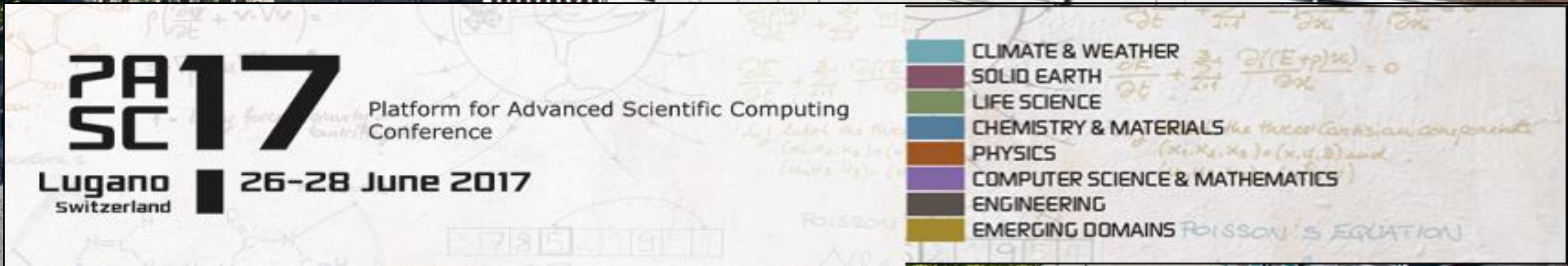
ETH zürich

spcl.inf.ethz.ch
@spcl_eth

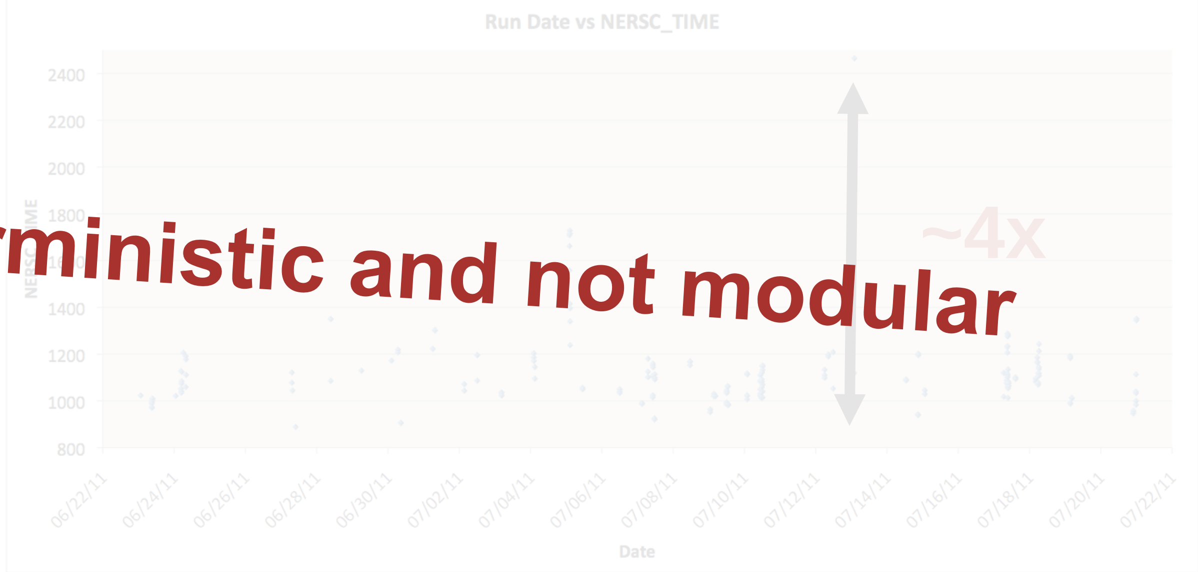
TORSTEN HOEFLER

Theory and practice in HPC: Modeling, Programming, and Networking

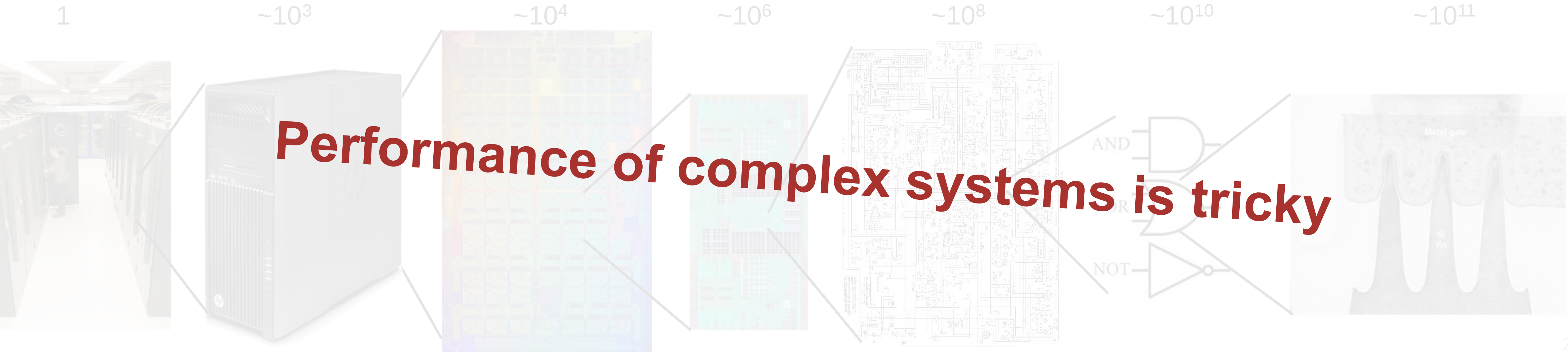
Systems@ETH zürich



High Performance Computing Practice

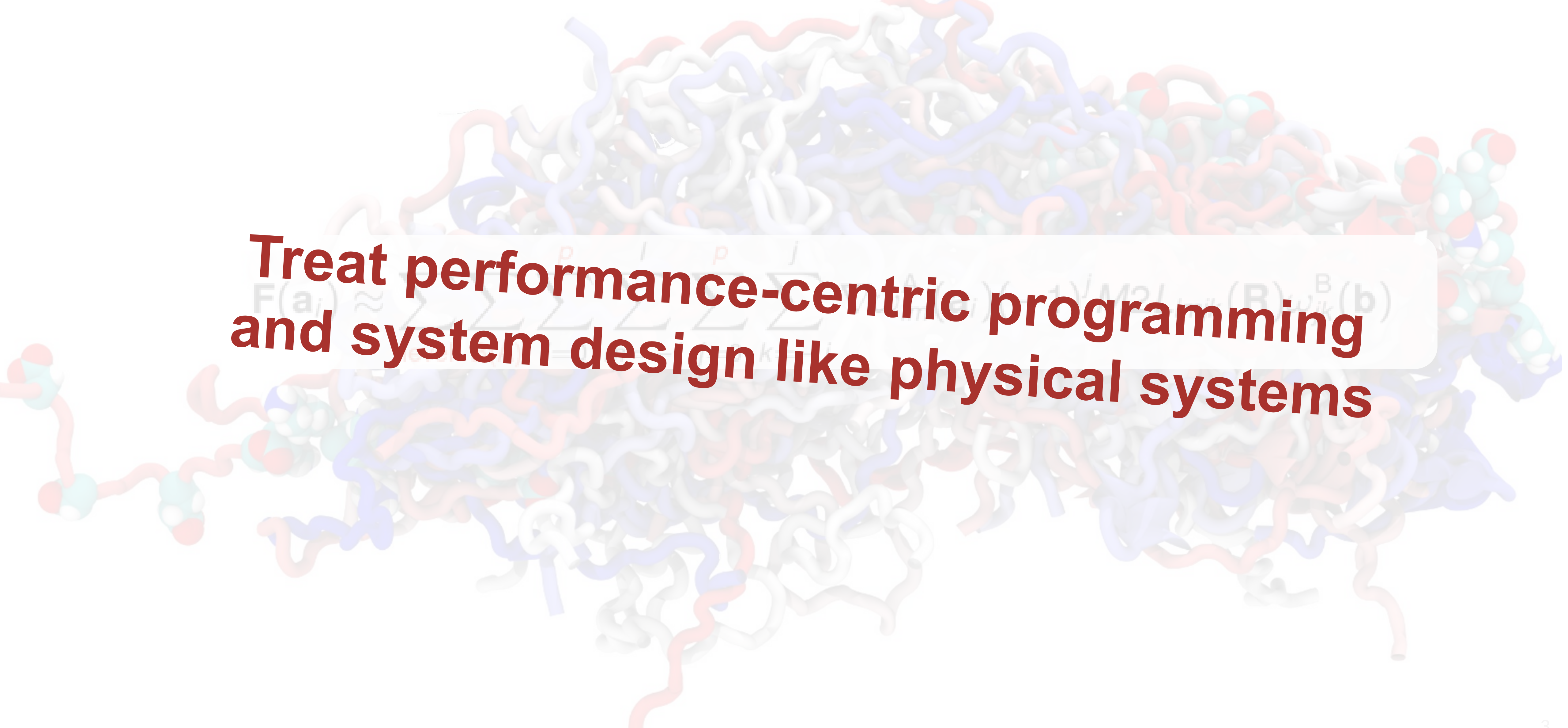


Performance is nondeterministic and not modular



Performance of complex systems is tricky

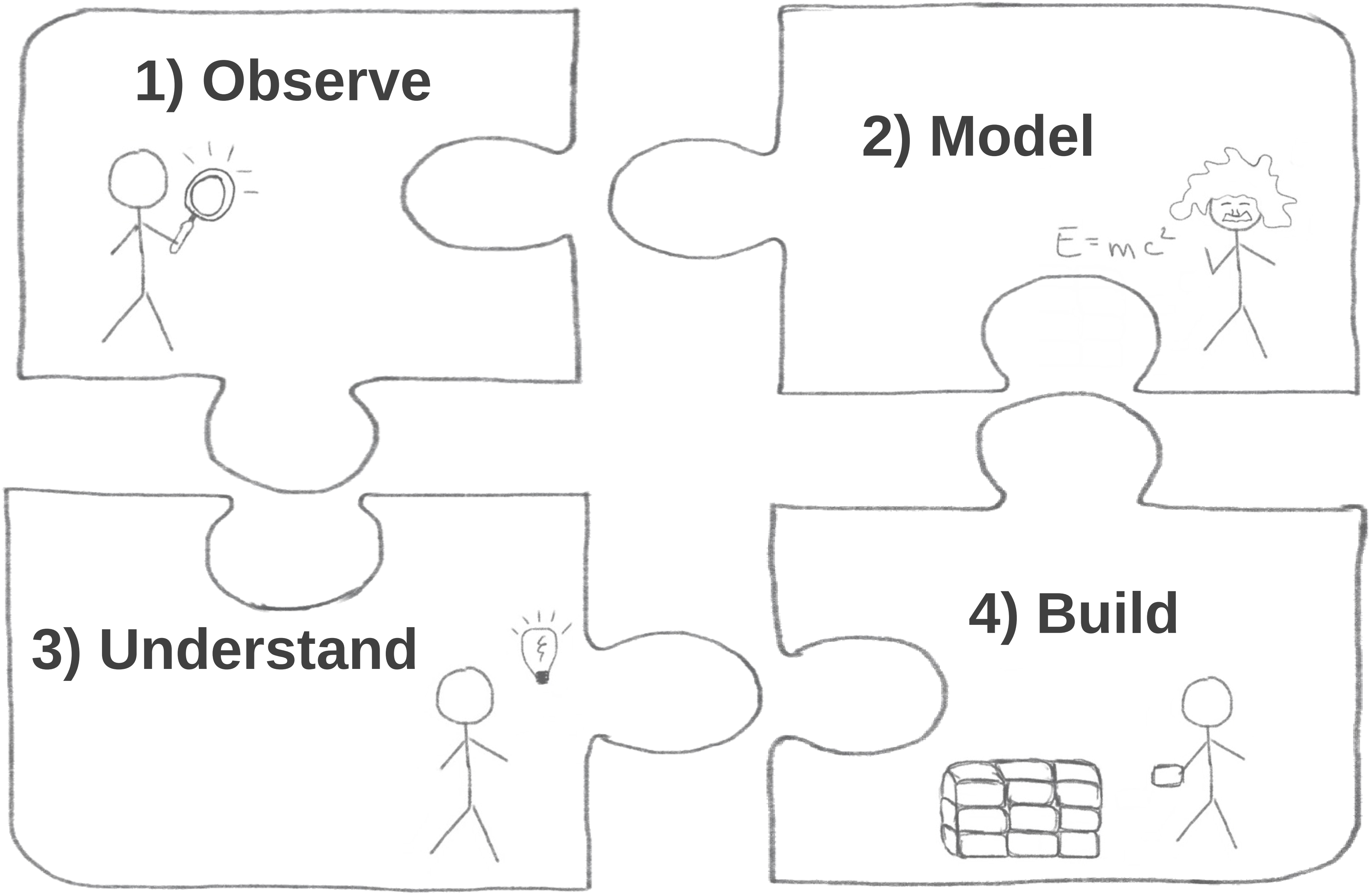
HPC is used to solve complex problems!



