

# HIGH PERFORMANCE COMPUTING: TOWARDS BETTER PERFORMANCE PREDICTIONS AND EXPERIMENTS

---

Tom Cornebize

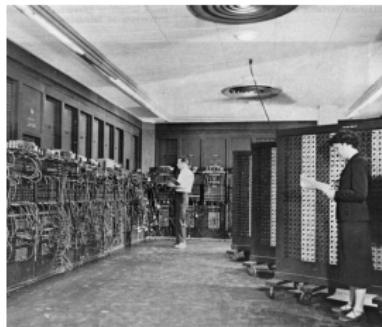
2 June 2021, PhD defense



# No SCIENCE WITHOUT COMPUTING



Arithmomètre (1851)



ENIAC (1945)

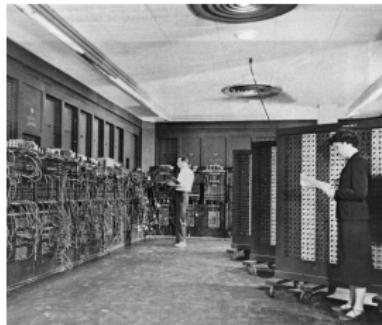


Fugaku (2021)

# No SCIENCE WITHOUT COMPUTING



Arithmomètre (1851)



ENIAC (1945)

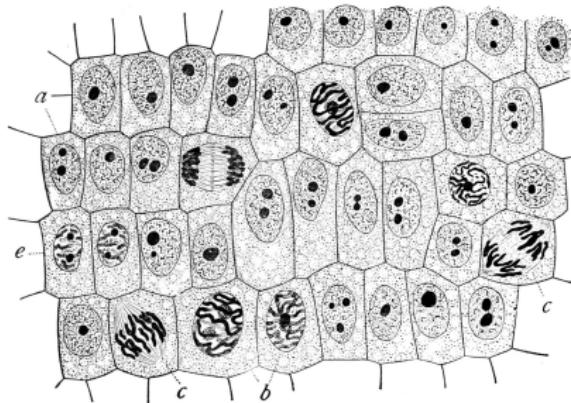


Fugaku (2021)

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



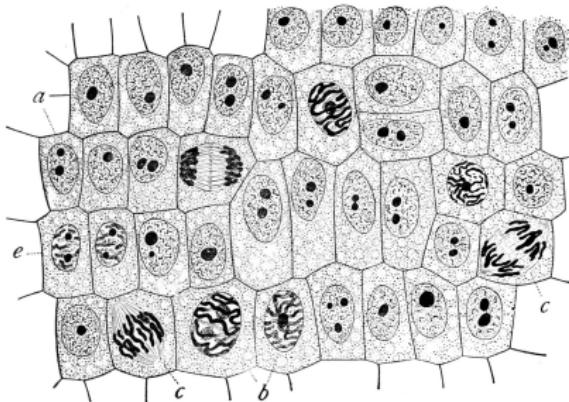
Similar to natural sciences

Complexity ⇒ Variability and Opacity

⇒ No perfect model

⇒ Need for [experiments](#)

# EXPERIMENTAL STUDY OF COMPUTER PERFORMANCE



Similar to natural sciences

Complexity ⇒ Variability and Opacity

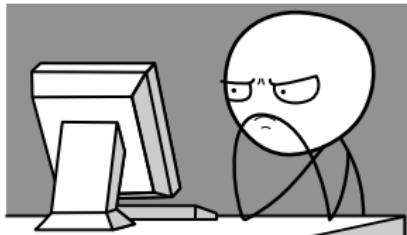
⇒ No perfect model

⇒ Need for experiments

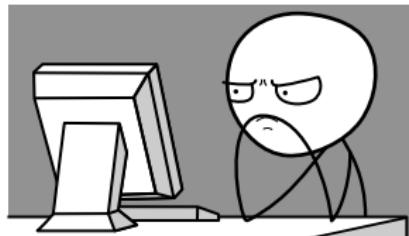
Experiments can be carried in [reality](#) or in [simulation](#)

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

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

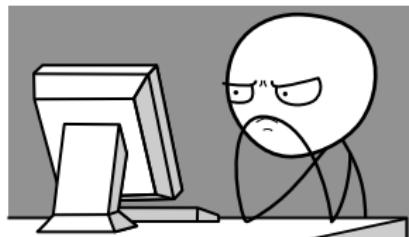


## Typical Performance Evaluation Questions (Given my application and a supercomputer)



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

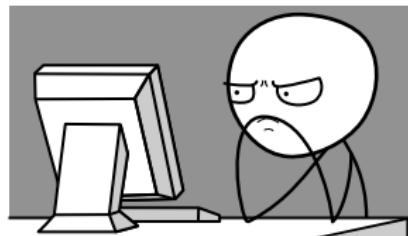
## Typical Performance Evaluation Questions (Given my application and a supercomputer)



- **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?



- Computations run 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



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

## Emulation: how?

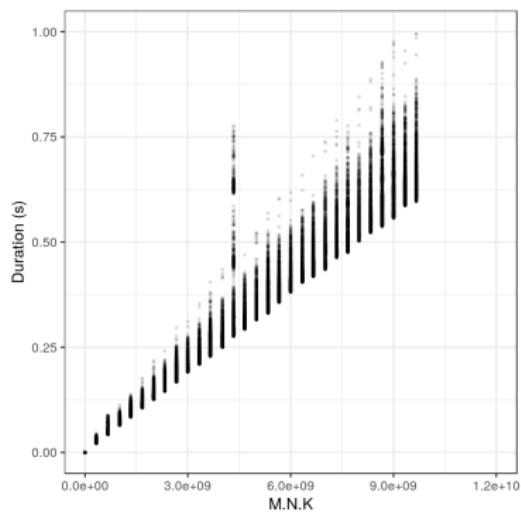


- Computations run for real on a laptop
- Communications are faked, good fluid network models
- **Performance model** for the target platform

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

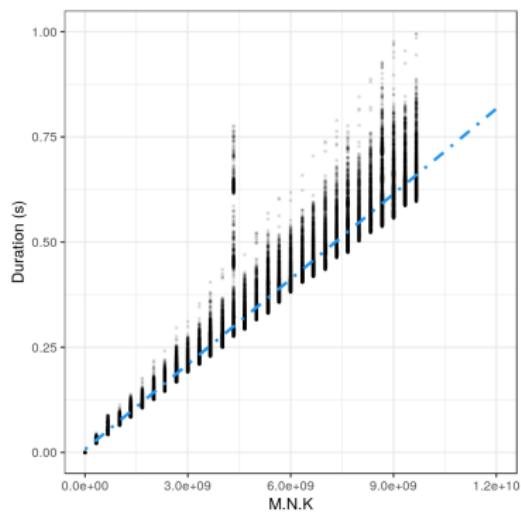
# MODELING COMPUTATIONS

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



# MODELING COMPUTATIONS

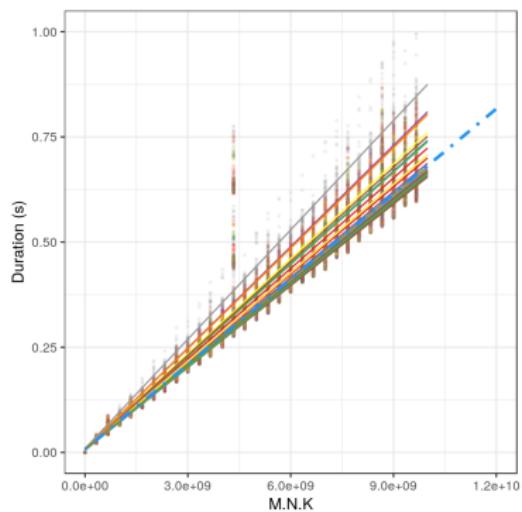
$$\text{dgemm } (M, N, K) = \alpha \cdot M \cdot N \cdot K$$



