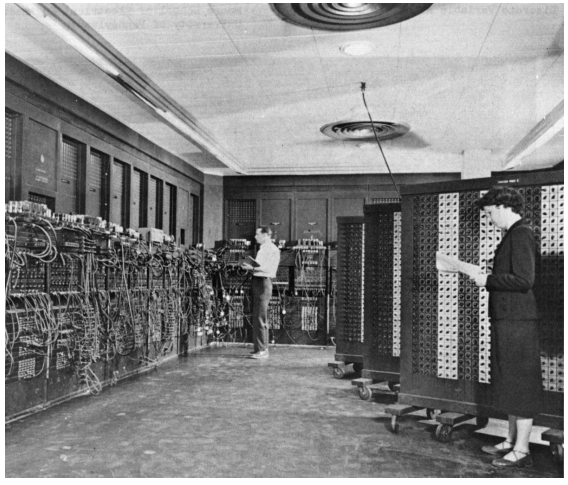
High Performance Computing: Towards Better Performance Predictions and Experiments

Tom Cornebize

2 June 2021, PhD defense

No science without computing

Arithmomètre (1851)

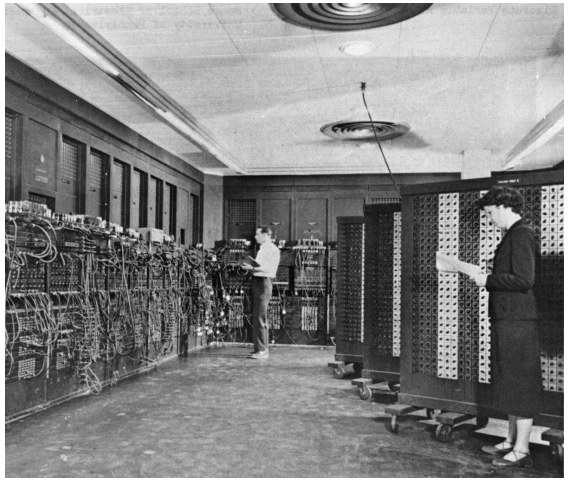


ENIAC (1945)

Fugaku (2021)

No science without computing

Arithmomètre (1851)



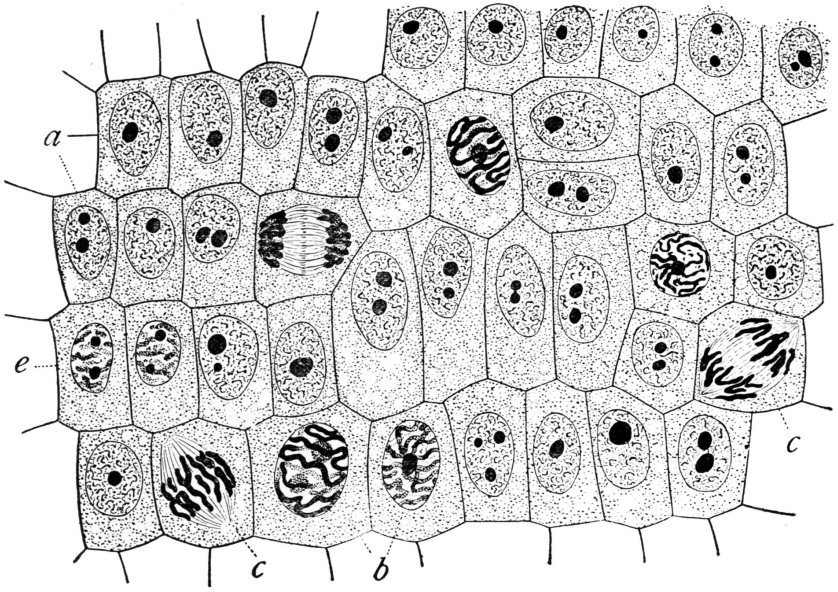
ENIAC (1945)

