

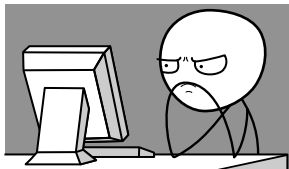
FAST AND FAITHFUL PERFORMANCE PREDICTION OF MPI APPLICATIONS: THE HPL CASE STUDY

Tom Cornebize, Arnaud Legrand, Franz C. Heinrich
Univ. Grenoble Alpes, CNRS, Inria

September 25, 2019

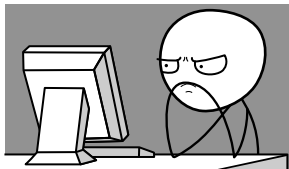


Typical Performance Evaluation Questions (Given my application and a supercomputer)



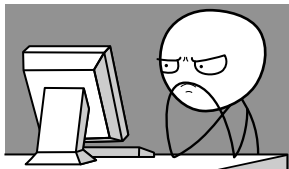
- **Before** running
 - How many nodes?
 - For how long?
 - Which parameters?

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?

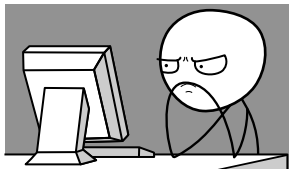
Typical Performance Evaluation Questions (Given my application and a supercomputer)



- **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)



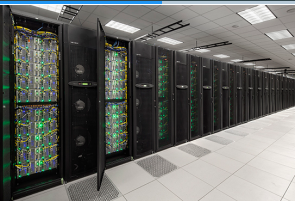
- **Before** running
 - How many nodes?
 - For how long?
 - Which parameters?
- **After** running
 - Performance as “expected”?
 - Problem in the app or the platform?

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

Holy Grail: Predictive Simulation on a “Laptop”

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

LET'S TRY HPL



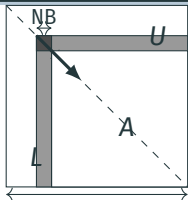
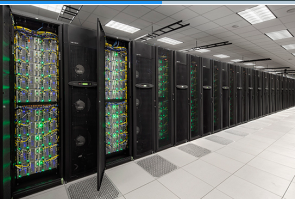
	Stampede@TACC	Theta@ANL	Dahu@G5K
Rpeak	8520.1 TFlop s ⁻¹	9627.2 TFlop s ⁻¹	62.26 TFlop s ⁻¹
<i>N</i>	3,875,000	8,360,352	500,000
<i>NB</i>	1024	336	128
<i>P</i> × <i>Q</i>	77 × 78 (6006)	32 × 101 (3232)	32 × 32 (1024)
RFACT [3]	Crout	Left	Right
SWAP [2]	Binary-exch.	Binary-exch.	Binary-exch.
BCAST [6]	Long modified	2 Ring modified	2 Ring
DEPTH	0	0	1
Rmax	5168.1 TFlop s ⁻¹	5884.6 TFlop s ⁻¹	24.55 TFlop s ⁻¹
Duration	2 hours	28 hours	1 hour
Memory	120 TB	559 TB	2 TB
MPI ranks	1/node	1/node	1/core

LET'S TRY HPL



	Stampede@TACC	Theta@ANL	Dahu@G5K
Rpeak	8520.1 TFlop s ⁻¹	9627.2 TFlop s ⁻¹	62.26 TFlop s ⁻¹
<i>N</i>	3,875,000	8,360,352	500,000
NB	1024	336	128
<i>P</i> × <i>Q</i>	77 × 78 (6006)	32 × 101 (3232)	32 × 32 (1024)
RFACT [3]	Crout	Left	Right
SWAP [2]	Binary-exch.	Binary-exch.	Binary-exch.
BCAST [6]	Long modified	2 Ring modified	2 Ring
DEPTH	0	0	1
Rmax	5168.1 TFlop s ⁻¹	5884.6 TFlop s ⁻¹	24.55 TFlop s ⁻¹
Duration	2 hours	28 hours	1 hour
Memory	120 TB	559 TB	2 TB
MPI ranks	1/node	1/node	1/core

LET'S TRY HPL

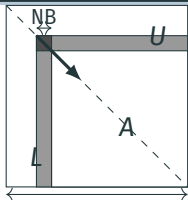
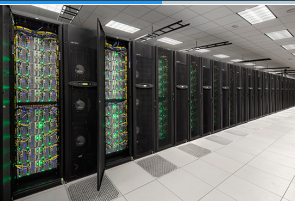


```

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;
    
```