# MODELING COMPUTATIONS

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

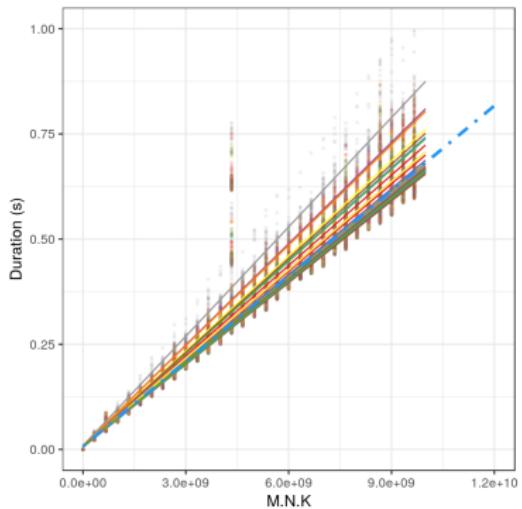
Different color  $\Rightarrow$  different host



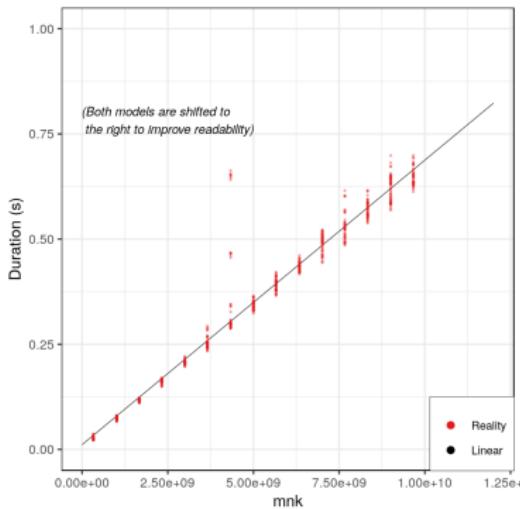
# MODELING COMPUTATIONS

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

Different color  $\Rightarrow$  different host



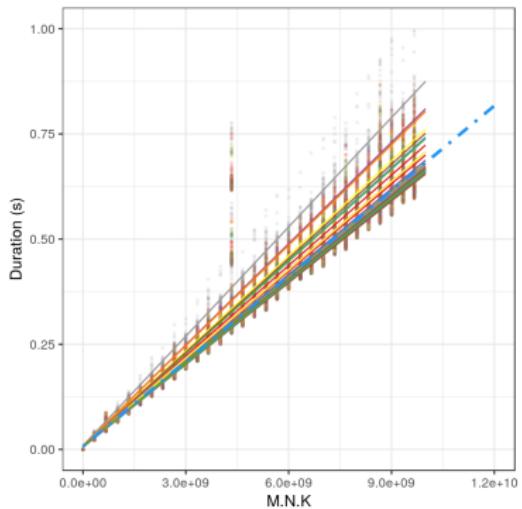
For a particular host



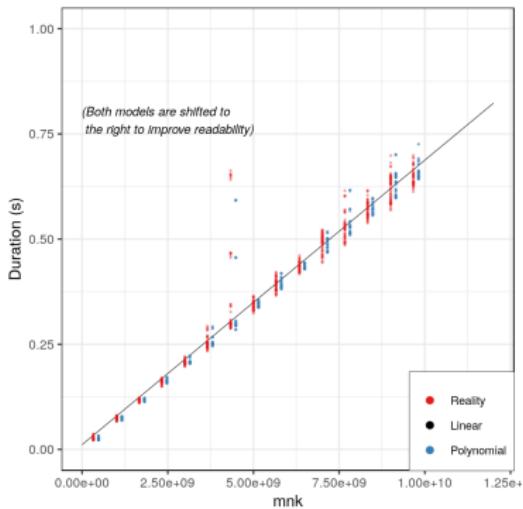
# MODELING COMPUTATIONS

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

Different color  $\Rightarrow$  different host



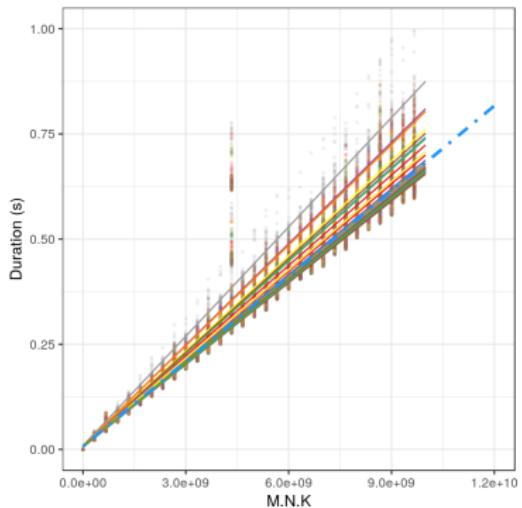
For a particular host



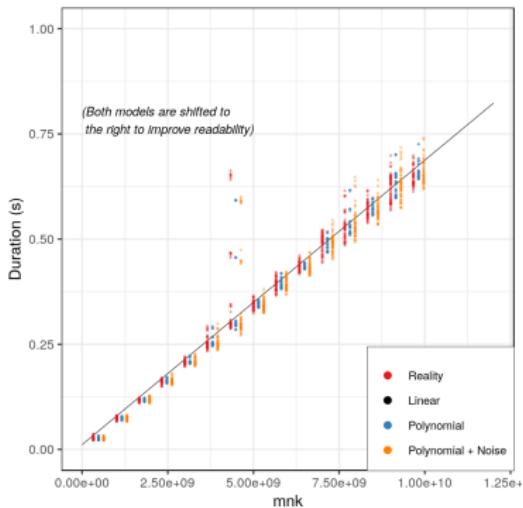