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



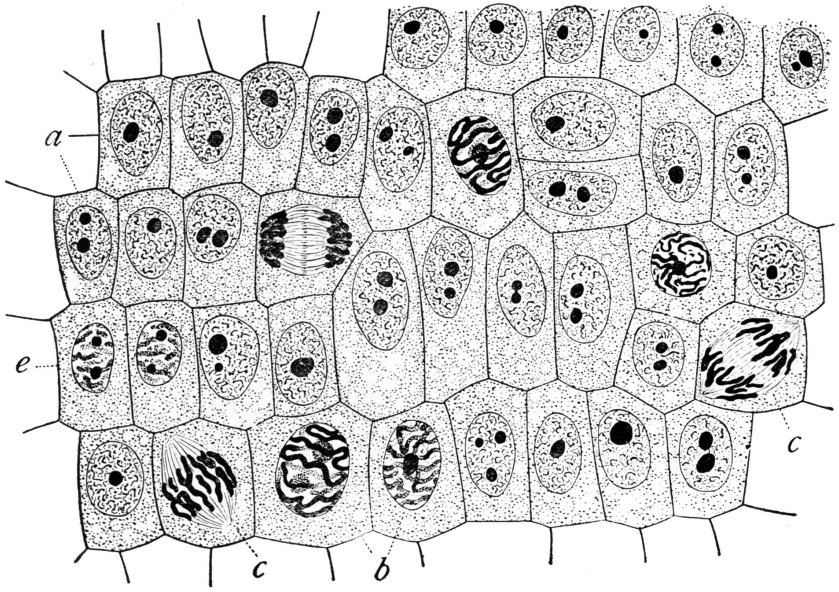
Similar to natural sciences

Complexity $\Rightarrow$ Variability and Opacity

$\Rightarrow$ No perfect model

$\Rightarrow$ Need for experiments

Experimental study of computer performance



Similar to natural sciences

Complexity $\Rightarrow$ Variability and Opacity

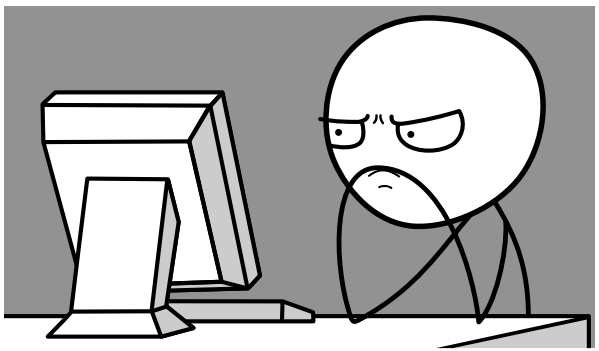
$\Rightarrow$ No perfect model

$\Rightarrow$ Need for experiments

Experiments can be carried in reality or in simulation

Context

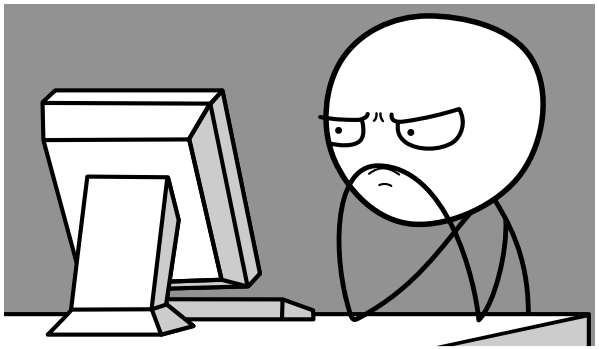
Typical Performance Evaluation Questions (Given my application and a supercomputer)



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

Context

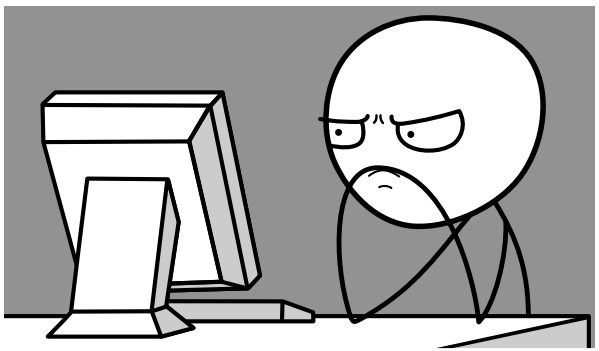
Typical Performance Evaluation Questions (Given my application and a supercomputer)



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

Context

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

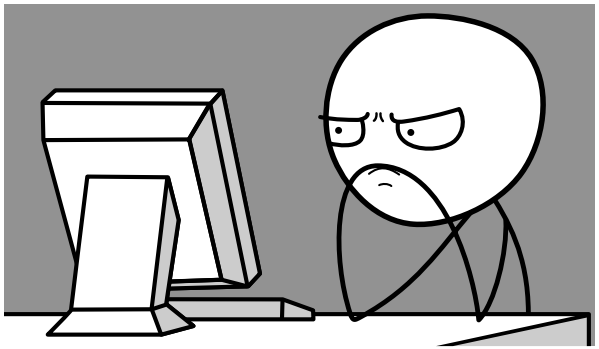


- **Before** running
  - How many nodes?
  - For how long?
  - Which parameters?
- **After** running
  - Performance as “expected”?
  - Problem in the app or the platform?