	Stampede@TACC	Theta@ANL	Dahu@G5K
R_{peak}	8520.1 TFlop s^{-1}	9627.2 TFlop s^{-1}	62.26 TFlop s^{-1}
N	3,875,000	8,360,352	500,000
NB	1024	336	128
$P \times Q$	77 \times 78 (6006)	32 \times 101 (3232)	32 \times 32 (1024)
RFACT [3]	Crout	Left	Right
SWAP [2]	Binary-exch.	Binary-exch.	Binary-exch.
BCAST [6]	Long modified	2 Ring modified	2 Ring
DEPTH	0	0	1
R_{max}	5168.1 TFlop s^{-1}	5884.6 TFlop s^{-1}	24.55 TFlop s^{-1}
Duration	2 hours	28 hours	1 hour
Memory	120 TB	559 TB	2 TB
MPI ranks	1/node	1/node	1/core

LET'S TRY HPL



```

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;
    
```

	Stampede@TACC	Theta@ANL	Dahu@G5K
Rpeak	8520.1 TFlop s ⁻¹	9627.2 TFlop s ⁻¹	62.26 TFlop s ⁻¹
N	3,875,000	8,360,352	500,000
NB	1024	336	128
$P \times Q$	77×78 (6006)	32×101 (3232)	32×32 (1024)
RFACT [3]	Crout	Left	Right
SWAP [2]	Binary-exch.	Binary-exch.	Binary-exch.
BCAST [6]	Long modified	2 Ring modified	2 Ring
DEPTH	0	0	1
Rmax	5168.1 TFlop s ⁻¹	5884.6 TFlop s ⁻¹	24.55 TFlop s ⁻¹
Duration	2 hours	28 hours	1 hour
Memory	120 TB	559 TB	2 TB
MPI ranks	1/node	1/node	1/core



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 

- 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
- MPI ranks folded as **contexts** in a unique UNIX process
- Global variables **automatically privatized** (dlopen)

Goal: **predict** the performance of HPL,
both **efficiently** and **faithfully**

STEP 1: EMULATING AT SCALE

1. BLAS kernels

$$\text{DGEMM}(M, N, K) = \Theta(M.N.K)$$

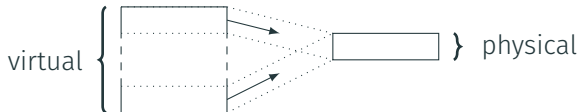
```
#define HPL_dgemm(layout, TransA, TransB, M, N, K,      \  
                alpha, A, lda, B, ldb, beta, C, ldc) {  \  
    double expected_time = M * N * K / (2 * flop_rate); \  
    smpi_execute_benched(expected_time);                \  
    })
```

STEP 1: EMULATING AT SCALE

1. BLAS kernels

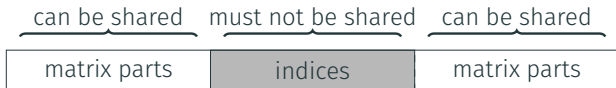
$$\text{DGEMM}(M, N, K) = \Theta(M.N.K)$$

2. Memory allocations (`SMPI_SHARED_MALLOC`)



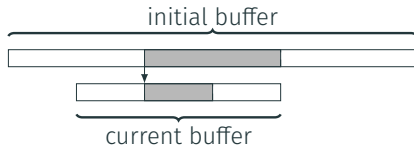
STEP 1: EMULATING AT SCALE

1. BLAS kernels $\text{DGEMM}(M, N, K) = \Theta(M.N.K)$
2. Memory allocations (`SMPI_SHARED_MALLOC`)
3. HPL specific tricks ([panel structure](#), reuse, huge pages, ...)



STEP 1: EMULATING AT SCALE

1. BLAS kernels $\text{DGEMM}(M, N, K) = \Theta(M.N.K)$
2. Memory allocations (`SMPI_SHARED_MALLOC`)
3. HPL specific tricks (panel structure, [reuse](#), huge pages, ...)



STEP 1: EMULATING AT SCALE

1. BLAS kernels $\text{DGEMM}(M, N, K) = \Theta(M.N.K)$
2. Memory allocations (`SMPI_SHARED_MALLOC`)
3. HPL specific tricks (panel structure, reuse, huge pages, ...)