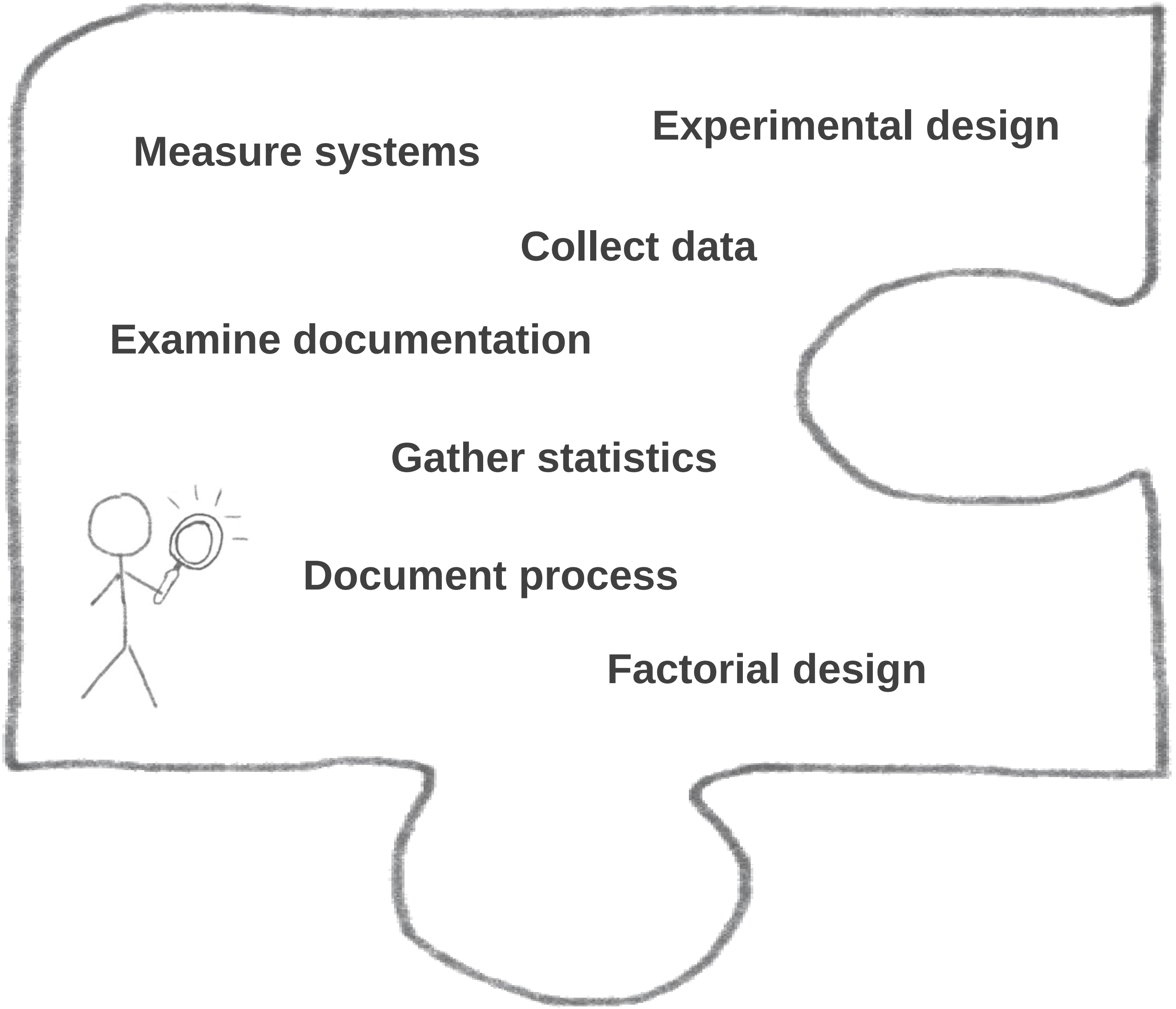
Treat performance-centric programming and system design like physical systems

Image credit: Serena Donnin, Sarah Rauscher, Ivo Kabashow

Scientific Performance Engineering

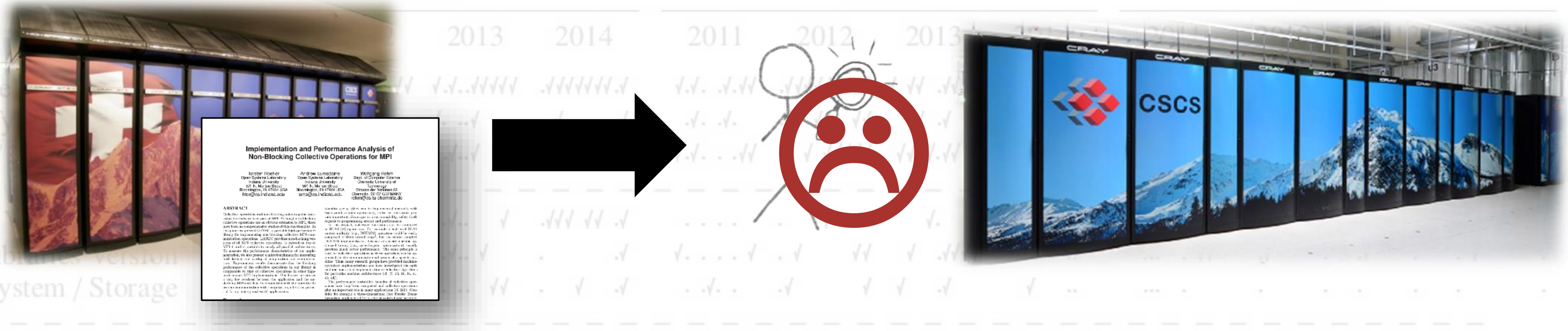


Part I: Observe



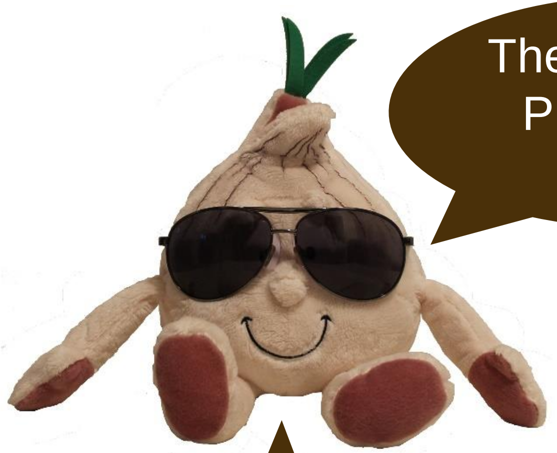
Observe the state of the art in performance measurement

- **Stratified random sample of three top HPC conferences for four years**
  - *HPDC, PPoPP, SC (years: 2011, 2012, 2013, 2014)*
  - *10 random papers from each (10-50% of population)*
  - *120 total papers, 20% (25) did not report performance (were excluded)*



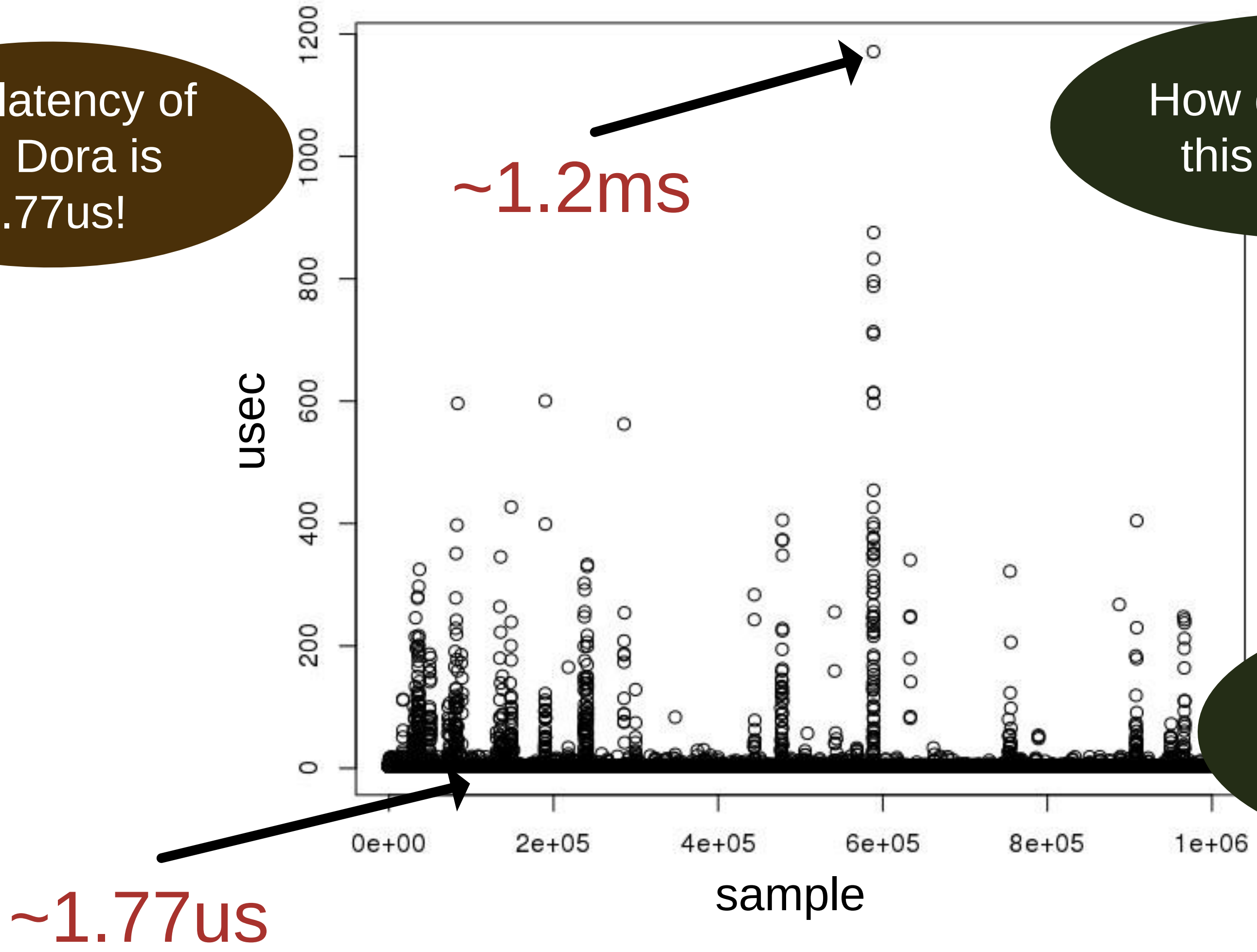
***Performance** results are often nearly **impossible to reproduce**! Thus, we need to provide enough information to allow scientists to understand the experiment, draw own conclusions, assess their certainty, and possibly generalize results.*

Example: Simple ping-pong latency benchmark



The latency of Piz Dora is 1.77us!