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

Context

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

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

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?

- Computations run 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

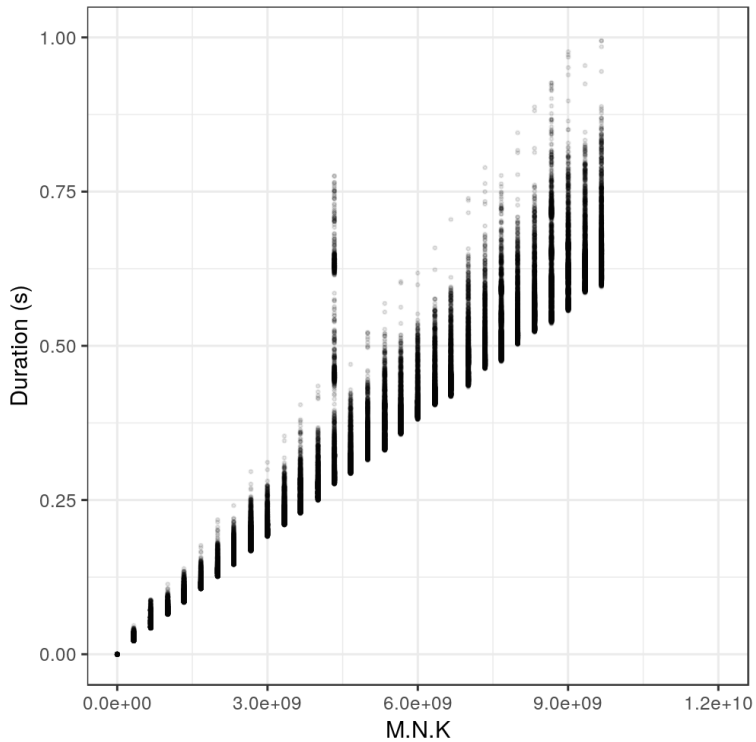
Emulation: how?