Take-Away Message: It works! ($\approx 50/16,000$ lines in 14/150 files)

		Reality	Simulation
Dahu	#nodes / #processes	32 / 1024	1 / 1
	Memory	2 TB	9 GB
	Duration (hours)	1	5
	Resources (node hours)	32	5
Stampede	#nodes / #processes	6006 / 6006	1 / 1
	Memory	120 TB	19 GB
	Duration (hours)	2	62
	Resources (node hours)	12,000	62

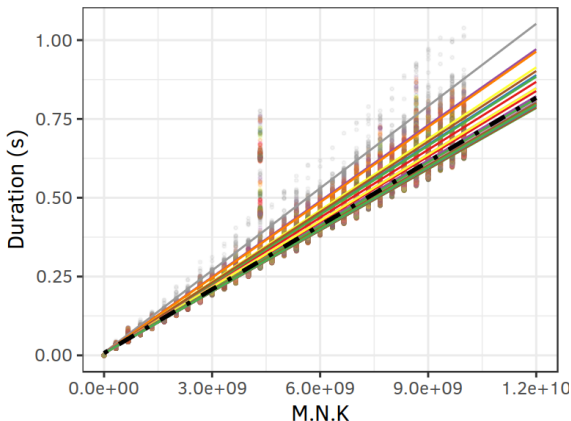
STEP 2: MODELING COMPUTATIONS

$$\text{DGEMM}(M, N, K) = \alpha.M.N.K$$

STEP 2: MODELING COMPUTATIONS

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

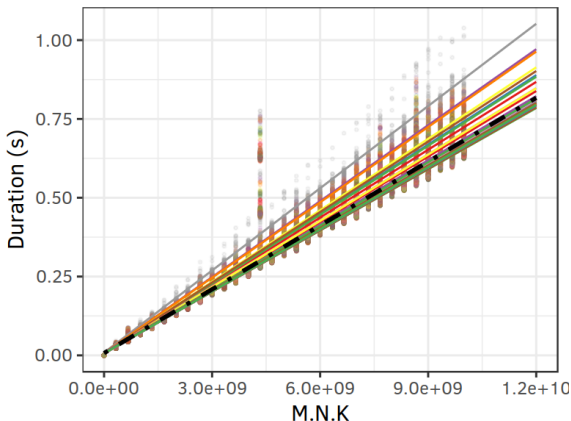
Different color \Rightarrow different host



STEP 2: MODELING COMPUTATIONS

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

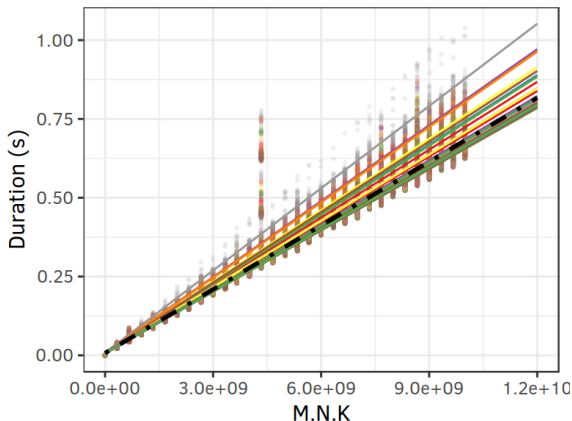
Different color \Rightarrow different host



STEP 2: MODELING COMPUTATIONS

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

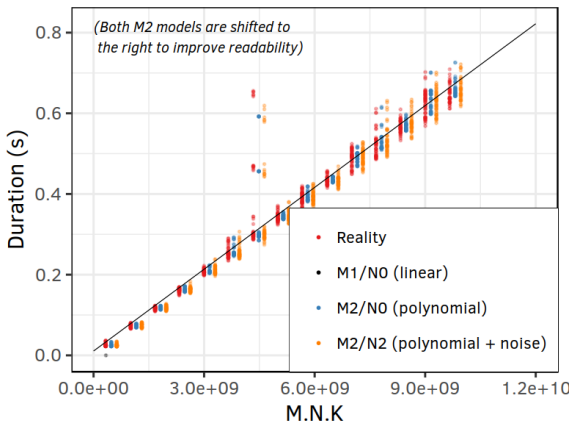
Different color \Rightarrow different host



STEP 2: MODELING COMPUTATIONS

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

For a particular host



STEP 2: MODELING COMPUTATIONS

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

Take-Away Message:

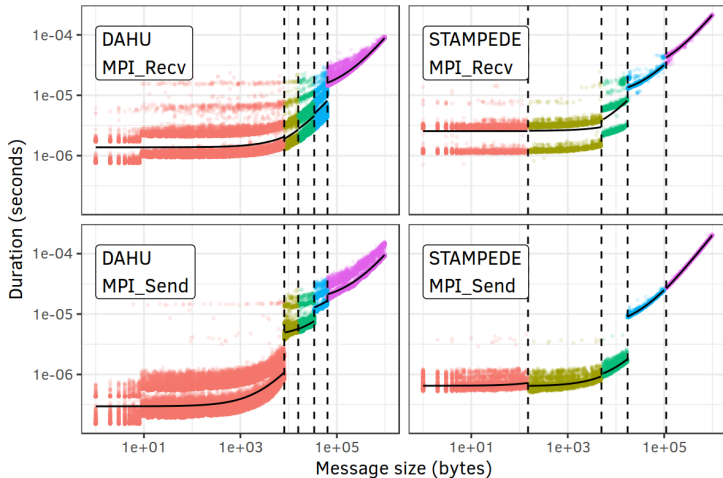
- Both **spatial** and **temporal** variability
- “Sophisticated” linear models are **excellent predictors** (for every function – DTRSM, DAXPY, ...)