# MODELING COMPUTATIONS

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

Different color  $\Rightarrow$  different host



For a particular host



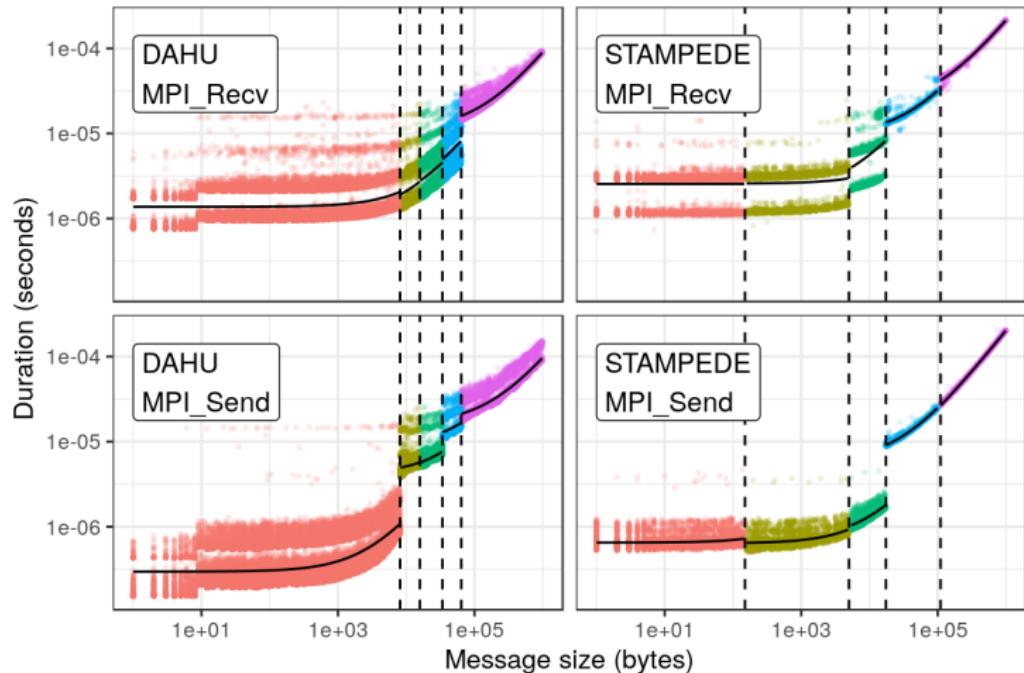
## MODELING COMMUNICATIONS

---

Hand-crafted non-blocking collective operations intertwined with computations

# MODELING COMMUNICATIONS

Hand-crafted non-blocking collective operations intertwined with computations



## 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`

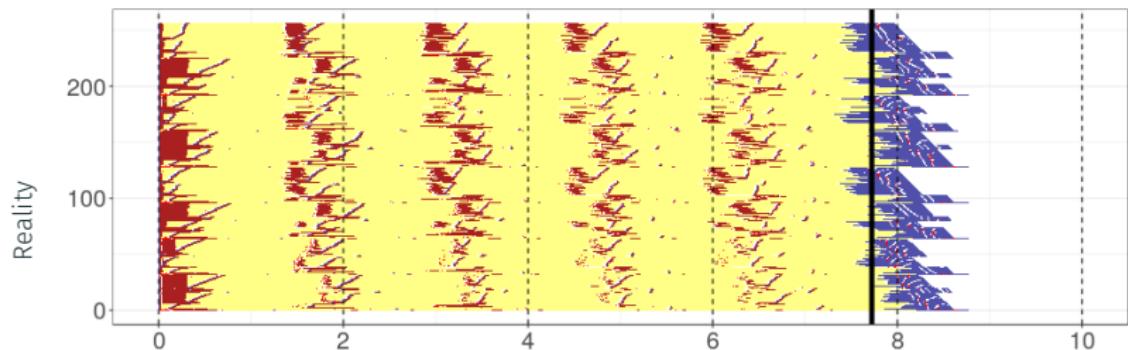
Bias may be desirable in some situations

## VALIDATING THE PREDICTIONS

---

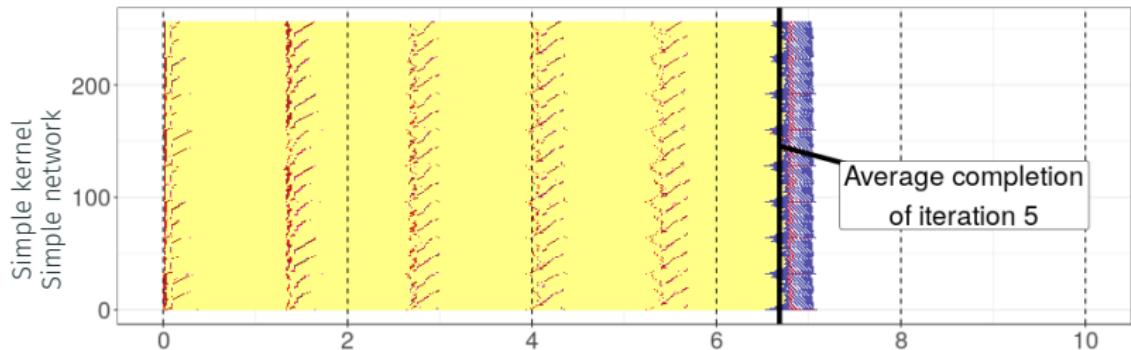
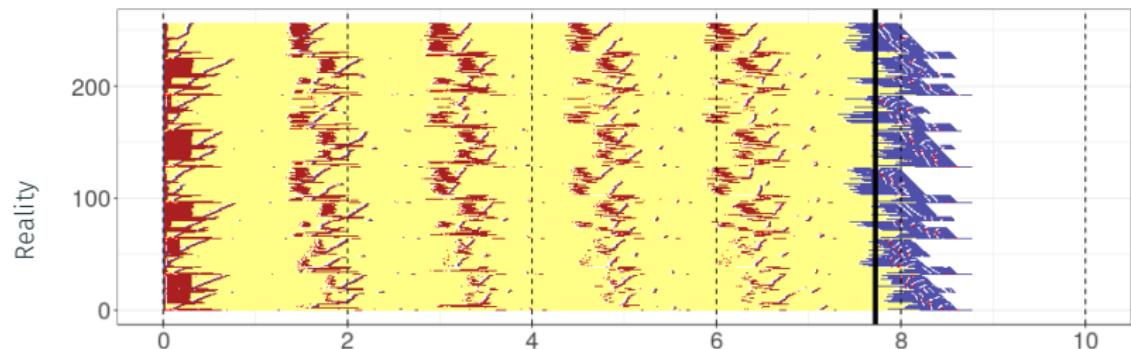
# INTERNAL BEHAVIOR OF THE APPLICATION