- 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

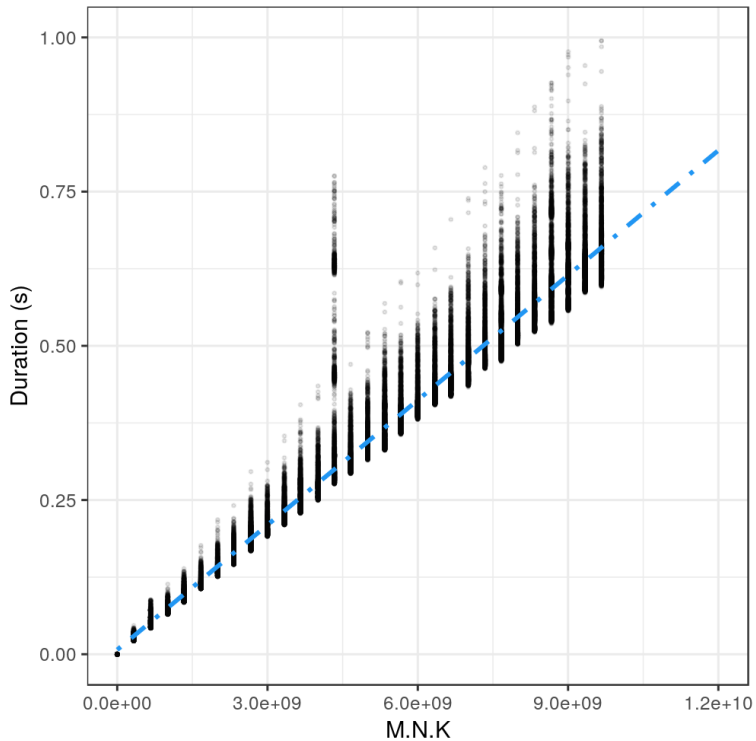
Modeling Commputations

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



Modeling Commputations

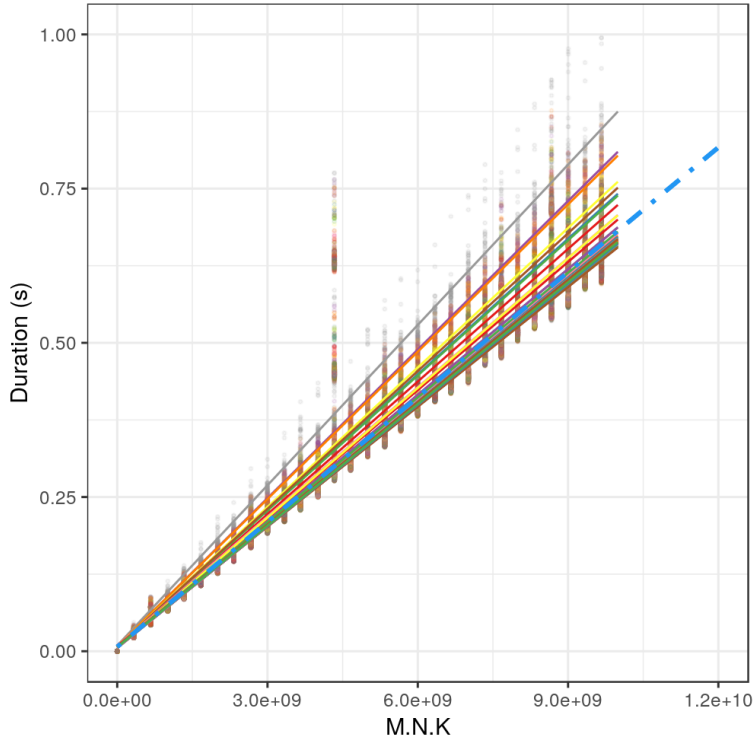
$$\texttt{dgemm}\,(M, N, K) = \alpha.M.N.K$$



Modeling Commputations

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

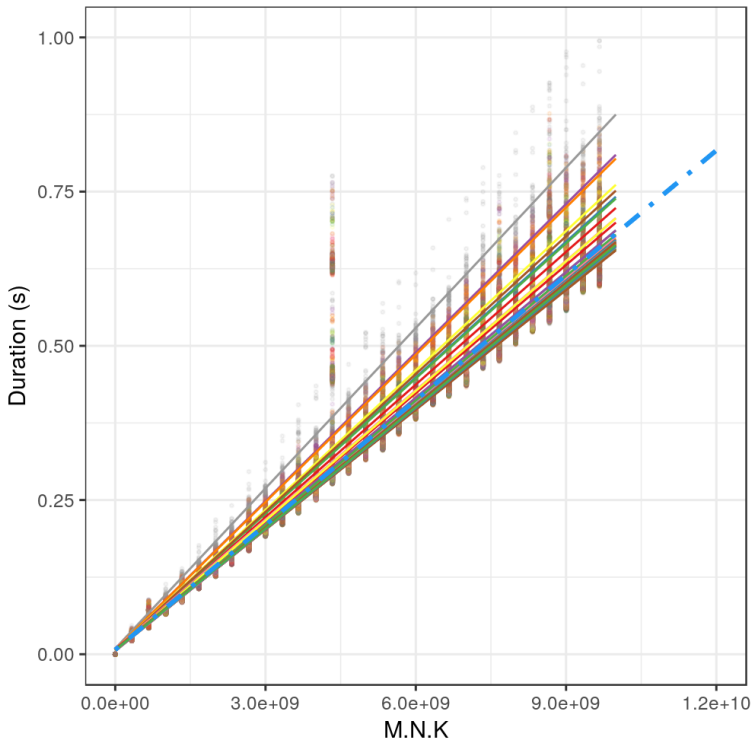
Different color $\Rightarrow$ different host