I averaged 10^6 runs, it must be right!



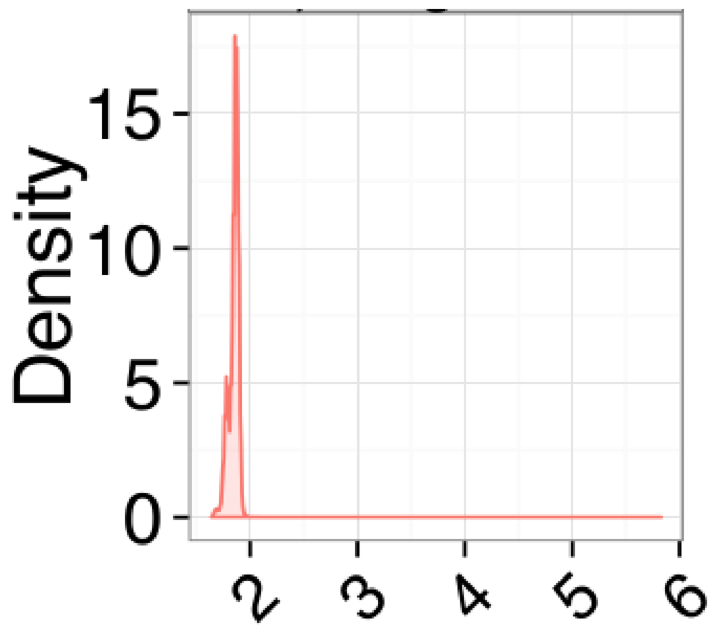
How did you get this number?

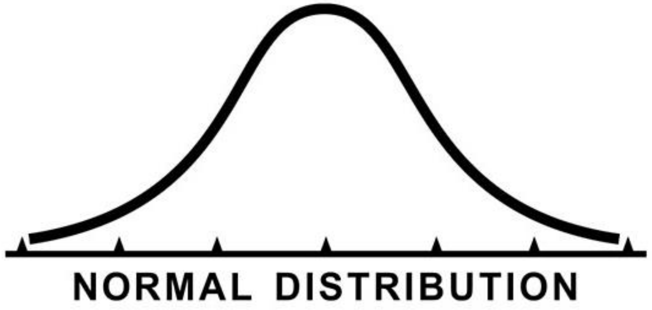
Why do you think so? Can I see the data?

Dealing with variation

The 99.9% confidence interval is 1.765us to 1.775us

Ugs, the data is not normal at all. The **nonparametric** 99.9% CI is much wider: 1.6us to 1.9us!

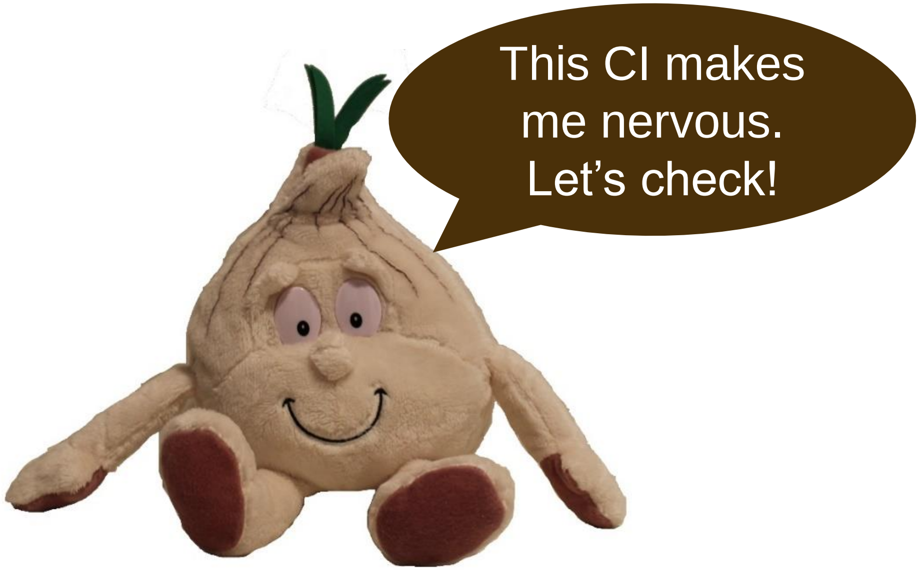




Did you assume normality?

Can we test for normality?

Looking at the data in detail



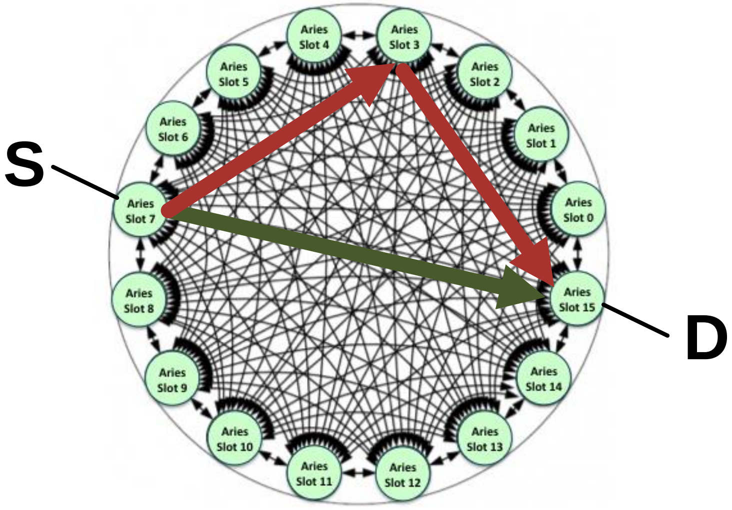
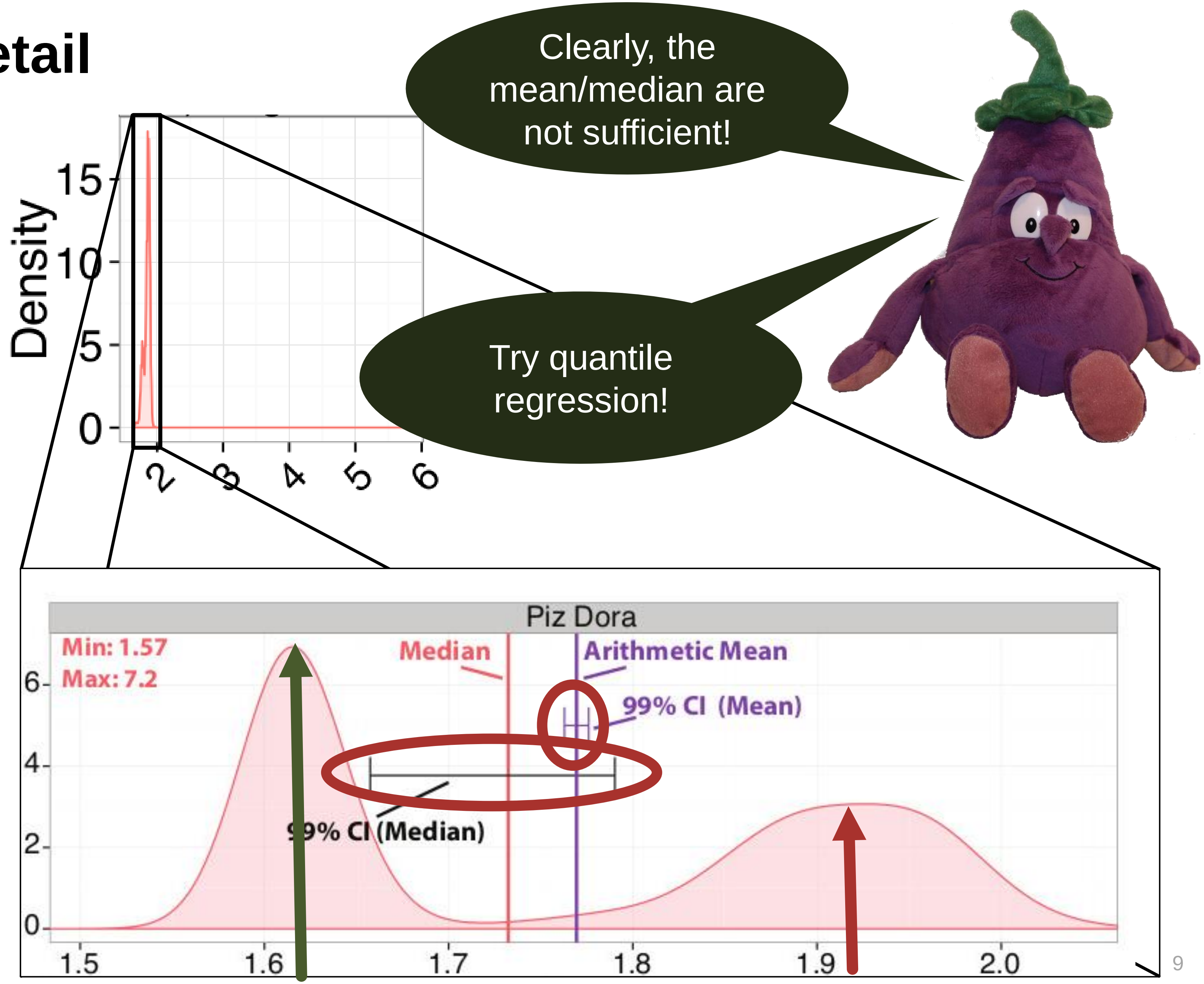


Image credit: nersc.gov



Scientific benchmarking of parallel computing systems

ACM/IEEE Supercomputing 2015 (SC15)

Scientific Benchmarking of Parallel Computing Systems

Twelve ways to tell the masses when reporting performance results

Torsten Hoefler
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
htor@inf.ethz.ch

Roberto Belli
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
bellir@inf.ethz.ch

ABSTRACT

Measuring and reporting performance of parallel computers constitutes the basis for scientific advancement of high-performance computing (HPC). Most scientific reports show performance improvements of new techniques and are thus obliged to ensure reproducibility or at least interpretability. Our investigation of a stratified sample of 120 papers across three top conferences in the field shows that the state of the practice is lacking. For example, it is often unclear if reported improvements are deterministic or observed by chance. In addition to distilling best practices from existing work, we propose statistically sound analysis and reporting techniques and simple guidelines for experimental design in parallel computing and codify them in a portable benchmarking library. We

Reproducing experiments is one of the main principles of the scientific method. It is well known that the performance of a computer program depends on the application, the input, the compiler, the runtime environment, the machine, and the measurement methodology [20,43]. If a single one of these aspects of *experimental design* is not appropriately motivated and described, presented results can hardly be reproduced and may even be misleading or incorrect.