256 MPI ranks, interrupted after the 5<sup>th</sup> iteration



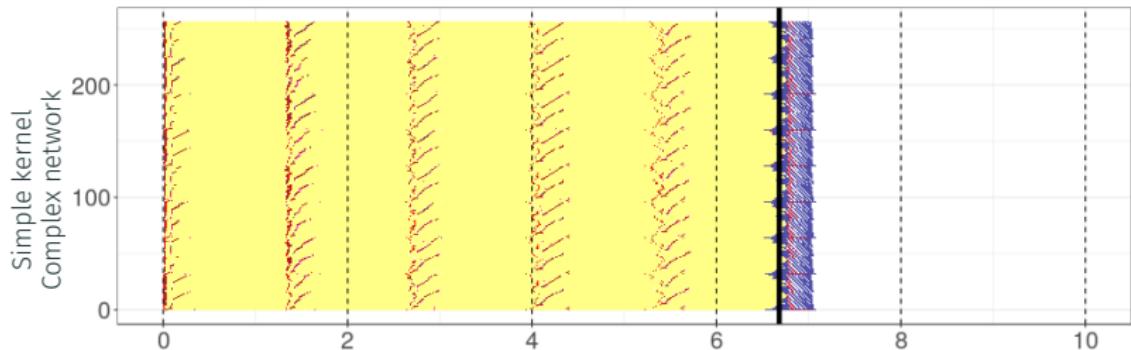
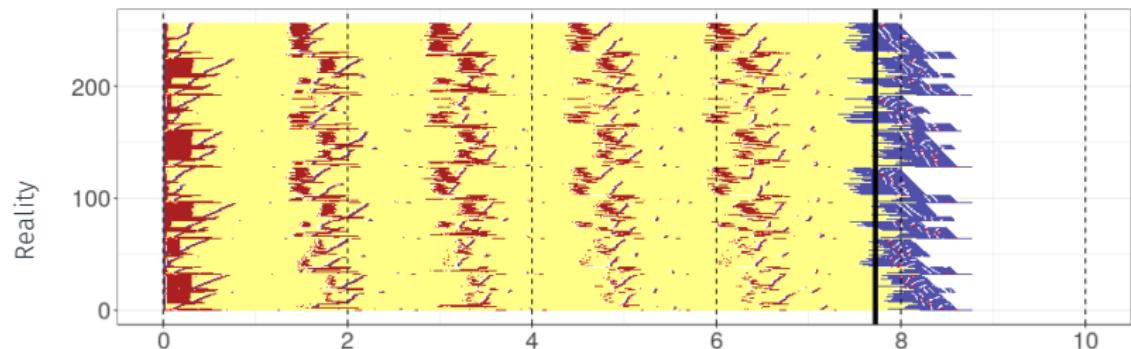
# INTERNAL BEHAVIOR OF THE APPLICATION

256 MPI ranks, interrupted after the 5<sup>th</sup> iteration



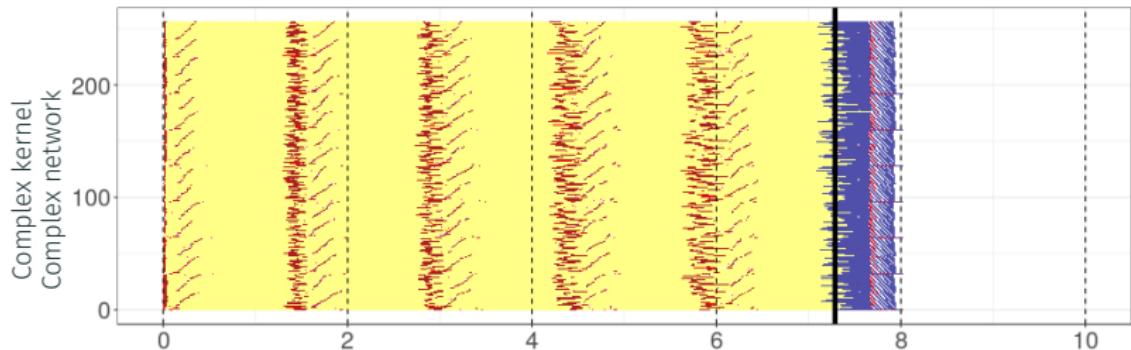
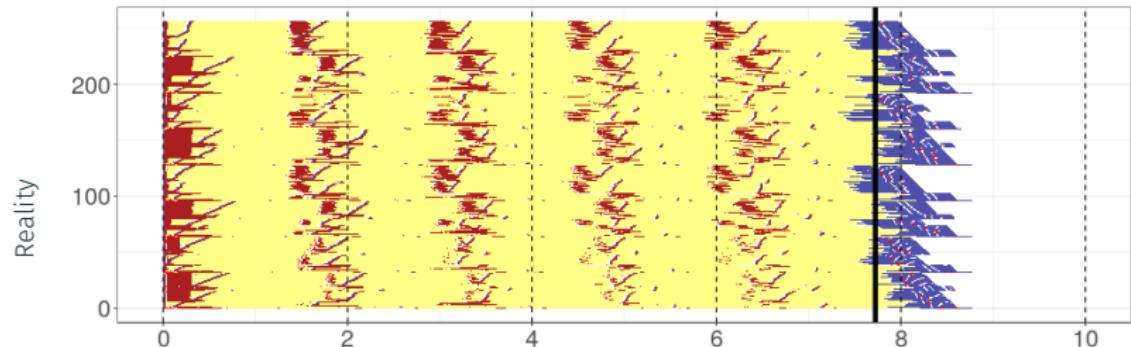
# INTERNAL BEHAVIOR OF THE APPLICATION