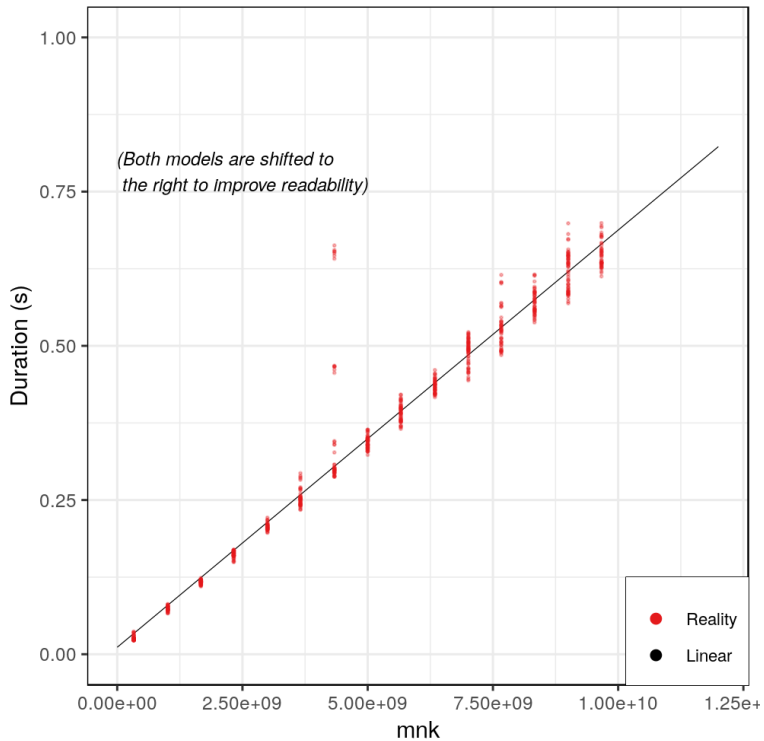
Modeling Commputations

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

Different color $\Rightarrow$ different host



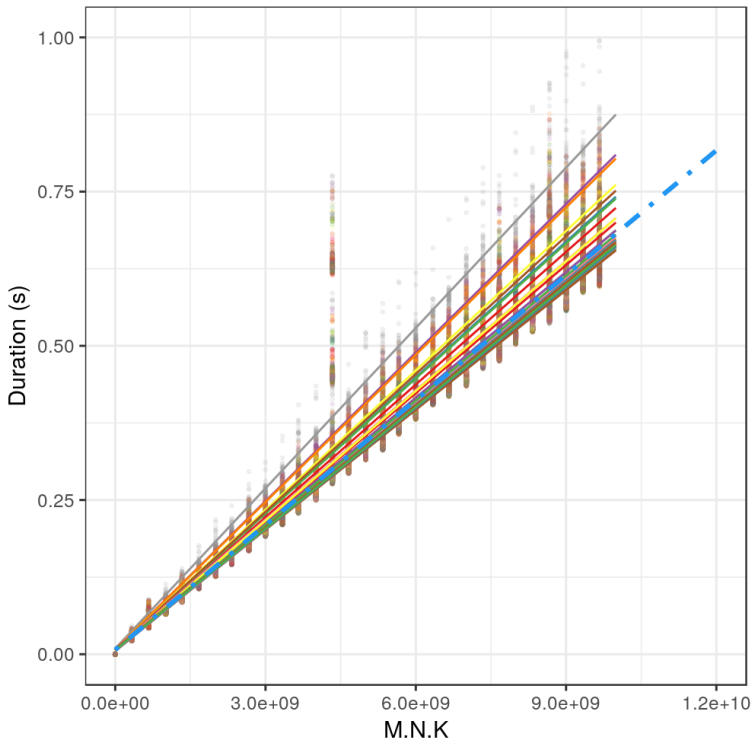
For a particular host



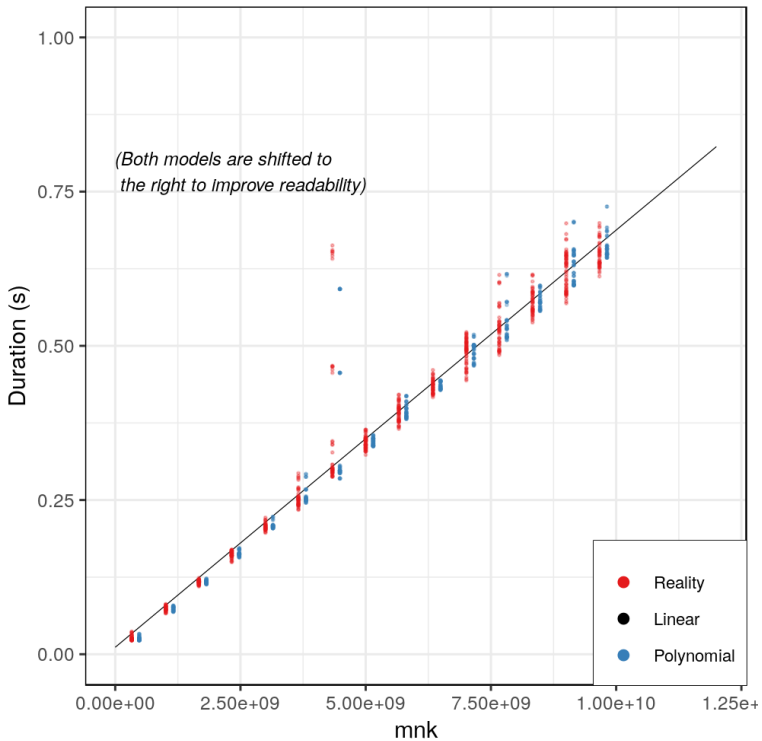
Modeling Commputations