STEP 2': MODELING COMMUNICATIONS

Hand-crafted non-blocking collective operations intertwined with computations

STEP 2': MODELING COMMUNICATIONS

Hand-crafted non-blocking collective operations intertwined with computations



STEP 2': MODELING COMMUNICATIONS

Hand-crafted non-blocking collective operations intertwined with computations

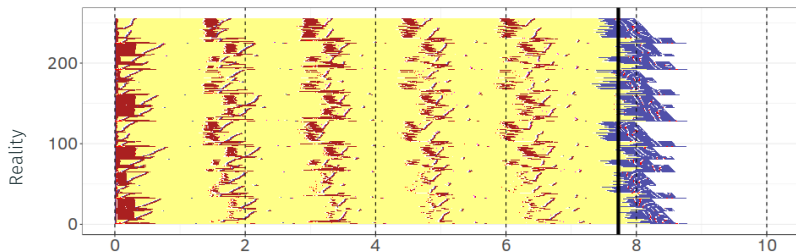
Take-Away Message:

- For small messages, the **variability can be huge**
- **Piece-wise mixture** of **linear** regressions

DOES ALL THIS MATTER?

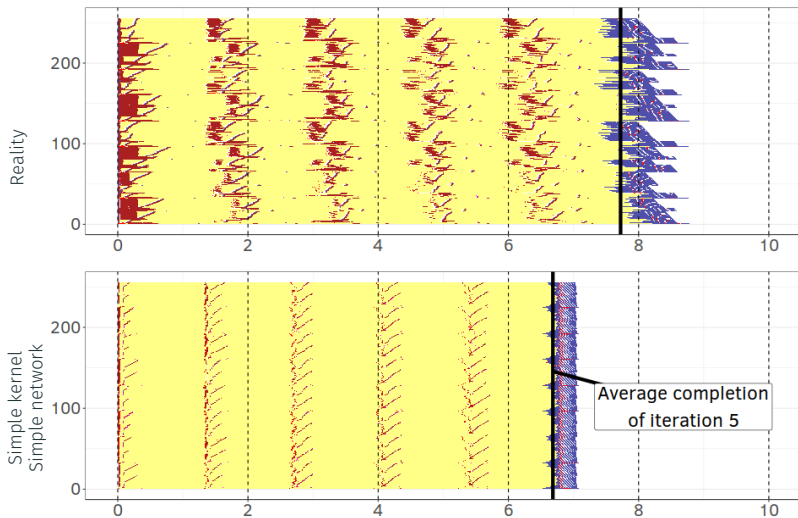
HPL STRUCTURE: PREDICTION VS. REALITY (DAHU @ G5K)

256 MPI ranks, interrupted after the 5th iteration



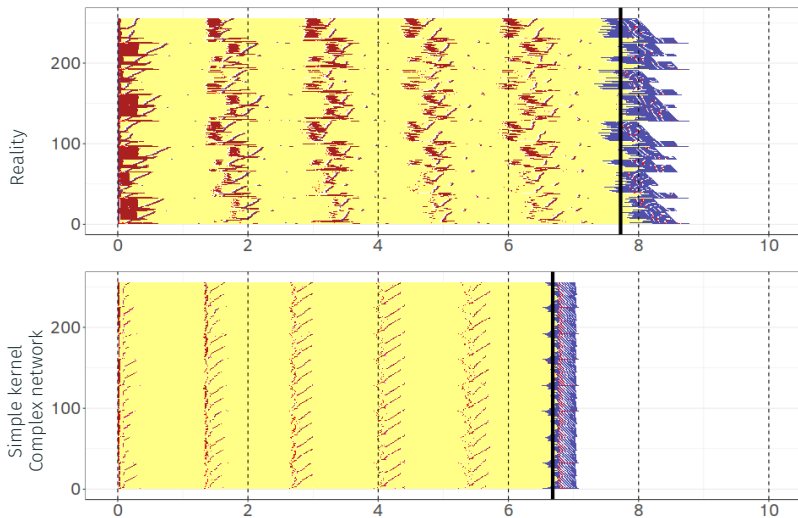
HPL STRUCTURE: PREDICTION VS. REALITY (DAHU @ G5K)

