Fast and Faithful Performance Prediction of MPI Applications: the HPL Case Study

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

June 25, 2019

Context

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

- Before running
  - How many nodes ? For how long ?
  - Which parameters / geometry / communication pattern ?
- After running (performance does not match my ”expectations”)
  - Does it come from my app or from the platform ?
  - What could I do to fix the problem (if any) ?

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

Context

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

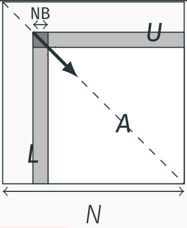
- Before running
  - How many nodes ? For how long ?
  - Which parameters / geometry / communication pattern ?
- After running (performance does not match my ”expectations”)
  - Does it come from my app or from the platform ?
  - What could I do to fix the problem (if any) ?

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

Holly Grail: Predictive Simulation on a ”Laptop” Capture the whole application and platform complexity

- Run the code (~~skeleton~~)
- Use sound performance models

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	32×32
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

Step 1: Emulating at Scale

SMPI = controled emulation of MPI programs using SimGrid

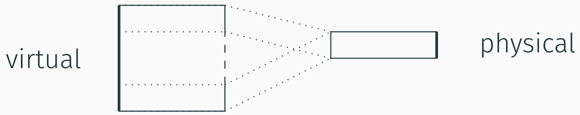
1. BLAS kernels $\texttt{DGEMM}(M, N, K) = \Theta(M.N.K)$

```
#define HPL_dgemm(layout, TransA, TransB, M, N, K,    \
           alpha, A, lda, B, ldb, beta, C, ldc) ({    \
   double expected_time = 1.029e-11 * M * N * K;      \
   smpi_execute_benched(expected_time);               \
  })
```

Step 1: Emulating at Scale

SMPI = controled emulation of MPI programs using SimGrid

1. BLAS kernels $\texttt{DGEMM}(M, N, K) = \Theta(M.N.K)$
2. Memory allocations (`SMPI_SHARED_MALLOC`)



Step 1: Emulating at Scale

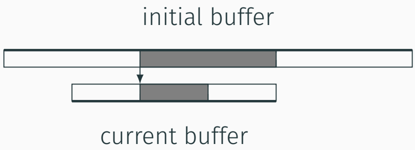
SMPI = controled emulation of MPI programs using SimGrid

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

can be shared	must not be shared	can be shared
matrix parts	indices	matrix parts

SMPI = controled emulation of MPI programs using SimGrid

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



Step 1: Emulating at Scale

SMPI = controled emulation of MPI programs using SimGrid

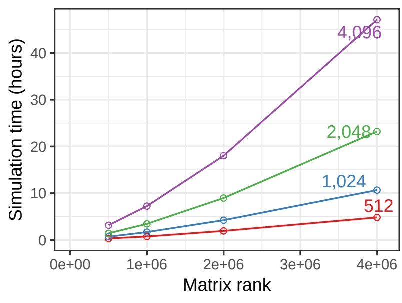
1. BLAS kernels $\texttt{DGEMM}(M, N, K) = \Theta(M.N.K)$

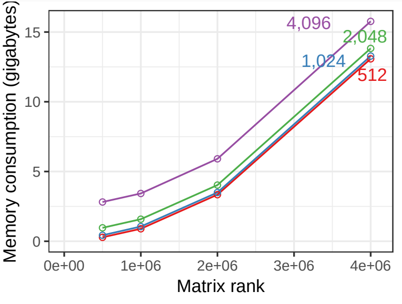
2. Memory allocations (`SMPI_SHARED_MALLOC`)

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

Reality: Computations $= \Theta(N^3)$ Communications $= \Theta(N^2)$

Simulation: Duration $\approx \Theta(N^2.Procs)$





Step 1: Emulating at Scale

SMPI = controled emulation of MPI programs using SimGrid

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

Take-Away Message: It works! (≈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	1
Stampede	#nodes / #processes	6006 / 6006	1 / 1
	Memory	120 TB	19 GB
	Duration (hours)	2	62
	Resources (node hours)	12,000	62