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

Different color $\Rightarrow$ different host



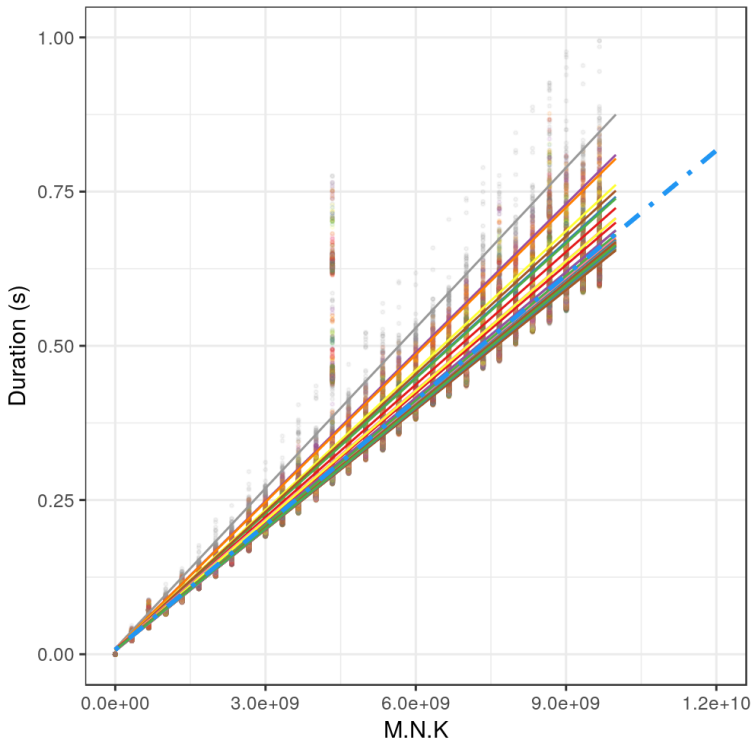
For a particular host



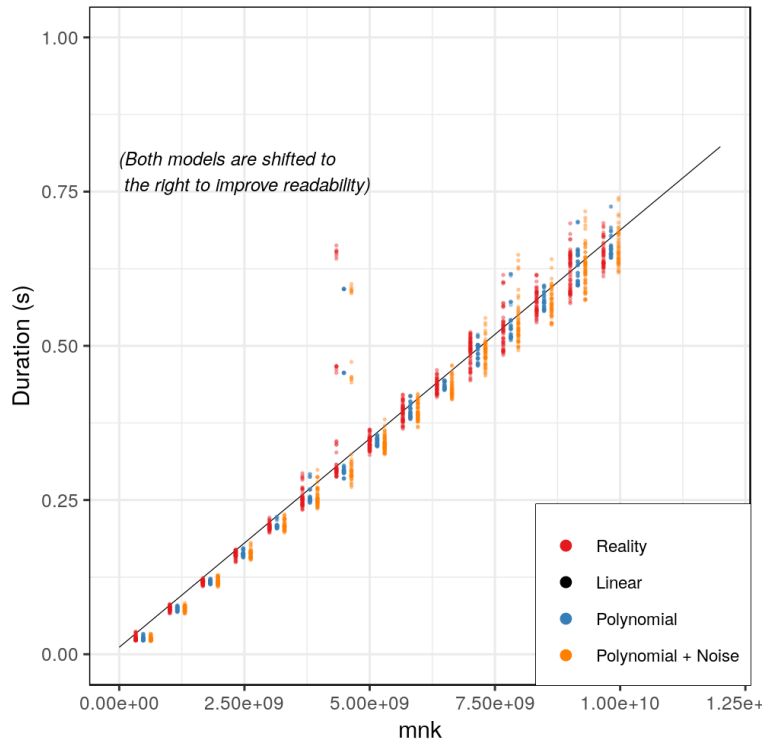
Modeling Commputations

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

Different color $\Rightarrow$ different host