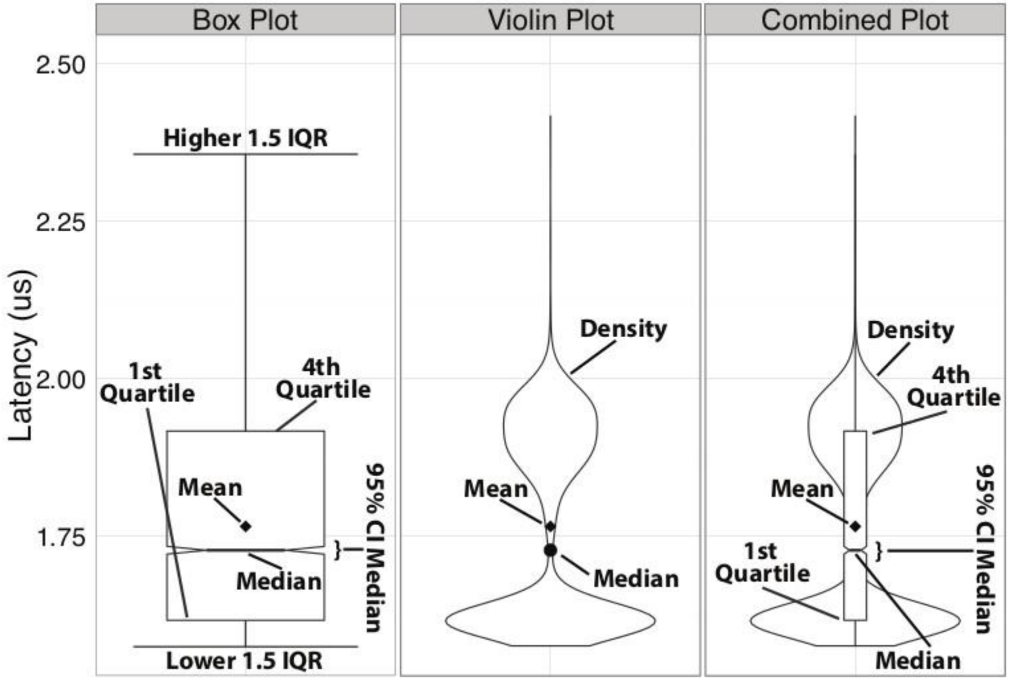
The complexity and uniqueness of many supercomputers makes reproducibility a hard task. For example, it is practically impossible to recreate most hero-runs that utilize the world's largest machines because these machines are often unique and their software configurations changes regularly. We introduce the notion of *interpretability*, which is weaker than reproducibility. *We call an ex-*

erpret the
by lines if
alid.

Simplifying Measuring and Reporting: LibSciBench

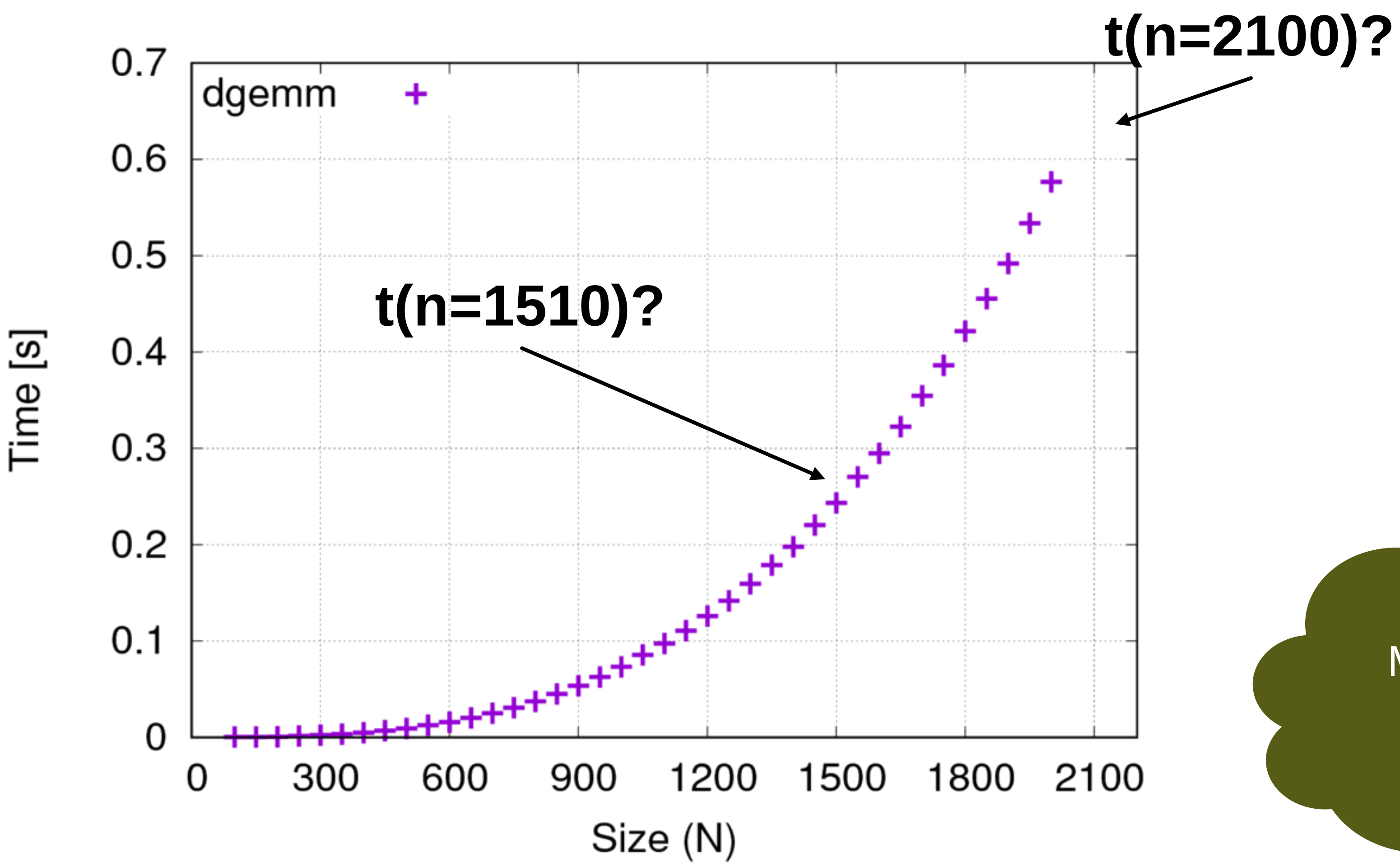
```
#include <mpi.h>
#include <liblsb.h>
#include <stdlib.h>

#define N 1024
#define RUNS 10

int main(int argc, char *argv[]){
    int i, j, rank, buffer[N];

    MPI_Init(&argc, &argv);
    LSB_Init("test_bcast", 0);

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    /* Output the info (i.e., rank, runs) in the results file */
    LSB_Set_Rparam_int("rank", rank);
    LSB_Set_Rparam_int("runs", RUNS);

    for (sz=1; sz<=N; sz*=2){
      for (j=0; j<RUNS; j++){
        /* Reset the counters */
        LSB_Res();

        /* Perform the operation */
        MPI_Bcast(buffer, sz, MPI_INT, 0, MPI_COMM_WORLD);

        /* Register the j-th measurement of size sz */
        LSB_Rec(sz);
      }
    }

    LSB_Finalize();
    MPI_Finalize();
    return 0;
}
```

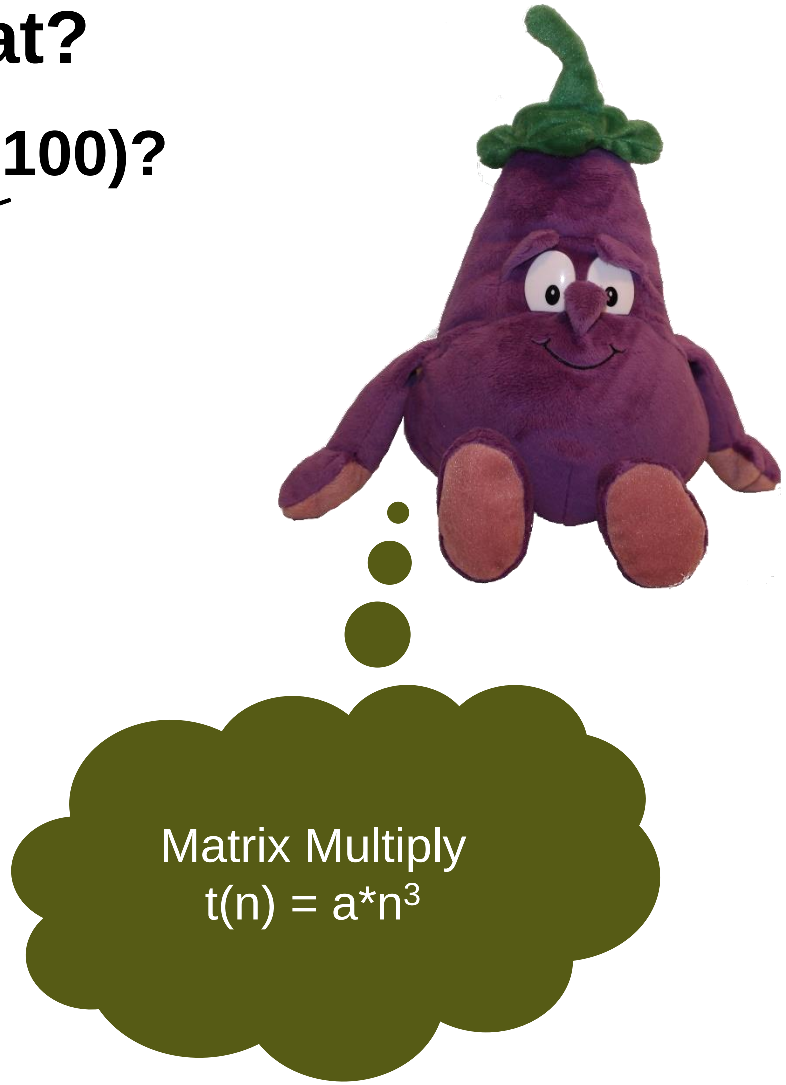
- Simple MPI-like C/C+ interface
- High-resolution timers
- Flexible data collection
- Controlled by environment variables
- Tested up to 512k ranks
- Parallel timer synchronization
- R scripts for data analysis and visualization



S. Di Girolamo, TH: http://spcl.inf.ethz.ch/Research/Performance/LibLSB/

We have the (statistically sound) data, now what?