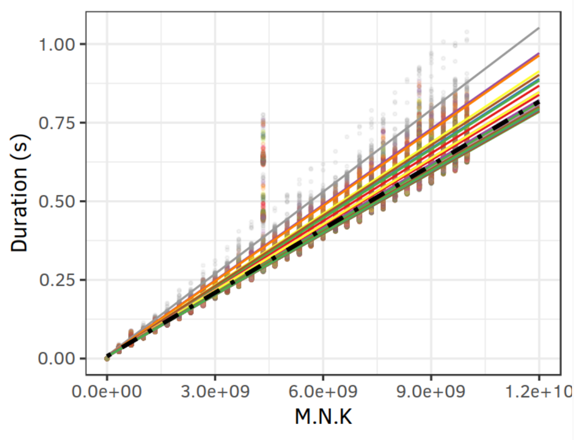
Step 2: Modeling Commputations

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

Step 2: Modeling Commputations

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

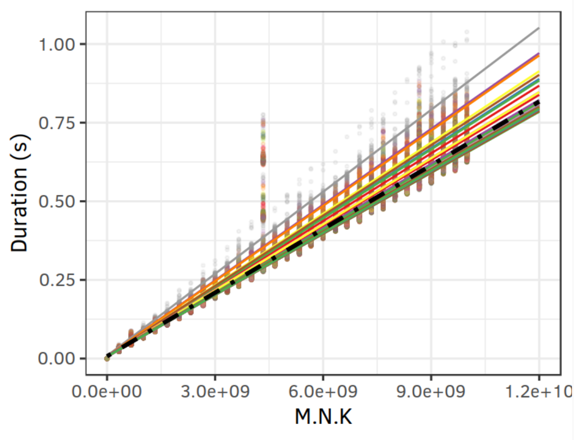
Different color ⇒ different host



Step 2: Modeling Commputations

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

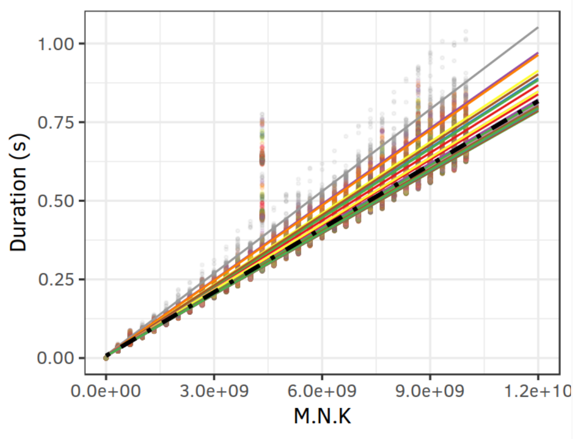
Different color ⇒ different host



Step 2: Modeling Commputations

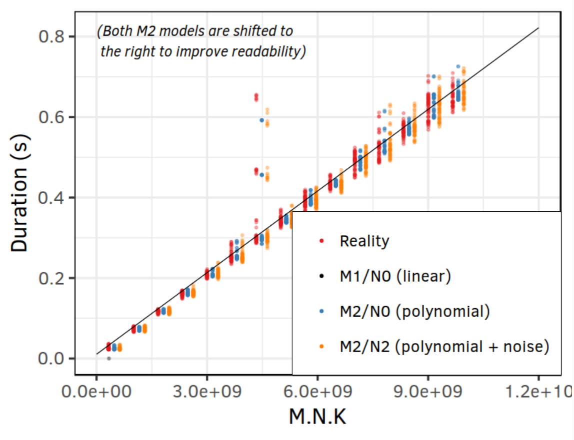
$$\texttt{DGEMM}_i(M,N,K) = \underbrace{\alpha_i.M.N.K}_{\text{per host}} + \underbrace{\beta_i.M.N + \gamma_i.N.K + \dots}_{\text{polynomial model}} + \underbrace{\mathcal{N}(0, \alpha'_i.M.N.K + \dots)}_{\text{polynomial noise}}$$