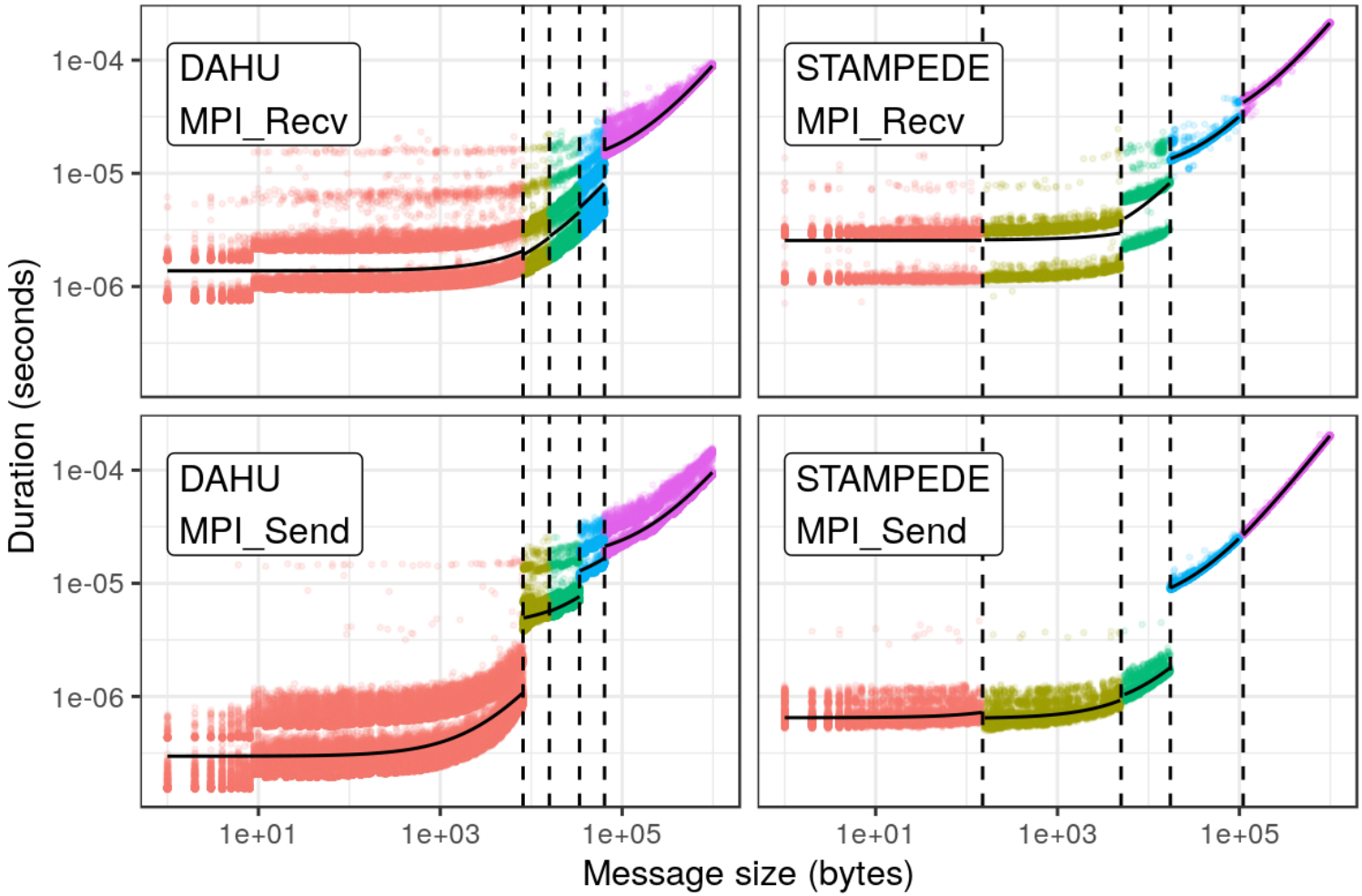
For a particular host



Modeling Communications

Hand-crafted non-blocking collective operations intertwinned with computations

Hand-crafted non-blocking collective operations intertwinned 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