256 MPI ranks, interrupted after the 5th iteration



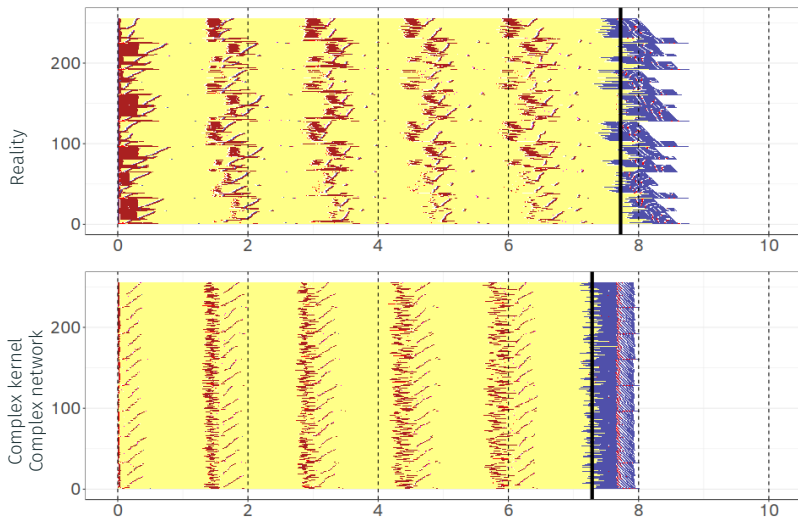
HPL STRUCTURE: PREDICTION VS. REALITY (DAHU @ G5K)

256 MPI ranks, interrupted after the 5th iteration

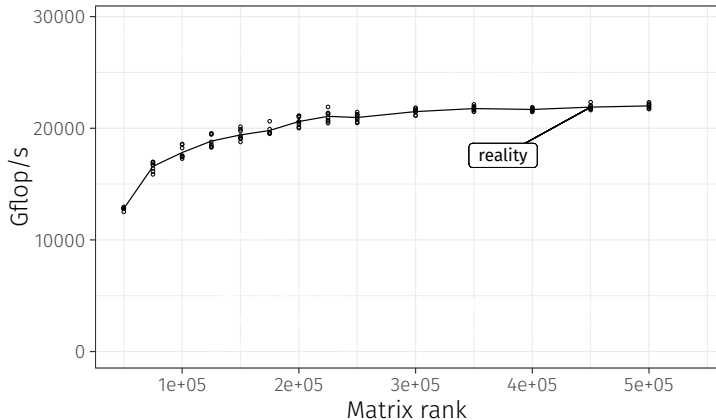


HPL STRUCTURE: PREDICTION VS. REALITY (DAHU @ G5K)

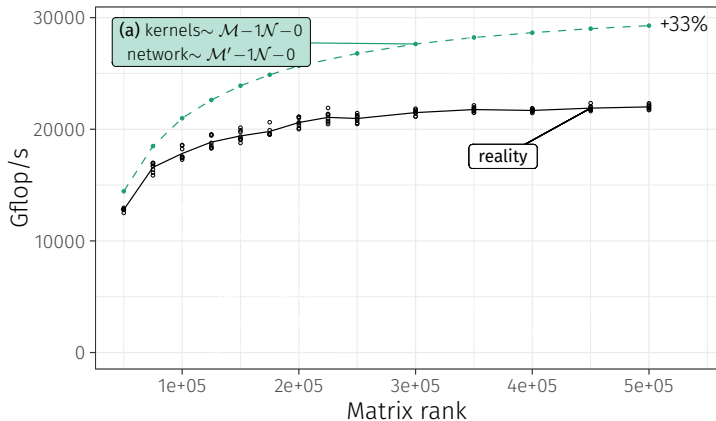
256 MPI ranks, interrupted after the 5th iteration



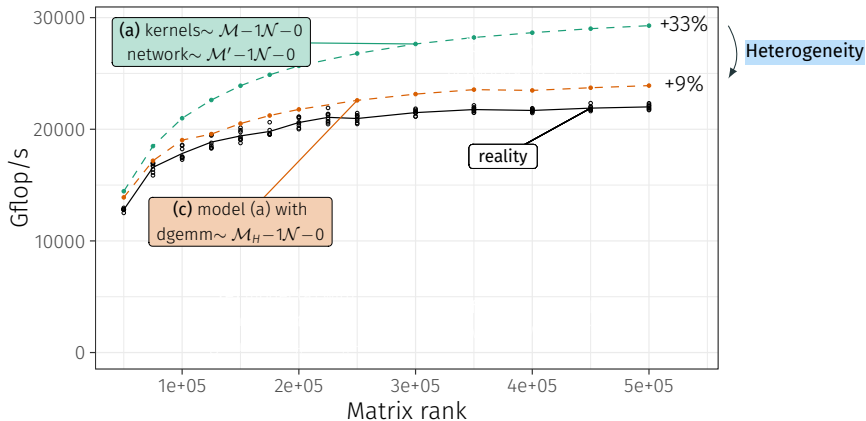
HPL PERFORMANCE: PREDICTION VS. REALITY (DAHU @ G5K)