Different color $\Rightarrow$ different host



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

For a particular host



Step 2: Modeling Commputations

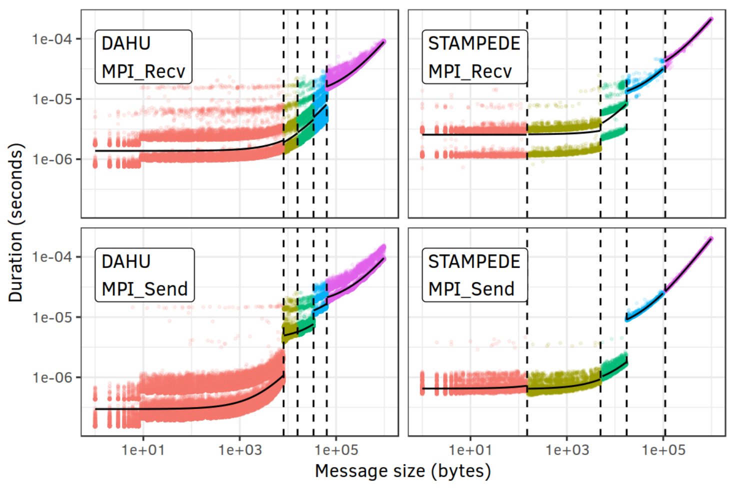
$$\texttt{DGEMM}_i(M,N,K) = \underbrace{\alpha_i.M.N.K}_{\text{per host}} + \underbrace{\beta_i.M.N + \gamma_i.N.K + \dots}_{\text{polynomial model}} + \underbrace{\mathcal{N}(0, \alpha'_i.M.N.K + \dots)}_{\text{polynomial noise}}$$

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 intertwinned with computations



Step 2': Modeling Communications

Hand-crafted non-blocking collective operations intertwinned with computations

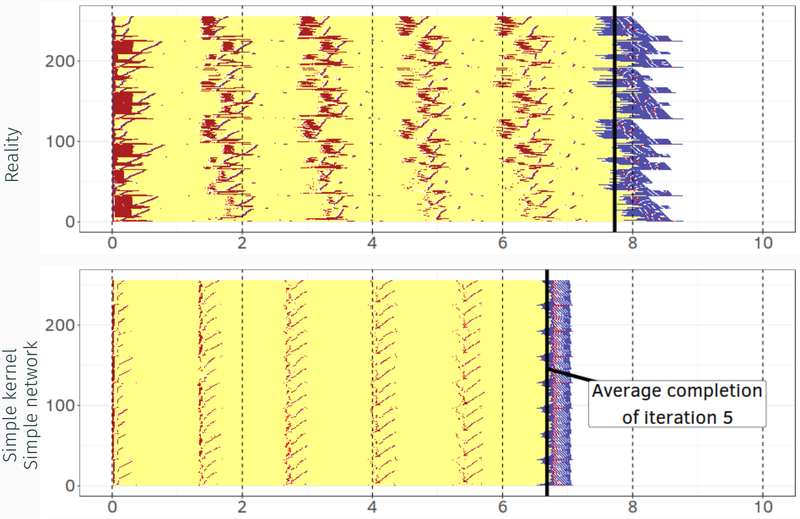
Take-Away Message:

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

Does all this matter ?

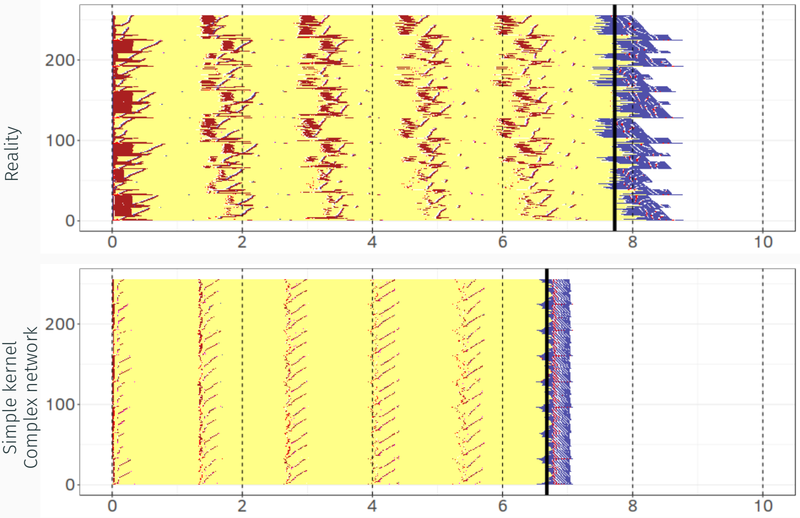
HPL Structure: Prediction vs. Reality (Dahu @ G5K)