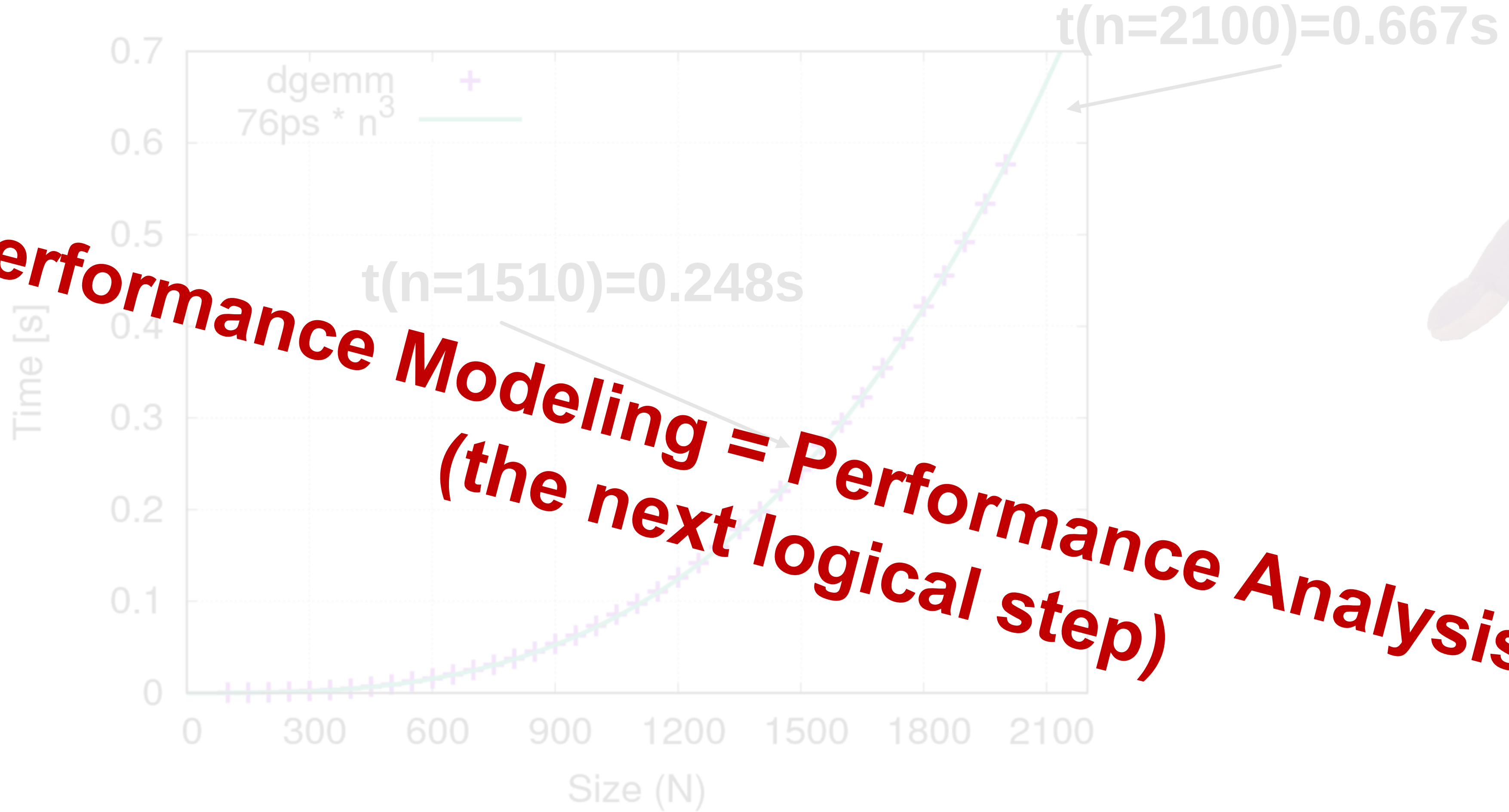




The 99% confidence interval is within 1% of the reported median.

TH, W. Gropp, M. Snir, W. Kramer: Performance Modeling for Systematic Performance Tuning, IEEE/ACM SC11

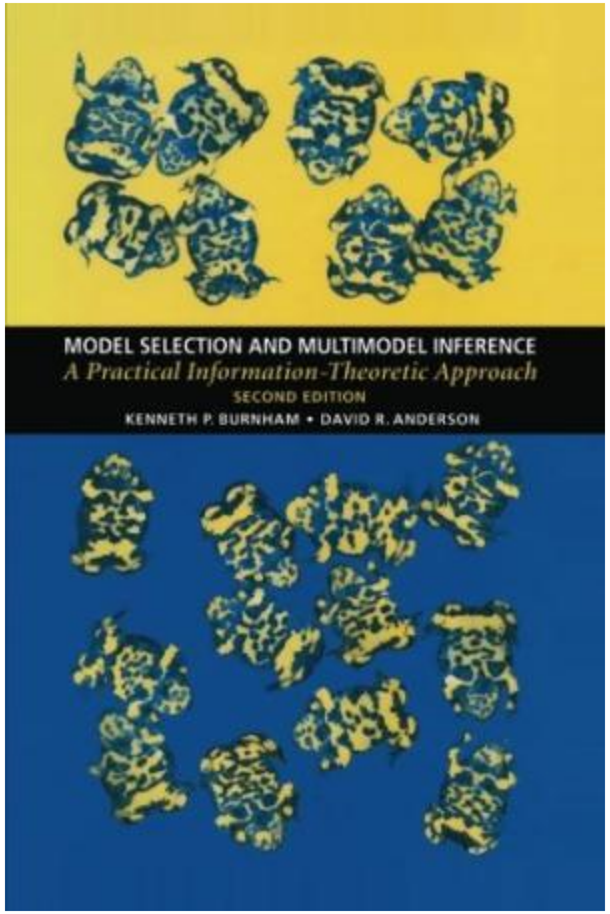
We have the (statistically sound) data, now what?

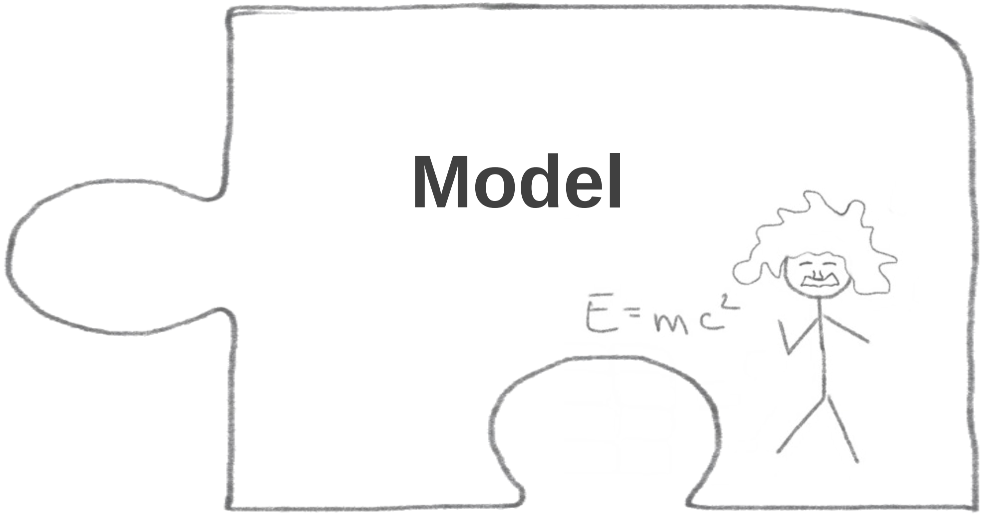


Performance Modeling = Performance Analysis v 2.0
(the next logical step)

The 99% confidence interval is within 1% of the reported median.
The adjusted R^2 of the model fit is 0.99

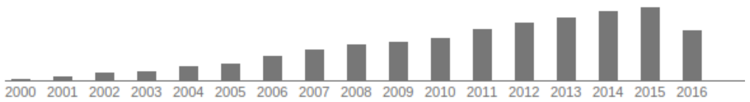
Part II: Model





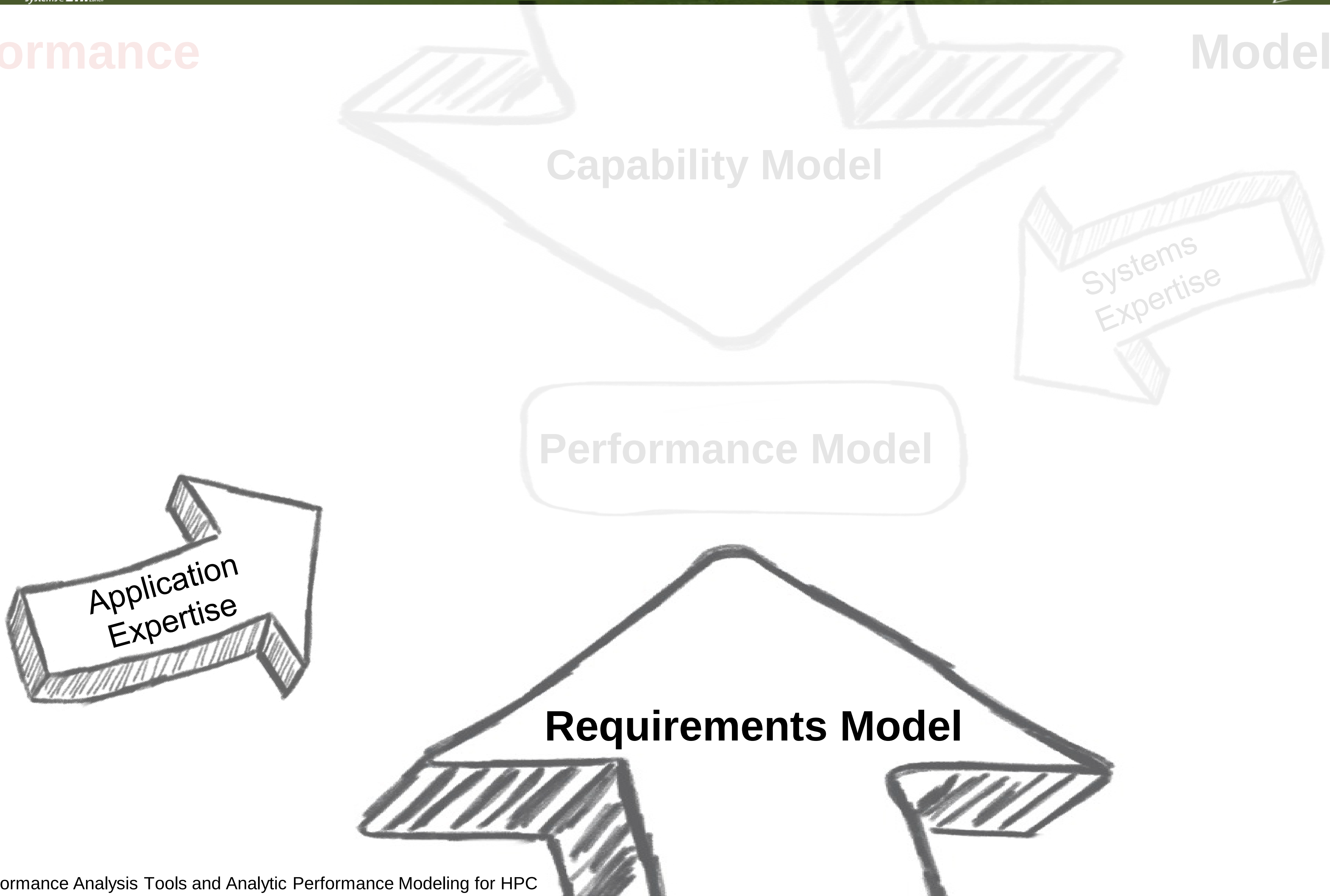
Burnham, Anderson: *"A model is a simplification or approximation of reality and hence will not reflect all of reality. ... Box noted that "all models are wrong, but some are useful." While a model can never be "truth," a model might be ranked from very useful, to useful, to somewhat useful to, finally, essentially useless."*

Cited by 33599



This is generally true for all kinds of modeling.
We focus on **performance modeling** in the following!