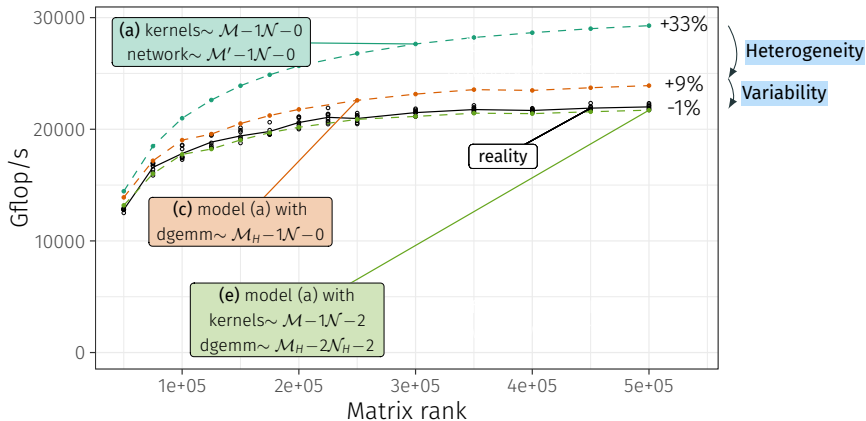
HPL PERFORMANCE: PREDICTION VS. REALITY (DAHU @ G5K)



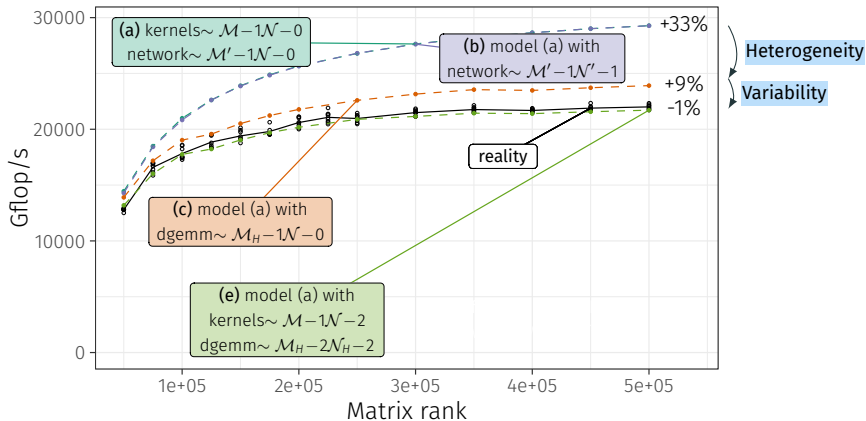
HPL PERFORMANCE: PREDICTION VS. REALITY (DAHU @ G5K)



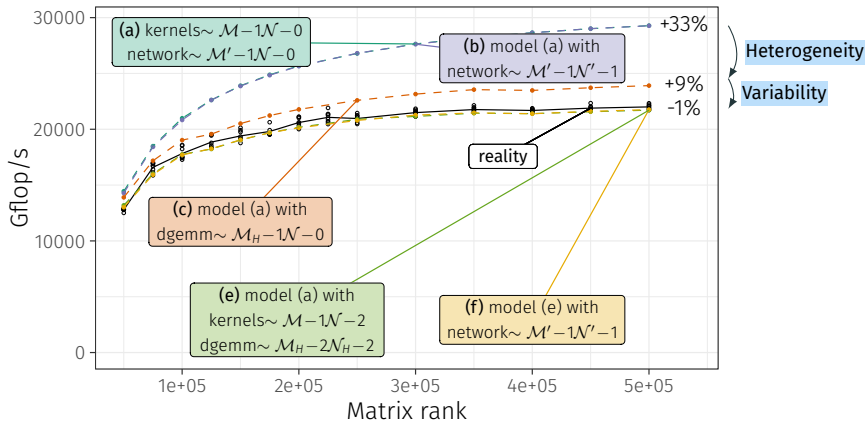
HPL PERFORMANCE: PREDICTION VS. REALITY (DAHU @ G5K)