32 nodes (2 Intel Xeon Gold 6130 CPU with 16 cores each), Omnipath



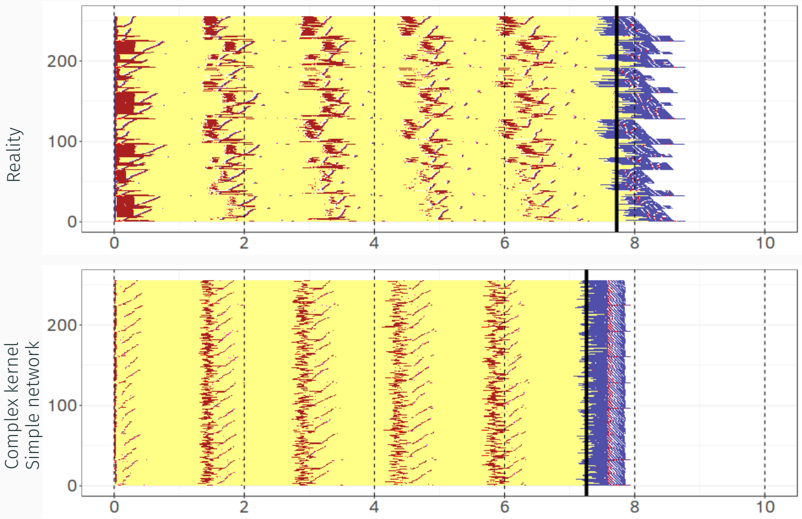
HPL Structure: Prediction vs. Reality (Dahu @ G5K)

32 nodes (2 Intel Xeon Gold 6130 CPU with 16 cores each), Omnipath

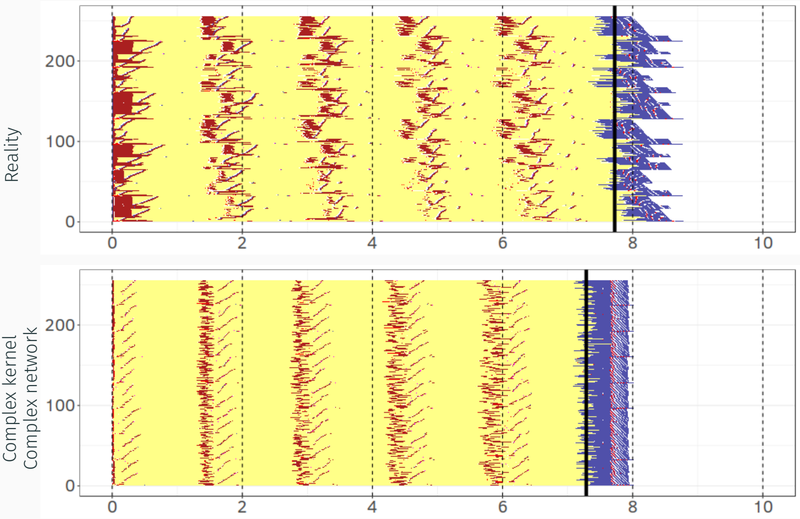


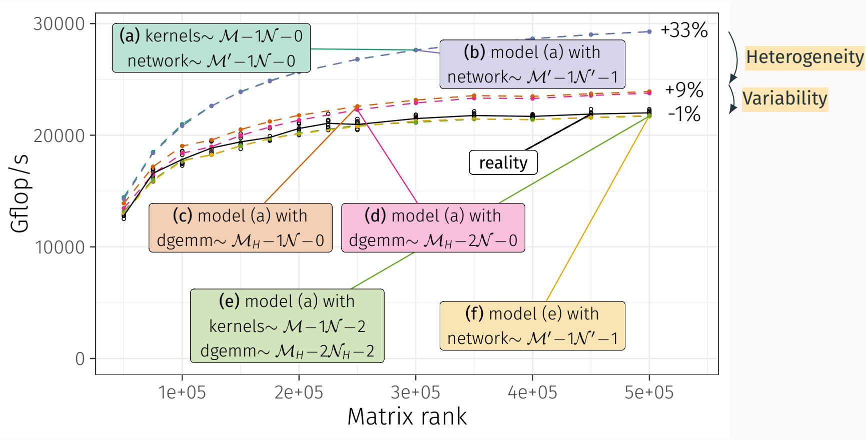
HPL Structure: Prediction vs. Reality (Dahu @ G5K)