Performance Modeling



Requirements modeling I: Six-step performance modeling

Input parameters

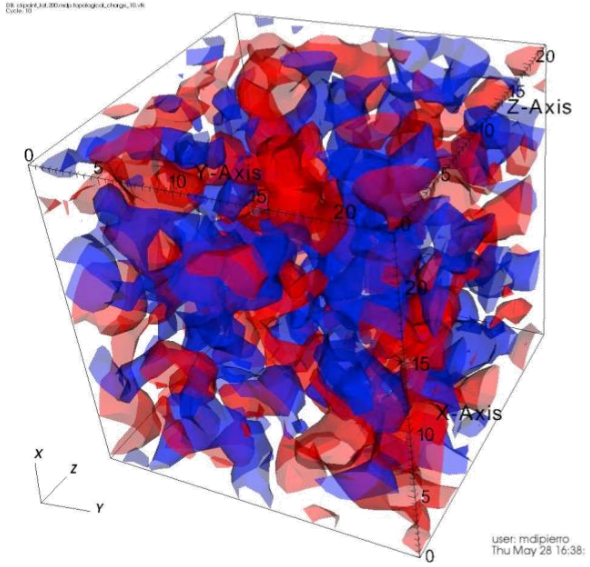
Describe application kernels

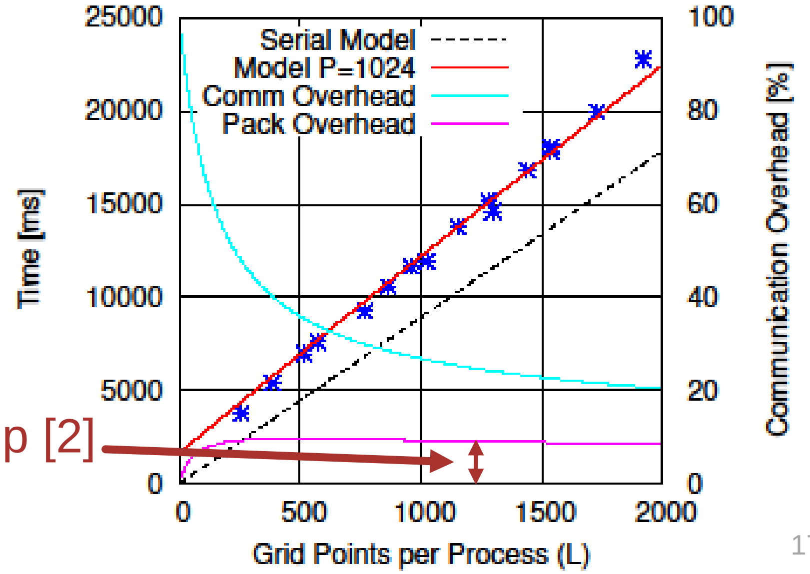
Communication pattern

Communication / computation overlap

Fit sequential baseline

Communication parameters





10-20% speedup [2]

[1] TH, W. Gropp, M. Snir and W. Kramer: Performance Modeling for Systematic Performance Tuning, SC11
[2] TH and S. Gottlieb: Parallel Zero-Copy Algorithms for Fast Fourier Transform and Conjugate Gradient using MPI Datatypes, EuroMPI'10

Requirements modeling II: Automated best-fit modeling

- **Manual kernel selection and hypothesis generation is time consuming (boring and tricky)**
- **Idea: Automatically select best (scalability) model from predefined search space**

Number of processes

number of terms

$$f(p) = \sum_{k=1}^{n} c_k \cdot p^{i_k} \cdot \log_2^{j_k}(p)$$

(model) constant

$n \in \mathbb{N}$
$i_k \in I$
$j_k \in J$
$I, J \in \mathbb{Q}$

$n = 1$
$I = \{0, 1, 2\}$
$J = \{0, 1\}$

c_1	$c_1 \cdot \log(p)$
$c_1 \cdot p$	$c_1 \cdot p \cdot \log(p)$
$c_1 \cdot p^2$	$c_1 \cdot p^2 \cdot \log(p)$

[1]: A. Calotoiu, TH, M. Poke, F. Wolf: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes, IEEE/ACM SC13

Requirements modeling II: Automated best-fit modeling

- **Manual kernel selection and hypothesis generation is time consuming (and boring)**
- **Idea: Automatically select best model from predefined space**

$$f(p) = \sum_{k=1}^{n} c_k \cdot p^{i_k} \cdot \log_2^{j_k}(p)$$

$$n \in \mathbb{N}$$
$$i_k \in I$$
$$j_k \in J$$
$$I, J \subset \mathbb{Q}$$

$$n = 2$$
$$I = \{0, 1, 2\}$$
$$J = \{0, 1\}$$

$c_1 + c_2 \cdot p$

$c_1 + c_2 \cdot p^2$

$c_1 + c_2 \cdot \log(p)$

$c_1 + c_2 \cdot p \cdot \log(p)$

$c_1 + c_2 \cdot p^2 \cdot \log(p)$

$c_1 \cdot \log(p) + c_2 \cdot p$

$c_1 \cdot \log(p) + c_2 \cdot p \cdot \log(p)$

$c_1 \cdot \log(p) + c_2 \cdot p^2$

$c_1 \cdot \log(p) + c_2 \cdot p^2 \cdot \log(p)$

$c_1 \cdot p + c_2 \cdot p \cdot \log(p)$

$c_1 \cdot p + c_2 \cdot p^2$

$c_1 \cdot p + c_2 \cdot p^2 \cdot \log(p)$

$c_1 \cdot p \cdot \log(p) + c_2 \cdot p^2$

$c_1 \cdot p \cdot \log(p) + c_2 \cdot p^2 \cdot \log(p)$

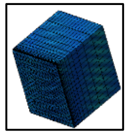
$c_1 \cdot p^2 + c_2 \cdot p^2 \cdot \log(p)$

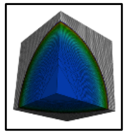
[1]: A. Calotoiu, T. Hoefler, M. Poke, F. Wolf: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes, IEEE/ACM SC13

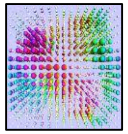
Tool support: Extra-P for automated best-fit modeling [1]

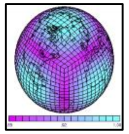
TECHNISCHE UNIVERSITÄT DARMSTADT

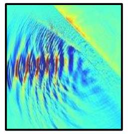


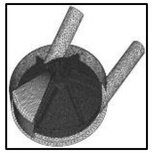

Sweep3d

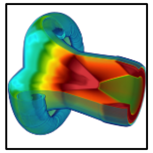

Lulesh


Milc

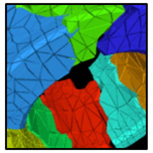

HOMME

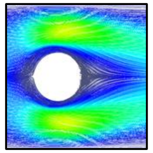

JUSPIC


XNS


BLAST


NEST


UG4


MP2C

Tutorial: **Insighful Automatic Performance Modeling**
A. Calotoiu, F. Wolf, TH, M. Schulz

Sunday, November 13th
1:30pm - 5pm

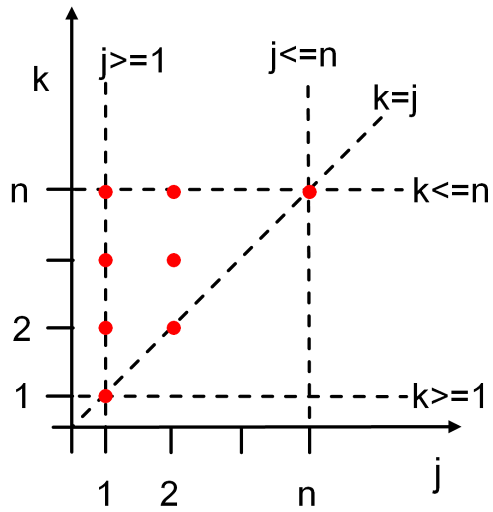
SC16 Salt Lake City, Utah | hpc matters.

[1] Download Extra-P at: http://www.scalasca.org/software/extra-p/download.html
[2] A. Calotoiu, D. Beckingsale, C. W. Earl TH, I. Karlin, M. Schulz, F. Wolf: Fast Multi-Parameter Performance Modeling, IEEE Cluster 2016

Requirements modeling III: Source-code analysis [1]

- **Extra-P selects model based on best fit to the data**
  - What if the data is not sufficient or too noisy?
- **Back to first principles**
  - The source code describes all possible executions
  - Describing all possibilities is too expensive, focus on counting loop iterations symbolically

```
for (j = 1; j <= n; j = j*2)
  for (k = j; k <= n; k = k++)
    OperationInBody(j,k);
```

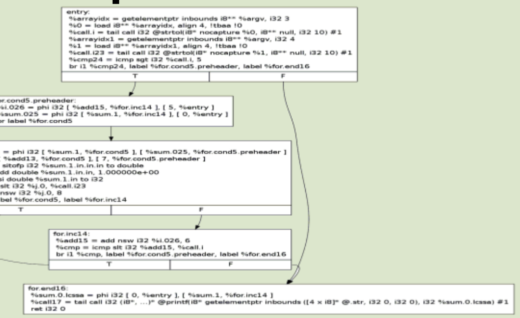


$$N = (n+1)\log_2 n - n + 2$$

Parallel program

```
    do i = 1, procCols
       call mpi_irecv( buff, 2, dp_type, reduce_exch_proc(i),
>                      i, mpi_comm_world, request, ierr )
       call mpi_send( buff2, 2, dp_type, reduce_exch_proc(i),
>                      i, mpi_comm_world, ierr )
       call mpi_wait( request, status, ierr )
    enddo

    do i = id *n/p, ( id +1)* n/p
       do j = 1, nSize
            call compute
```

Loop extraction



Number of iterations

$$N = \sum_{i_1=0}^{n_1(x_{0,1})} \sum_{i_2=0}^{n_2(x_{0,2})} \cdots \sum_{i_{r-1}=0}^{n_{r-1}(x_{0,r-1})} n_r(x_{0,r}).$$

Requirements Models

$$W = N\big|_{p=1}$$

$$D = N\big|_{p\to\infty}$$

[1]: TH, G. Kwasniewski: Automatic Complexity Analysis of Explicitly Parallel Programs, ACM SPAA'14