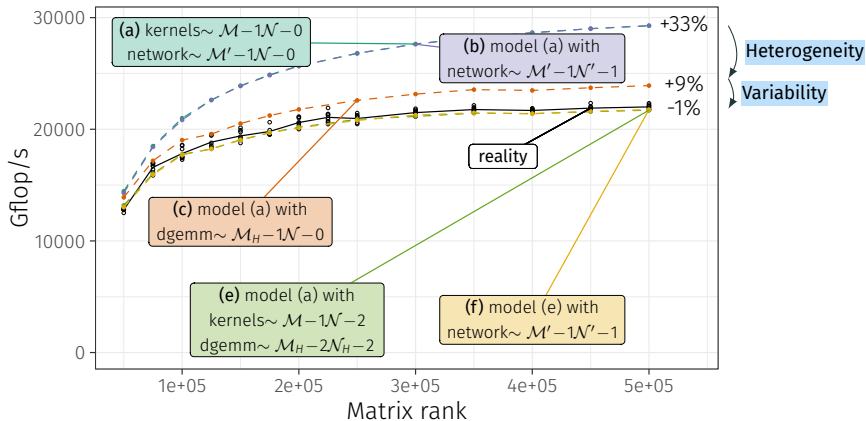
HPL PERFORMANCE: PREDICTION VS. REALITY (DAHU @ G5K)



HPL PERFORMANCE: PREDICTION VS. REALITY (DAHU @ G5K)



HPL PERFORMANCE: PREDICTION VS. REALITY (DAHU @ G5K)



Take-Away Message: accurate prediction

- Modeling both **spatial** and **temporal** computation variability is essential
- Network did not matter much here. But it could have...

PERSPECTIVES

Open Science: we know how to **predict HPL performance...**

- SimGrid is a **free software**: <https://simgrid.org>
- All the scripts, notebooks and data used for this paper are **publicly available**: <https://zenodo.org/record/3135479>

Open Science: we know how to **predict HPL performance...**

- SimGrid is a **free software**: <https://simgrid.org>
- All the scripts, notebooks and data used for this paper are **publicly available**: <https://zenodo.org/record/3135479>

Calibration: ...provided we have the right model (and overall, the right data)

- Requires to perform a (correct) calibration on the target platform
- Still needs expert knowledge for the calibration

Open Science: we know how to **predict HPL performance...**

- SimGrid is a **free software**: <https://simgrid.org>
- All the scripts, notebooks and data used for this paper are **publicly available**: <https://zenodo.org/record/3135479>

Calibration: ...provided we have the right model (and overall, the right data)

- Requires to perform a (correct) calibration on the target platform
- Still needs expert knowledge for the calibration

Modeling **complexity**

- **Spatial** variability was expected (**manufacturing** heterogeneity)
→ 20% performance loss
- **Temporal** variability is important (system noise, **temperature**)
→ 10% performance loss
- Nothing that specific to HPL in what we did

Generalizing our work: towards a SimBLAS library

- **Automatic calibration** of the major BLAS kernels
- **Injection** of their estimated durations in simulation

WHAT NEXT?

Generalizing our work: towards a **SimBLAS** library

- **Automatic calibration** of the major BLAS kernels
- **Injection** of their estimated durations in simulation

Leveraging our work: **predictive what-if** scenarios

- **Application tuning**

Which broadcast implementation should we use for this platform?

Which block size for the matrix?

- **Deployment tuning**


Quantify the impact on HPL performance of lowering CPU frequencies

What about removing the slowest node?

- **Platform sizing**

Quantify the impact of using a slower network


LET'S TRY HPL



Allocate and initialize A
for the $M \times N$ stencil
Allocate the panel
Broadcast the panel
Update the sub-matrix

	N	Stamps@GSK	Thes@GML	Dahu@GSK
Epwks	8200.1 Tllops	602.2 Tllops	82.28 Tllops	
N	3,875,000	8,360,352	500,000	
MB	10%	336	128	
P * Q	774 * 78 (6006)	224 * 201 (2222)	224 * 22 (2024)	
RFAC1 [3]	Cloud	Left	Right	
SWAP [2]	Binary-switch	Binary-switch	Binary-switch	
BCAST [6]	Long residual	2 Ring residual	2 Ring	
DEPTH	8	8	8	
Rmax	5168.1 Tllops * 1	5846.6 Tllops * 1	24.55 Tllops * 1	
Duration	2 hours	28 hours	1 hour	
Memory	3.28 TB	559 TB	2 TB	
MPI ranks	1/node	1/node	1/cwe	

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 simpic/simpirun

Emulation: how?