256 MPI ranks, interrupted after the 5<sup>th</sup> iteration



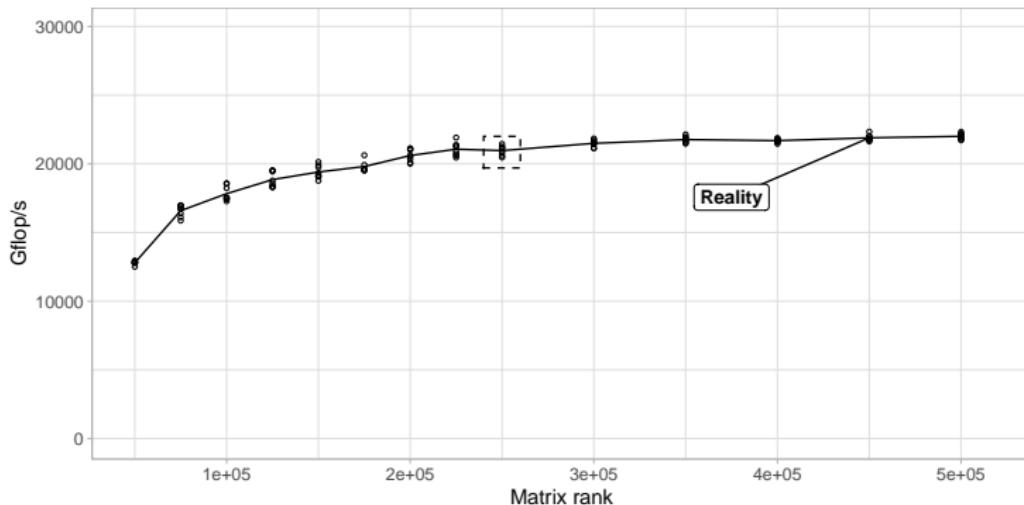
# INTERNAL BEHAVIOR OF THE APPLICATION

256 MPI ranks, interrupted after the 5<sup>th</sup> iteration



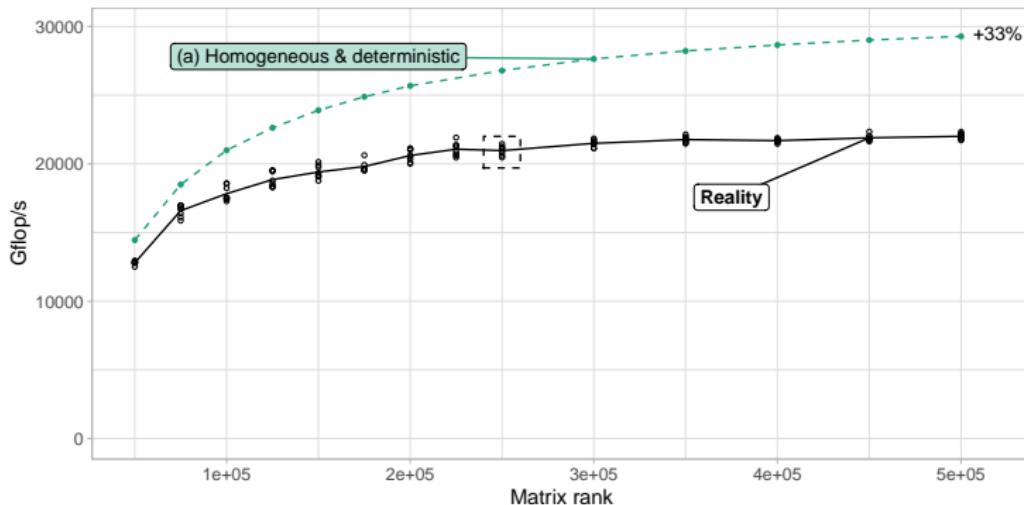
# INFLUENCE OF THE PROBLEM SIZE

Now the complete run, with 1024 MPI ranks



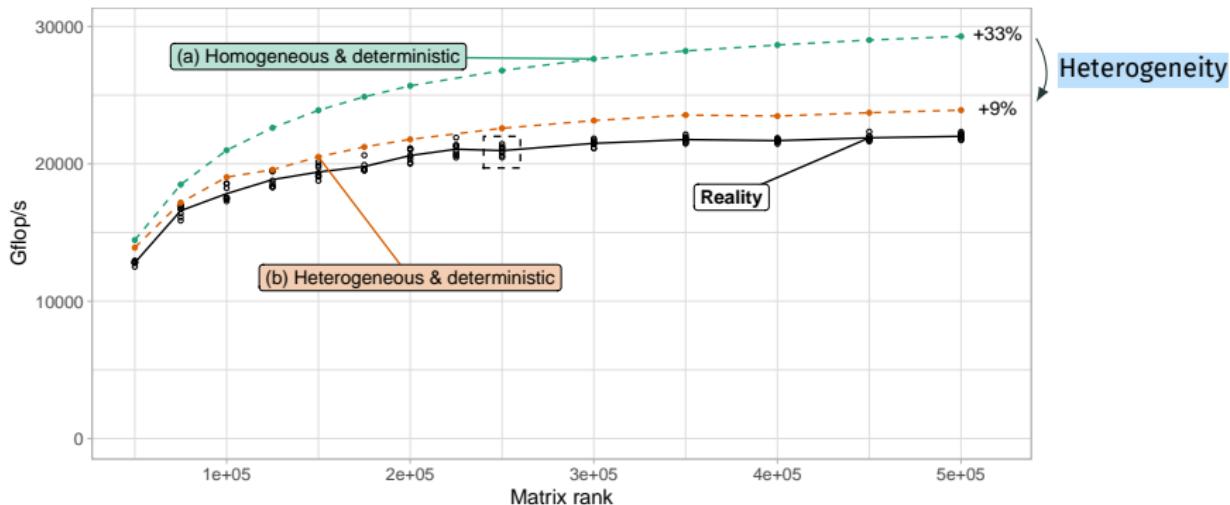
# INFLUENCE OF THE PROBLEM SIZE

Now the complete run, with 1024 MPI ranks



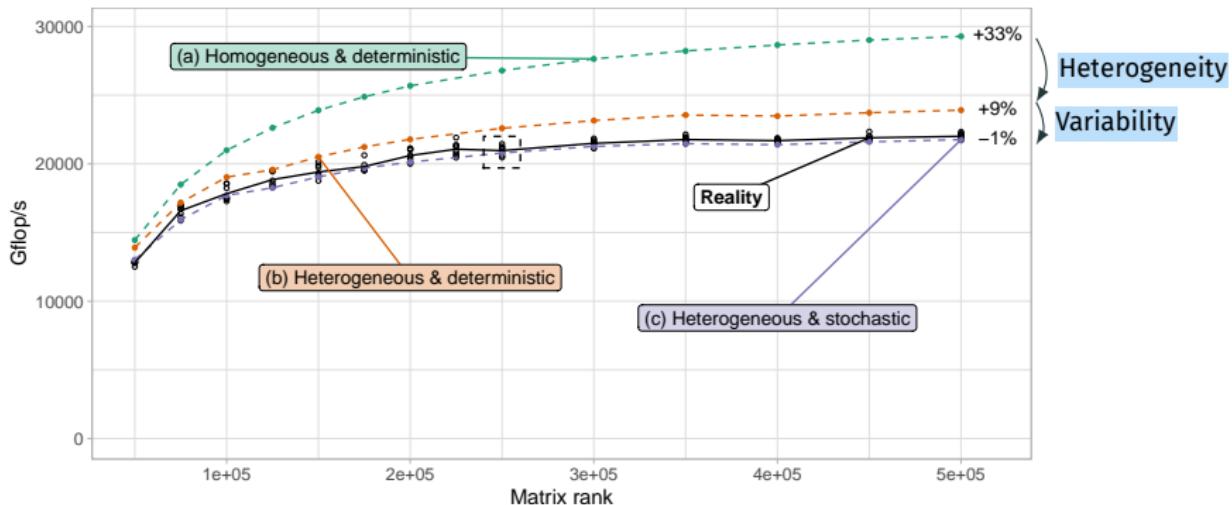
# INFLUENCE OF THE PROBLEM SIZE

Now the complete run, with 1024 MPI ranks



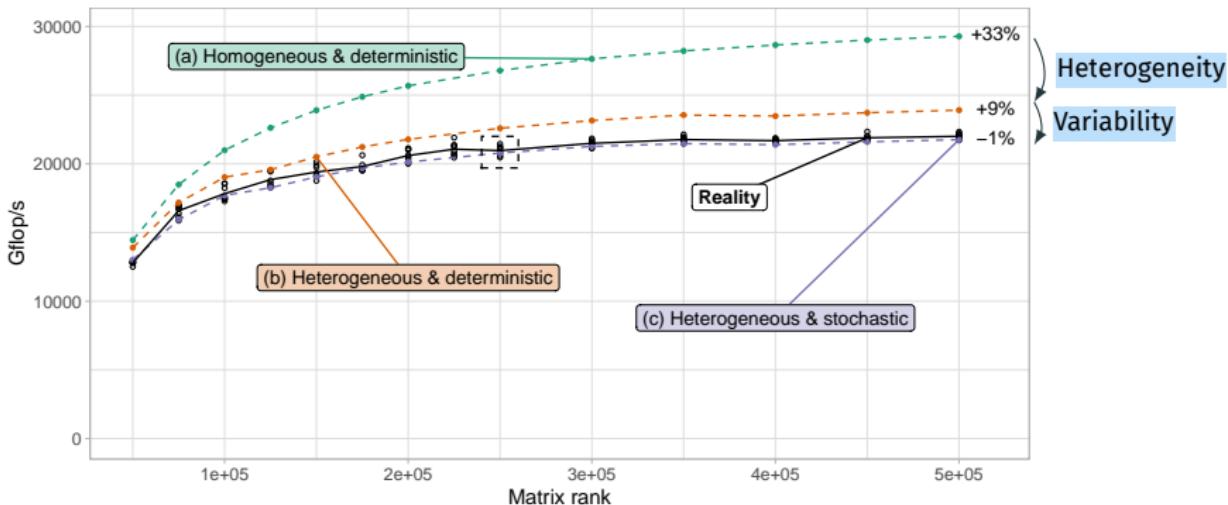
# INFLUENCE OF THE PROBLEM SIZE

Now the complete run, with 1024 MPI ranks



# INFLUENCE OF THE PROBLEM SIZE

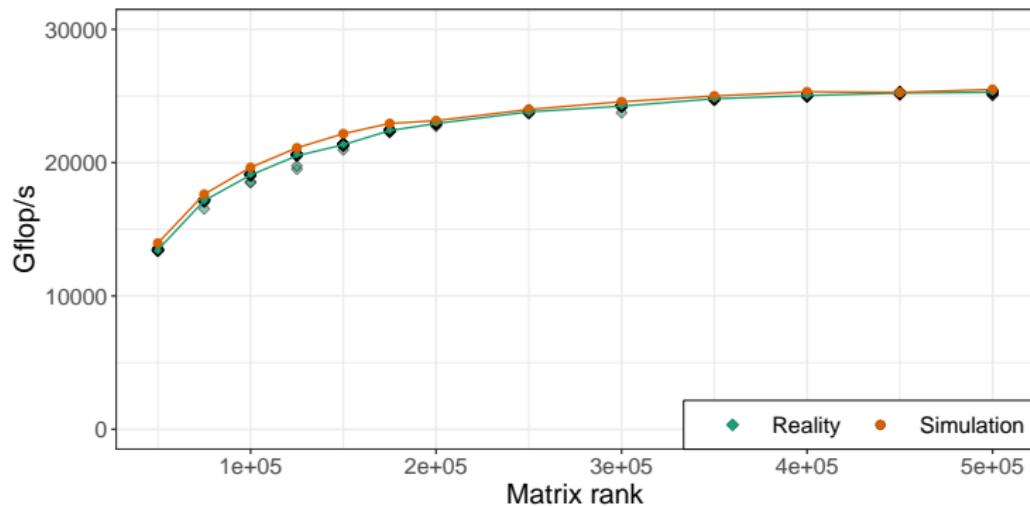
Now the complete run, with 1024 MPI ranks