32 nodes (2 Intel Xeon Gold 6130 CPU with 16 cores each), Omnipath



32 nodes (2 Intel Xeon Gold 6130 CPU with 16 cores each), Omnipath

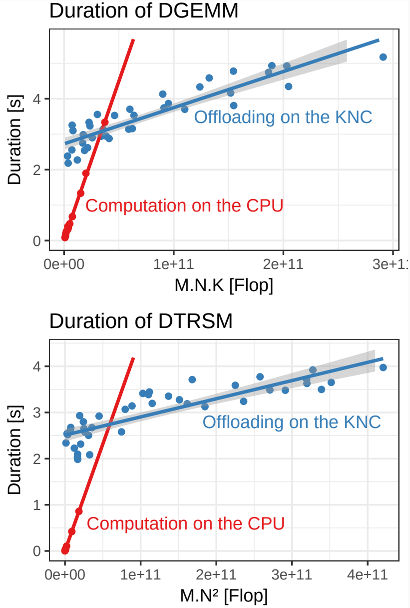




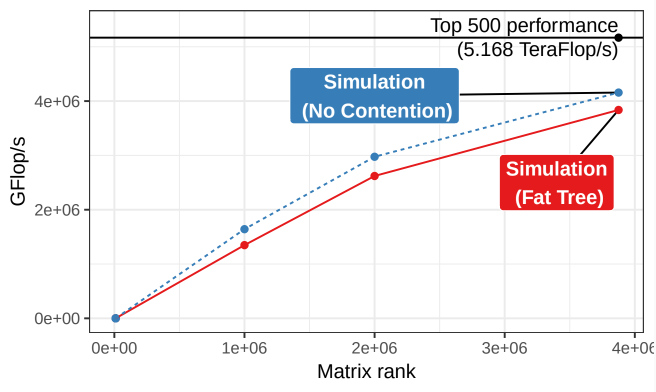
Take-Away Message: accurate prediction

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

Stampede Archeology (2013): Down the Rabbit Hole

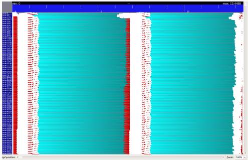


Stampede Archeology (2013): Down the Rabbit Hole



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 input is wrong (total `Update` time $\gg$ makespan)

Perspectives

Stepping Back (1/2)

HPLinpack vs. Intel HPL We have a good HPL ”surrogate”

- Modeling complexity:
  - Spatial variability was expected
  - Temporal variability is important (system noise, temperature)
  - Only `DGEMM` requires a faithful model
- I’m sick of open secrets (Ghidra, NSA reverse engineering)
  - Anyone interested in helping with a large-scale validation or useful applications?

Calibrating a platform toward a `libsimblas` and SMPI calibration

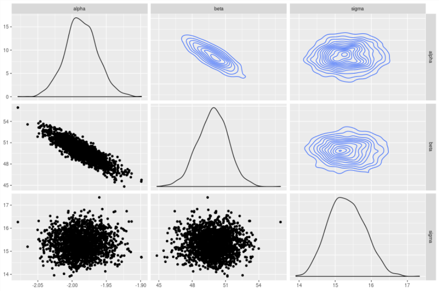
- Generic fitting through Bayesian sampling with STAN

$$\underbrace{y}_{\texttt{DGEMM}\text{ duration}} \sim \underbrace{\mathcal{M}}_{\text{Polynomial}}(\underbrace{\theta}_{\text{Parameters}}, \underbrace{x}_{M,N,K})$$

Calibrating a platform toward a `libsimblas` and SMPI calibration

- Generic fitting through Bayesian sampling with STAN

$$\underbrace{y}_{\texttt{DGEMM}\text{ duration}} \sim \underbrace{\mathcal{M}}_{\text{Polynomial}}(\underbrace{\theta}_{\text{Parameters}}, \underbrace{x}_{M,N,K})$$



Stepping Back (2/2)

Calibrating a platform toward a `libsimblas` and SMPI calibration

- Generic fitting through Bayesian sampling with STAN

$$\underbrace{y}_{\texttt{DGEMM}\text{ duration}} \sim \underbrace{\mathcal{M}}_{\text{Polynomial}}(\underbrace{\theta}_{\text{Parameters}}, \underbrace{x}_{M,N,K})$$

- Hierarchical modeling to extrapolate from a few machines

$$\boxed{\begin{array}{l} y_i \sim \mathcal{M}(\theta_i, x) \text{ for each } i \\ \theta_i \sim \mathcal{M}'(\theta') \quad \text{(e.g., Gaussian Mixture)} \end{array}}$$