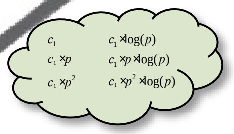
Performance Modeling

Capability Model

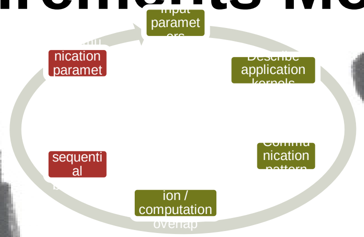
Systems Expertise

Performance Model

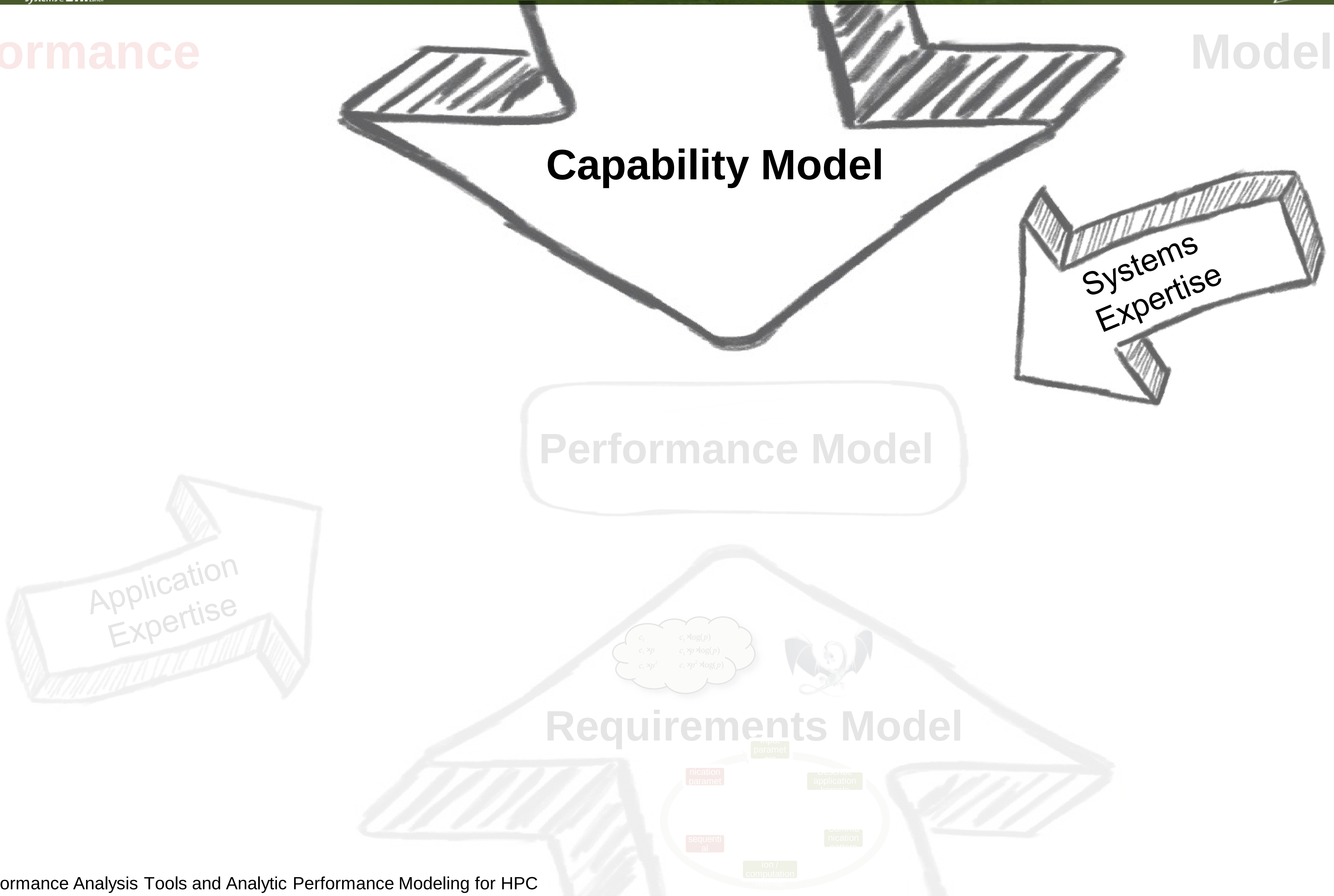
Application Expertise



Requirements Model



Performance Modeling



Capability models for network communication

The LogP model family and the LogGOPS model [1]


A new parallel machine model reflects the critical technology trends underlying parallel computers

A PRACTICAL MODEL of PARALLEL COMPUTATION

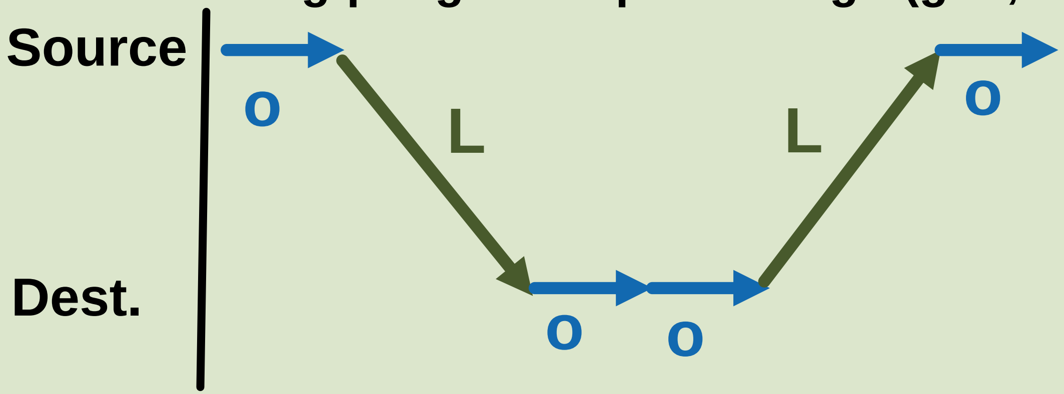
LogP

UR GOAL IS TO DEVELOP A MODEL OF PARALLEL COMPUTATION THAT WILL serve as a basis for the design and analysis of fast, portable parallel algorithms, such as algorithms that can be implemented effectively on a wide variety of current and future parallel machines. If we look at the body of parallel algorithms developed under current parallel models, many are impractical because they exploit artificial factors not present in any rea-

PRAM consists of a collection of processors which compute synchronously in parallel and communicate with a global random access

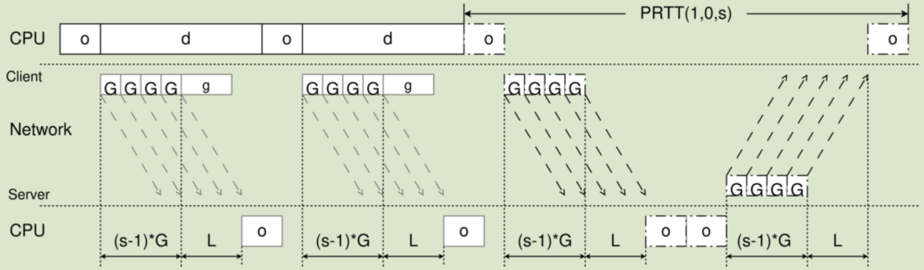
David E. Culler, Richard M. Karp, David Patterson, Abhijit Sahay, Eunice E. Santos, Klaus Erik Schauser, Ramesh Subramonian, and Thorsten von Eicken

Ping-pong in simplified LogP (g<o, P=2)



Finding LogGOPS parameters

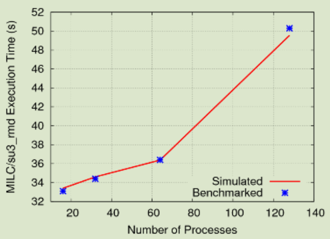
Netgauge [2], model from first principles, fit to data using special kernels

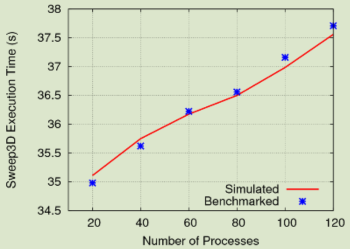


Large scale LogGOPS Simulation

LogGOPSim [1], simulates LogGOPS with 10 million MPI ranks

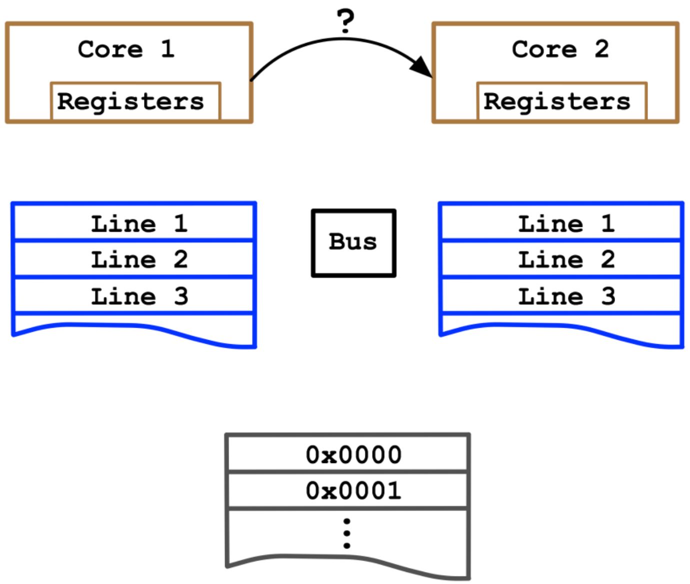
<5% error



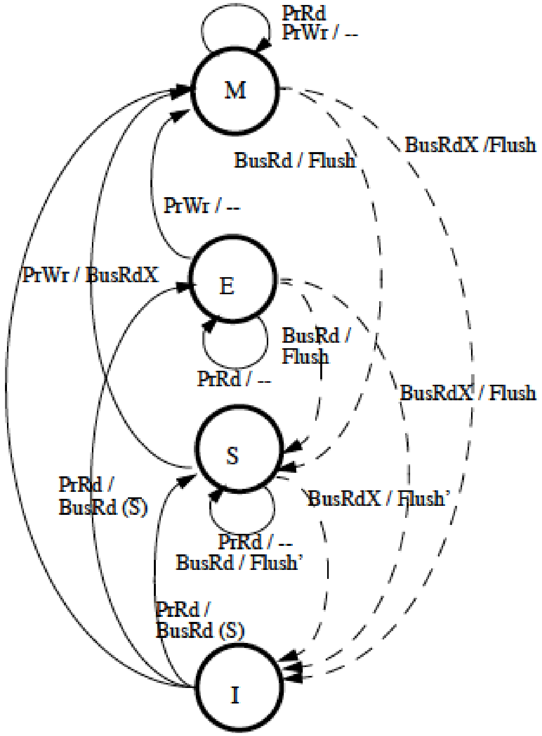


[1]: TH, T. Schneider and A. Lumsdaine: LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model, LSAP 2010, https://spcl.inf.ethz.ch/Research/Performance/LogGOPSim/
[2]: TH, T. Mehlan, A. Lumsdaine and W. Rehm: Netgauge: A Network Performance Measurement Framework, HPCC 2007, https://spcl.inf.ethz.ch/Research/Performance/Netgauge/

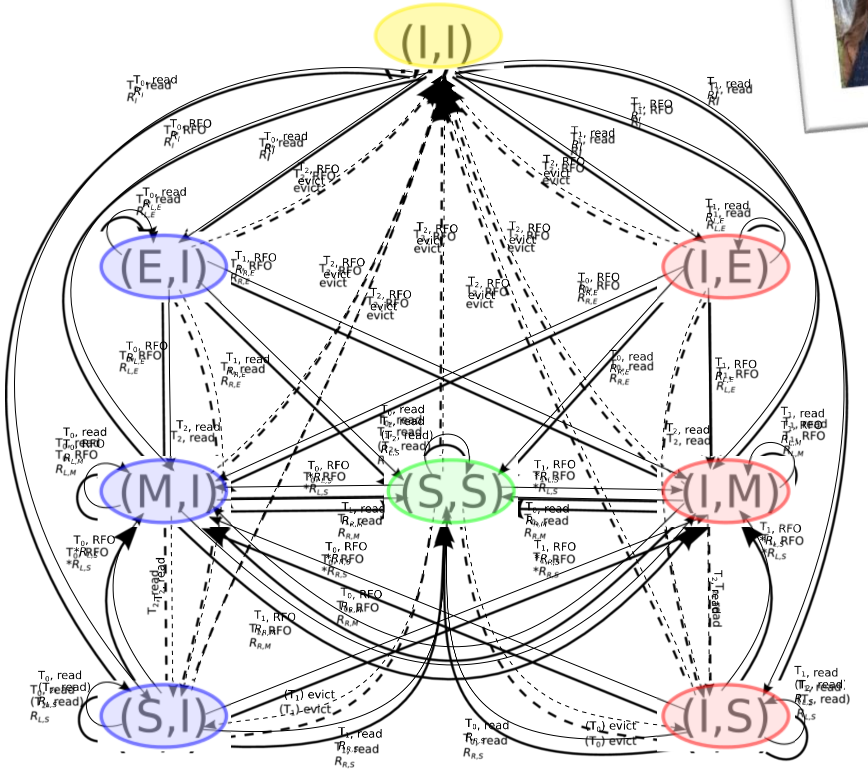
Capability models for cache-to-cache communication