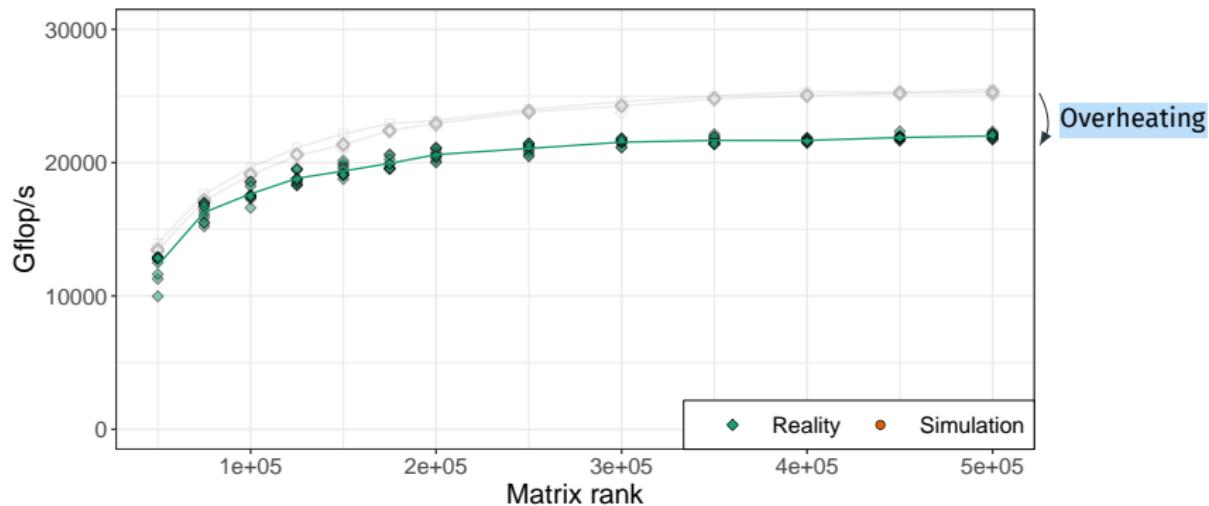
Take-Away Message: accurate prediction

Modeling both **spatial** and **temporal** computation variability is essential

# INFLUENCE OF A PLATFORM CHANGE

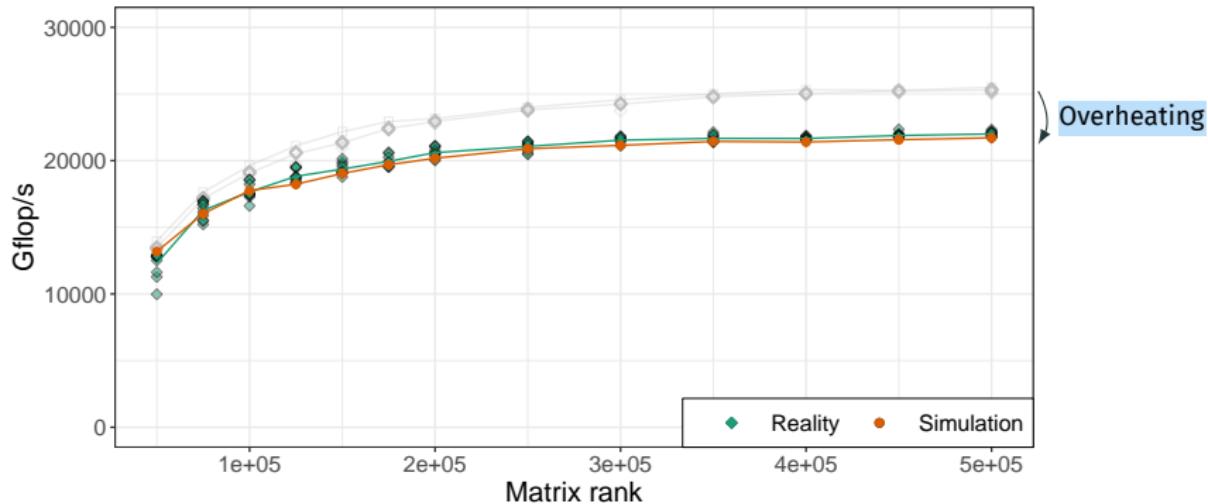


# INFLUENCE OF A PLATFORM CHANGE



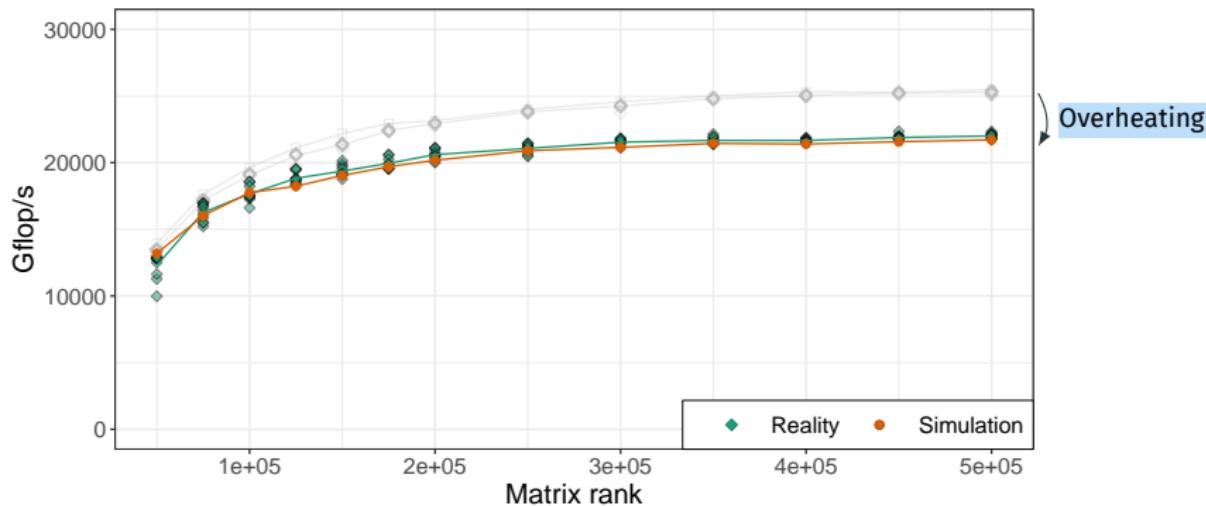
On four nodes, the cooling system malfunctionned for several weeks

# 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

## 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



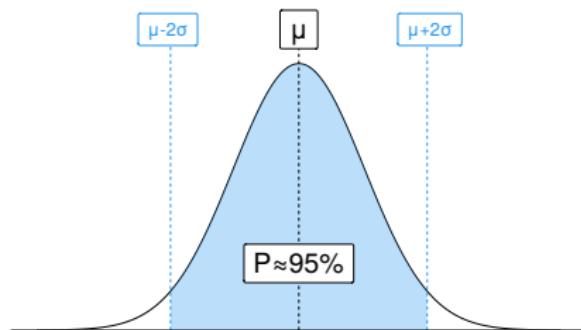
For each CPU, collect:

- 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  $x_{n+1}$ ?



