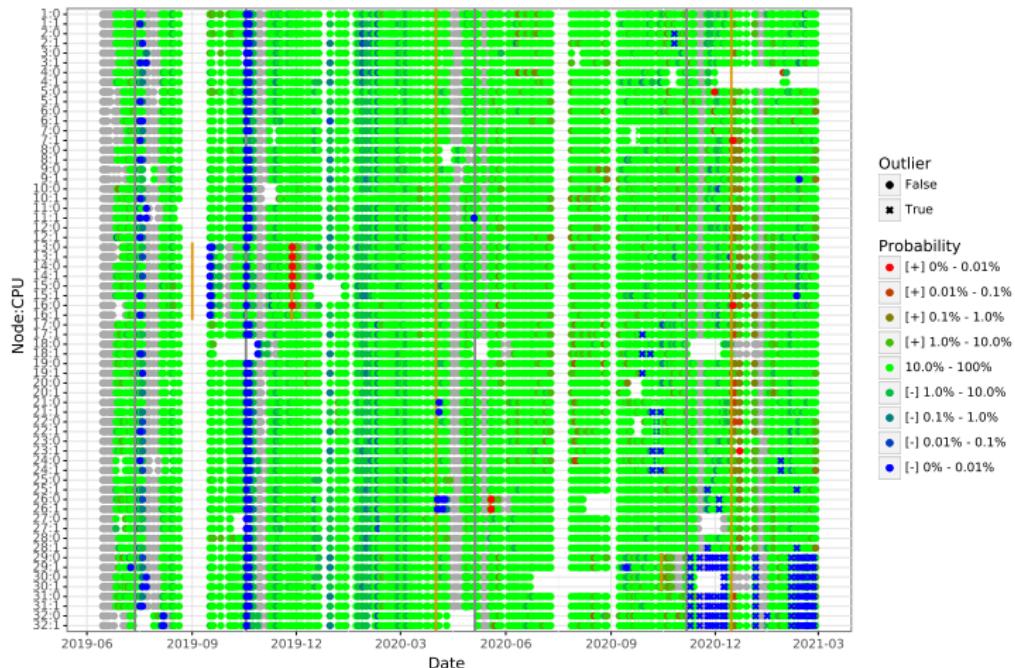


Performance fluctuation of the node dahu-32 (5-day window)



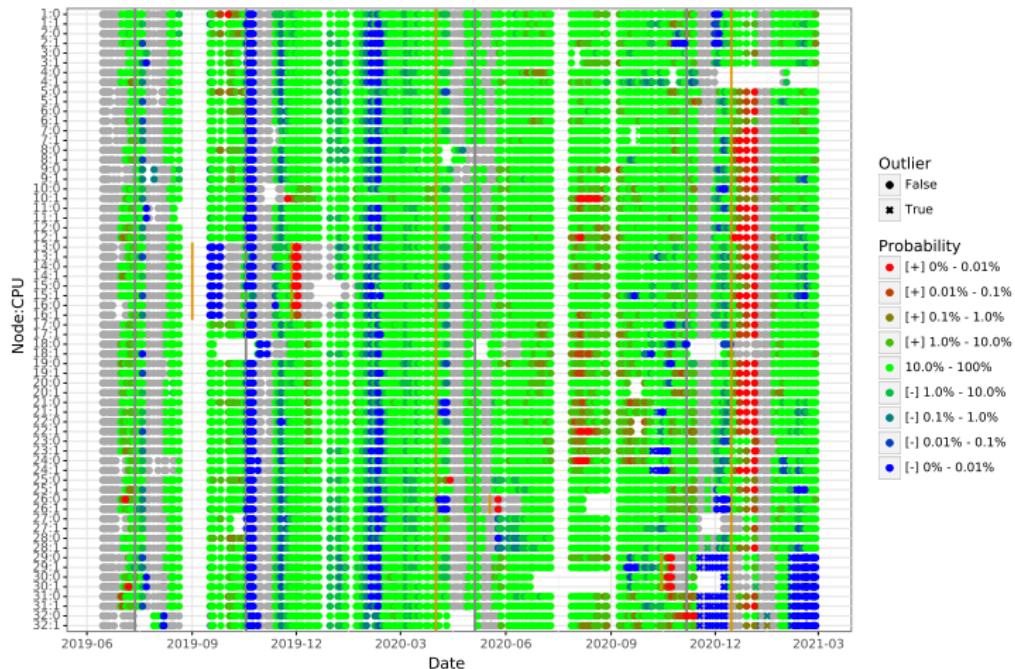
RESULT: PERFORMANCE OVERVIEW

Overview of the performance on cluster dahu



RESULT: PERFORMANCE OVERVIEW

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



PERFORMANCE TESTS: WRAPING UP

Multi-variable test also implemented

PERFORMANCE TESTS: WRAPING UP

Multi-variable test also implemented

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



PERFORMANCE TESTS: WRAPING UP

Multi-variable test also implemented

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

Cluster	Performance	Performance _{PS48}	Frequency	Power _{CPU}	Power _{DRAM}	Temperature	Model
chezbeni							
chidlet							
dahu							
ecotype							
grossu							
gras							
gringst							
parasito							
panitanice							
pyxis							
troll							
yeti							

Detected events

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

PERFORMANCE TESTS: WRAPING UP

Multi-variable test also implemented

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

Cluster	Performance	Performance _{PS4}	Frequency	Power _{CPU}	Power _{DRAM}	Temperature	Model
chezeti							
chidet							
dahu							
ecotype							
grisou							
gris							
gringst							
parasito							
pantharce							
pyxis							
troll							
yeti							

Detected events

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

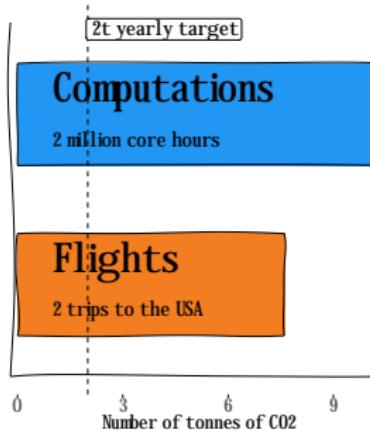
All went unnoticed by both Grid'5000 staff and users, despite significant effects

⇒ Great help potential

CONCLUDING THOUGHTS

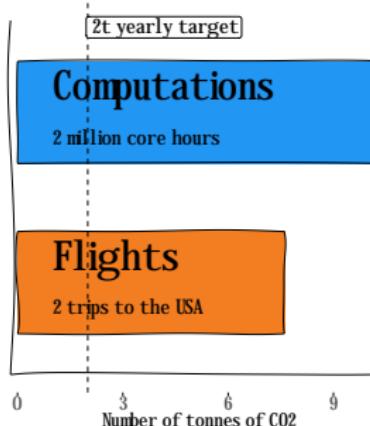
THERE IS NO PLANET B

About 18t of CO₂eq were emitted for this thesis



THERE IS NO PLANET B

About 18t 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, use the standard image instead)
- Test less frequently (e.g. only once a week)

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, use the standard image instead)
- Test less frequently (e.g. only once a week)

Who should be responsible of tests?

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