- Computations run for real on a laptop
- Communications are faked
- Performance model for the target platform
- MPI ranks folded as contexts in a unique UNIX process
- Global variables automatically privatized (`dlopen`)


STEP 1: EMULATING AT SCALE

BLAS kernels

$$DGEMM(M, N, K) = \Theta(M, N, K)$$

2. Memory allocations (`SMPi_SHARED_MALLOC`)

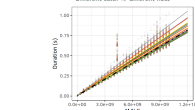
3. HPL specific tricks (panel structure, reuse, huge pages, ...) initial buffer




STEP 2: MODELING COMPUTATIONS

$$DGEMM(M, N, K) = \underbrace{\alpha \cdot M \cdot N \cdot K}_{\text{ops / host}} + \underbrace{\beta \cdot M \cdot N + \gamma \cdot N \cdot K + \dots}_{\text{polynomial model}} + \underbrace{+ N^2 \cdot \alpha^2 \cdot M \cdot N \cdot K + \dots}_{\text{polynomial noise}}$$

Different color => different host

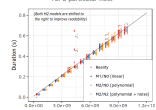




STEP 2: MODELING COMPUTATIONS

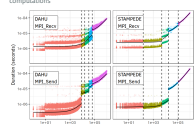
$$DGEMM(M, N, K) = \underbrace{\alpha \cdot M \cdot N \cdot K}_{\text{ops / host}} + \underbrace{\beta \cdot M \cdot N + \gamma \cdot N \cdot K + \dots}_{\text{polynomial model}} + \underbrace{+ N^2 \cdot \alpha^2 \cdot M \cdot N \cdot K + \dots}_{\text{polynomial noise}}$$

For a particular host



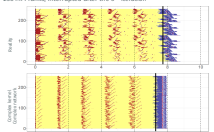
STEP 2': MODELING COMMUNICATIONS

Hand-crafted non-blocking collective operations intertwined with computations

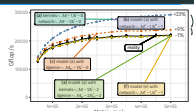


HPL STRUCTURE: PREDICTION VS. REALITY (DAHU @ GSK)

256 MPI ranks, interrupted after the 5th iteration



HPL PERFORMANCE: PREDICTION VS. REALITY (DAHU @ GSK)



Take-Away Message: **accurate prediction**

- Modeling both **spatial** and **temporal** computation variability is essential
- Network did not matter much here. But it could have...

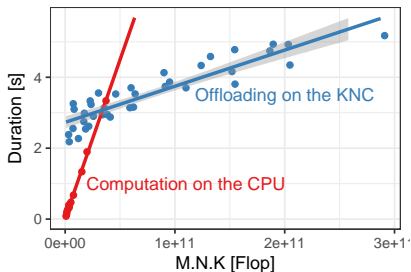
✓ Prediction of HPL performance: **efficiently** and **faithfully**

✉ tom.cornebe@univ-grenoble-alpes.fr

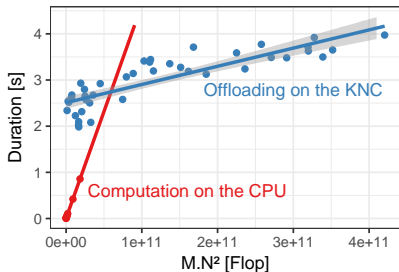
📄 <https://zenodo.org/record/3135479>

STAMPEDE ARCHEOLOGY (2013): DOWN THE RABBIT HOLE

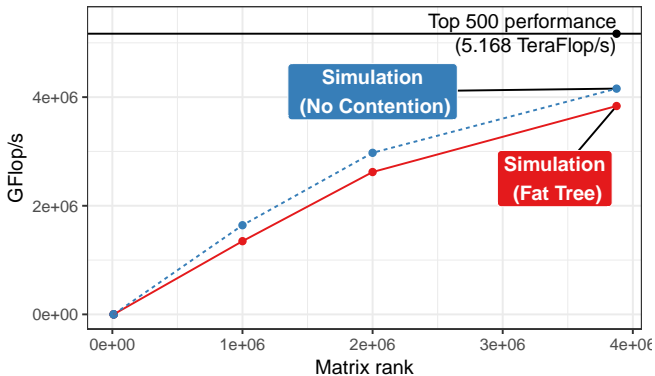
Duration of DGEMM



Duration of DTRSM



STAMPEDE ARCHEOLOGY (2013): DOWN THE RABBIT HOLE

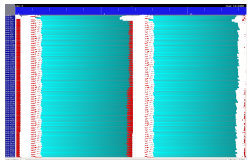


STAMPEDE ARCHEOLOGY (2013): DOWN THE RABBIT HOLE

```
=====
HPLinpack 2.1 -- High-Performance Linpack benchmark -- October 26, 2012
Written by A. Petitet and R. Clint Whaley, Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
=====
```

The following parameter values will be used:

```
N          : 3875000
NB         : 1024
PMAP      : Column-major process mapping
...
BCAST     : BlongM
DEPTH     : 0
SWAP      : Binary-exchange
...
```



Take-Away Message:

- Intel HPL was used (HPL_bcast_bpush, non-blocking sends)
- The reported output is wrong (total Update time \gg makespan)