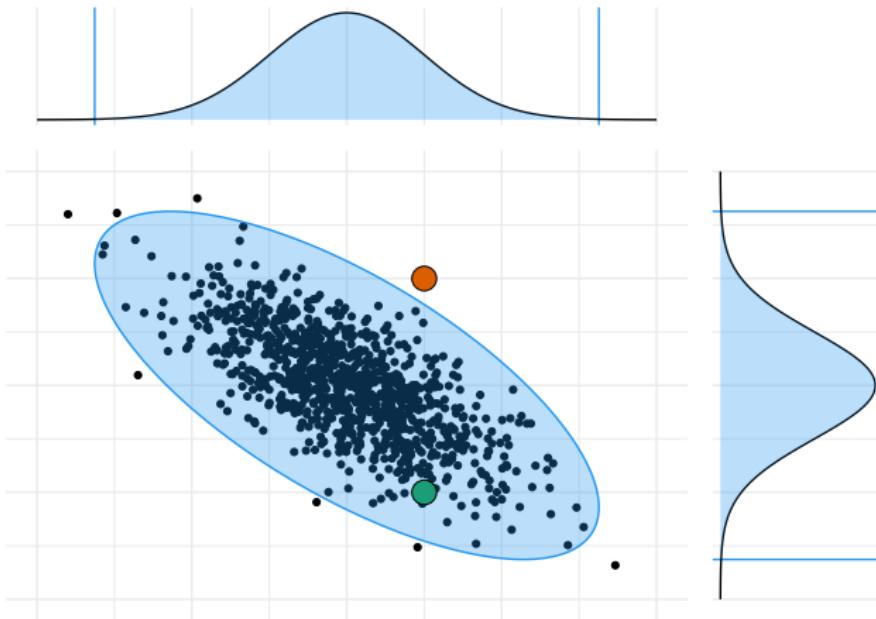
Take the sample mean  $\mu$  and standard deviation  $\sigma$  of the old observations

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

## FLUCTUATION INTERVAL FOR SEVERAL VARIABLES

With several variables, using their [covariance matrix](#)

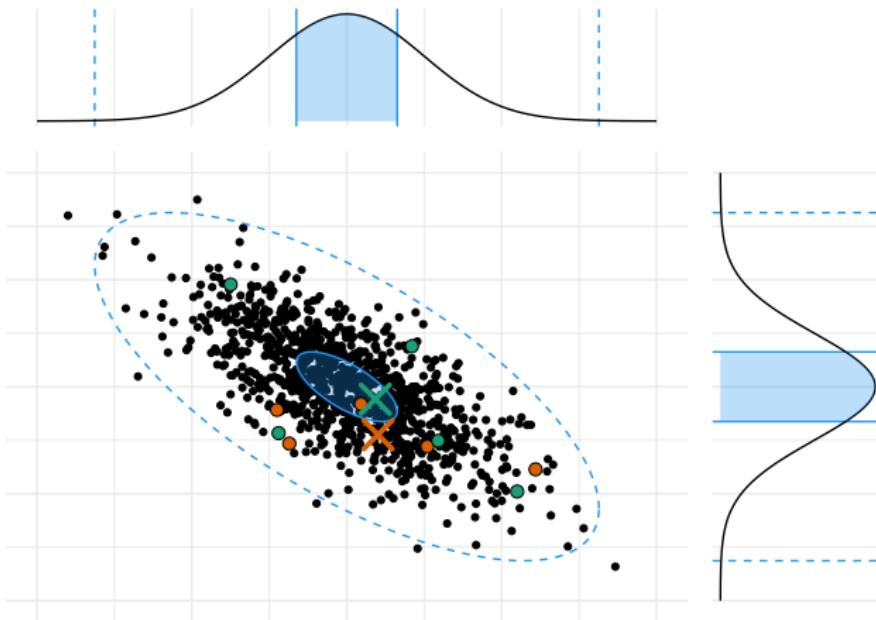
Example in dimension 2, with  $\mathbb{P}(x_{n+1} \in \text{interval}) \approx 99.5\%$



## FLUCTUATION INTERVAL FOR SEVERAL MEASURES

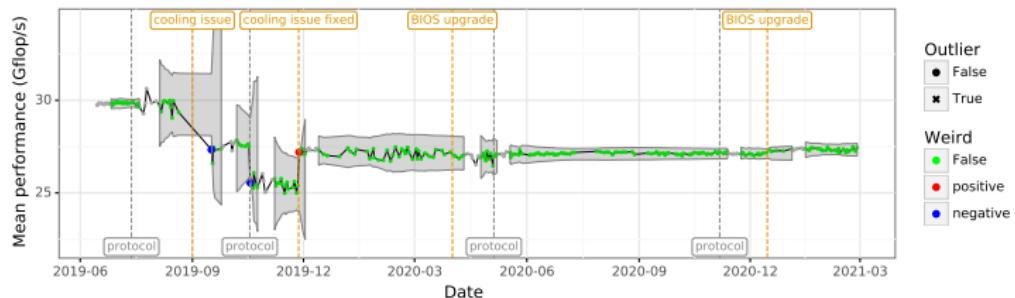
With several measures, using their **average** and shrinking the interval

Example with 5 measures (averages represented by crosses)



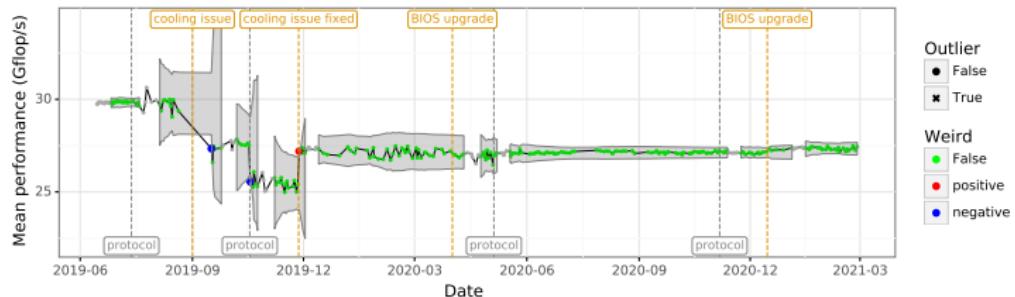
# RESULT: PERFORMANCE FLUCTUATION

## Performance fluctuation of the node dahu-14

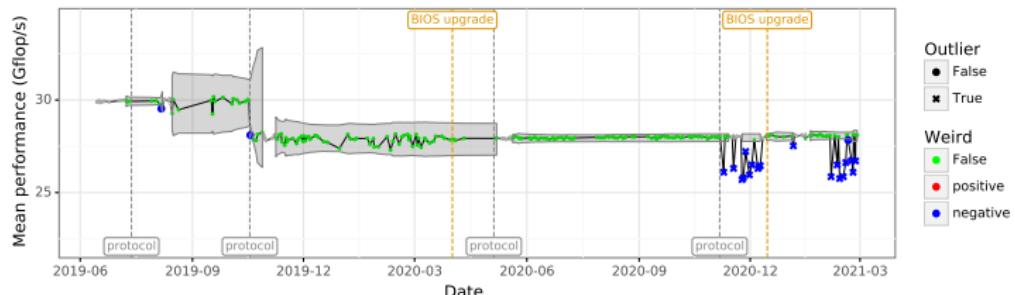


# RESULT: PERFORMANCE FLUCTUATION

Performance fluctuation of the node dahu-14

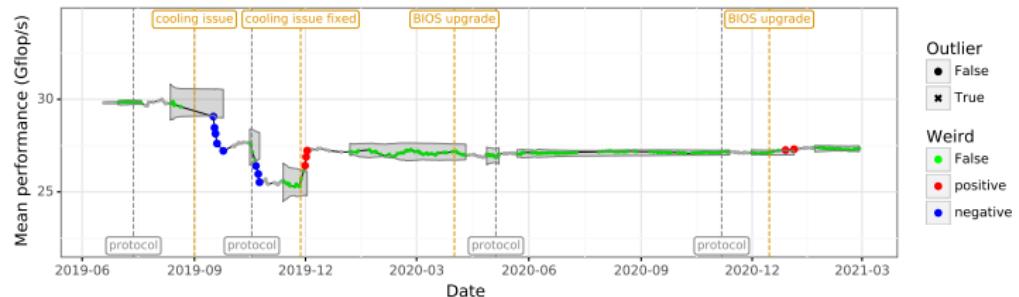


Performance fluctuation of the node dahu-32

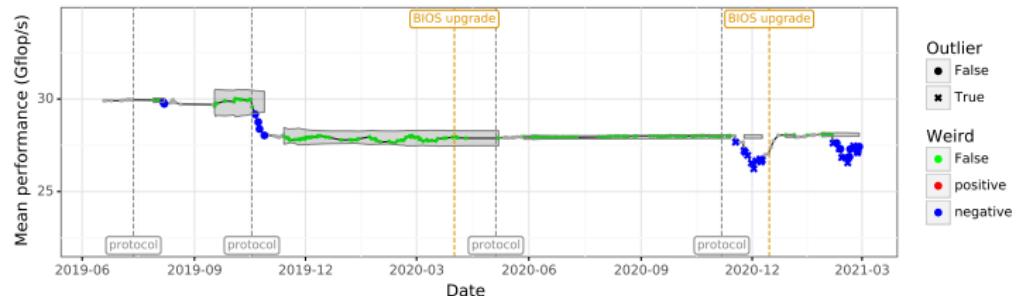


# RESULT: PERFORMANCE FLUCTUATION

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

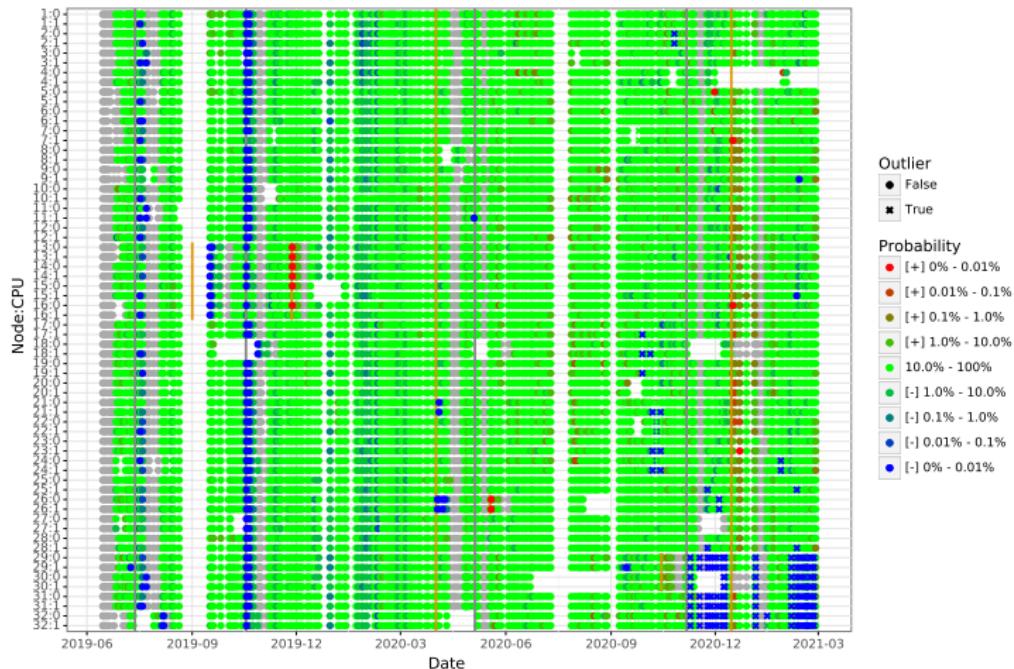


Performance fluctuation of the node dahu-32 (5-day window)



# RESULT: PERFORMANCE OVERVIEW

Overview of the performance on cluster dahu



# RESULT: PERFORMANCE OVERVIEW

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