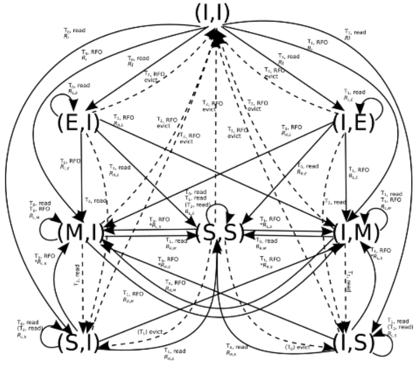


X

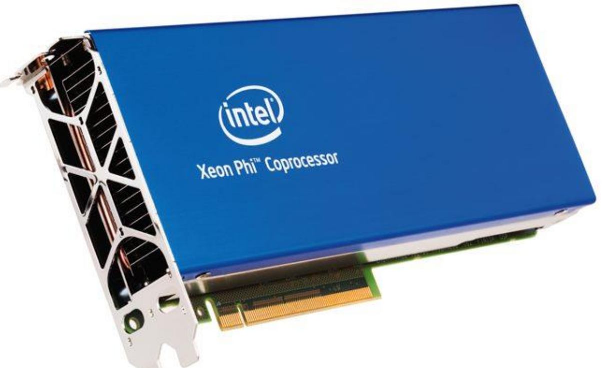


=

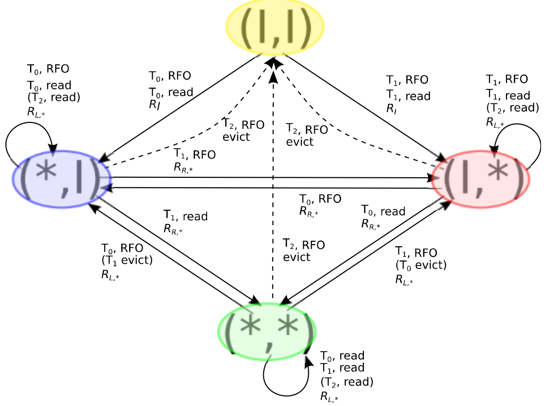




|



=



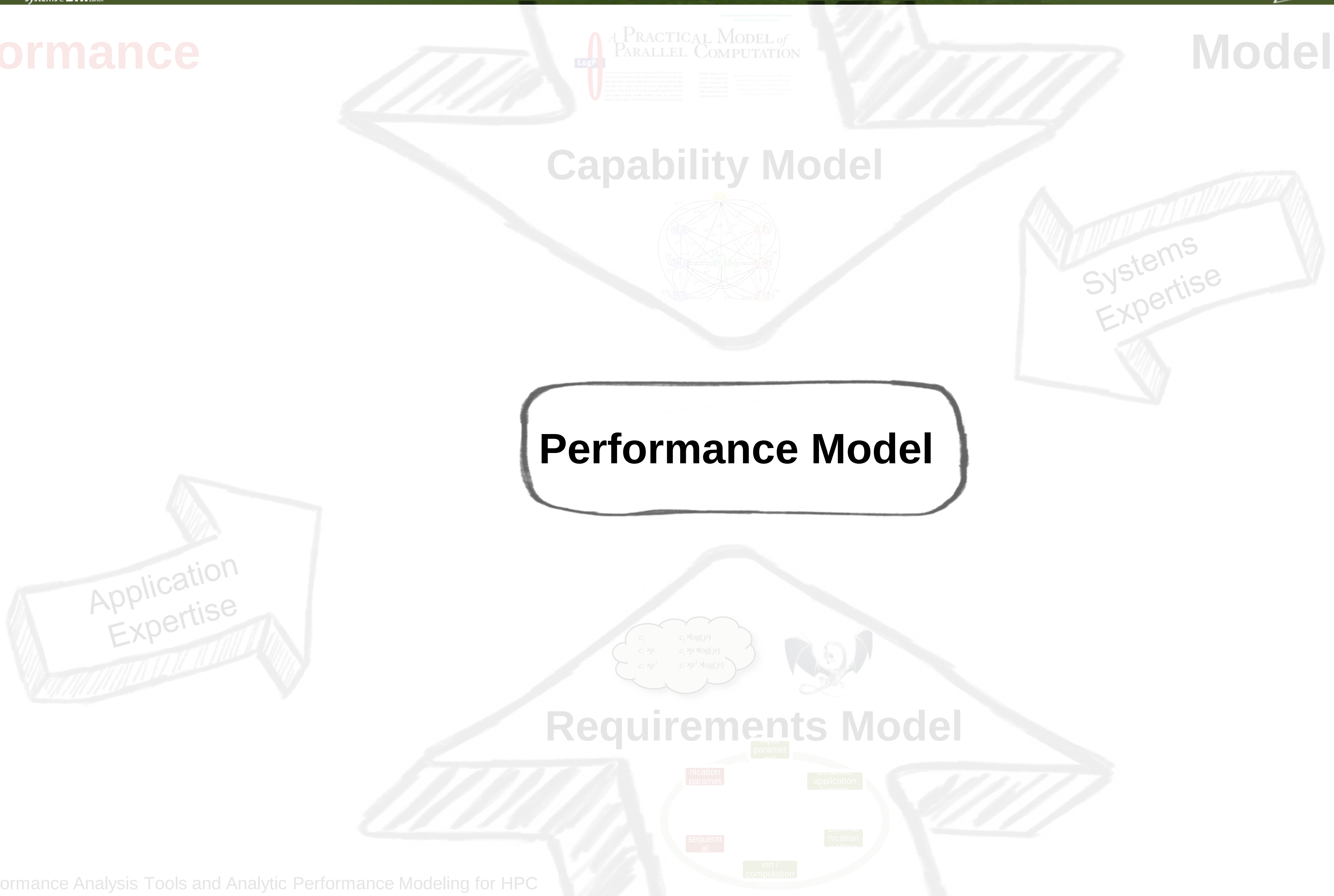
Invalid read R_I = 278 ns

Local read: R_L= 8.6 ns

Remote read R_R = 235 ns

S. Ramos, TH: *"Modeling Communication in Cache-Coherent SMP Systems - A Case-Study with Xeon Phi ", ACM HPDC'13*

Performance Modeling



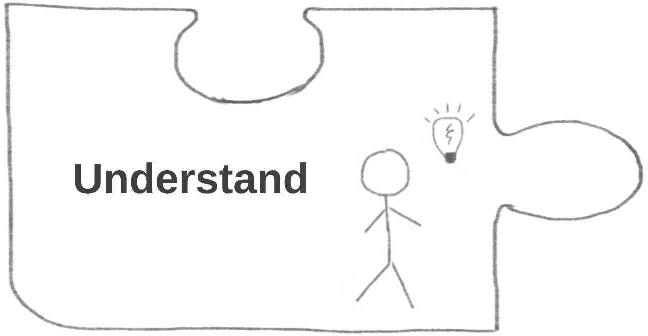
Part III: Understand

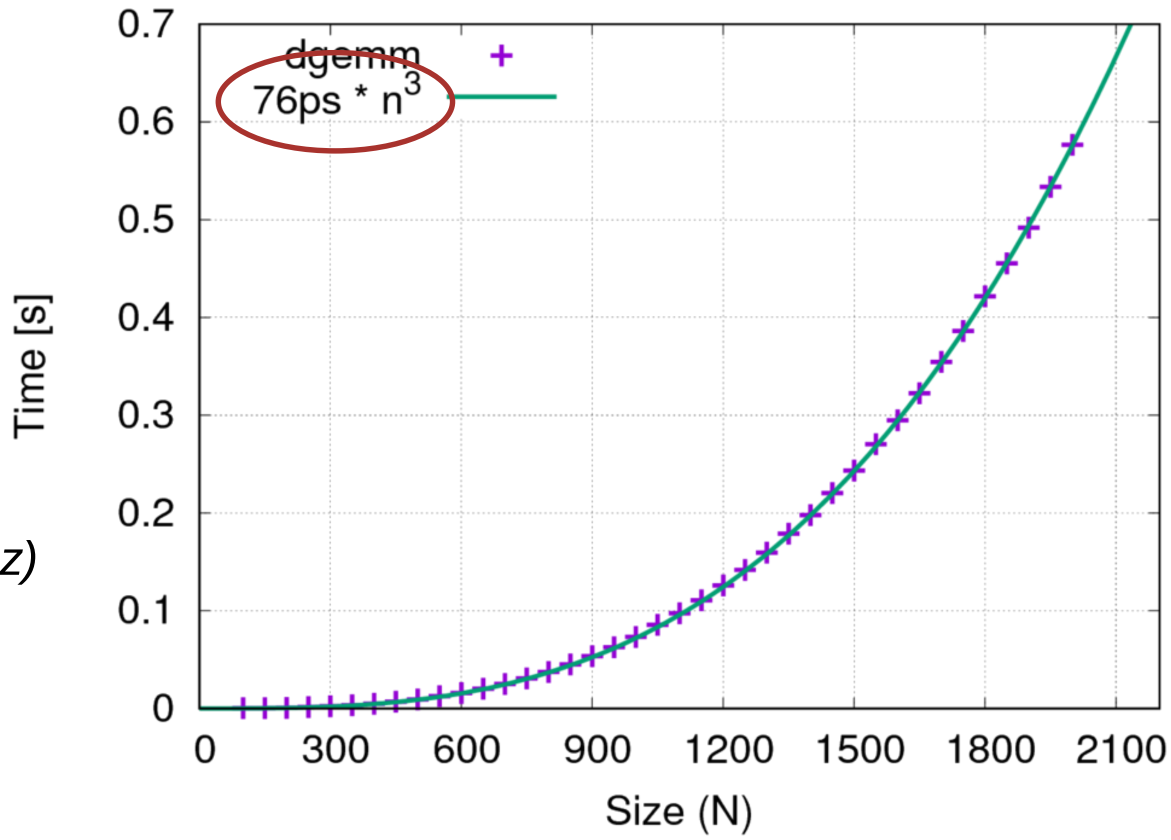
- **Use models to**
  1. Proof optimality of real implementations
     - *Stop optimizing, step back to algorithm level*
  2. Design optimal algorithms or systems in the model
     - *Can lead to non-intuitive designs*

- **Proof optimality of matrix multiplication**
  - Intuition: flop rate is the bottleneck
  - $t(n) = 76ps * n^3$
  - **Flop rate:** $R = 2flop * n^3/(76ps * n^3)$ = **27.78 Gflop/s**
  - **Flop peak**: 3.864 GHz * 8 flops = **30.912 Gflop/s**

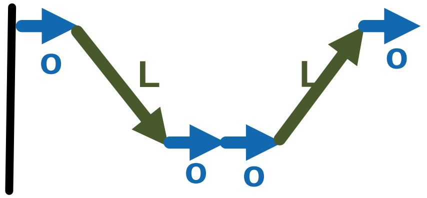
    Achieved ~90% of peak (IBM Power 7 IH @3.864GHz)

- **Gets more complex quickly**
  - Imagine sparse matrix-vector

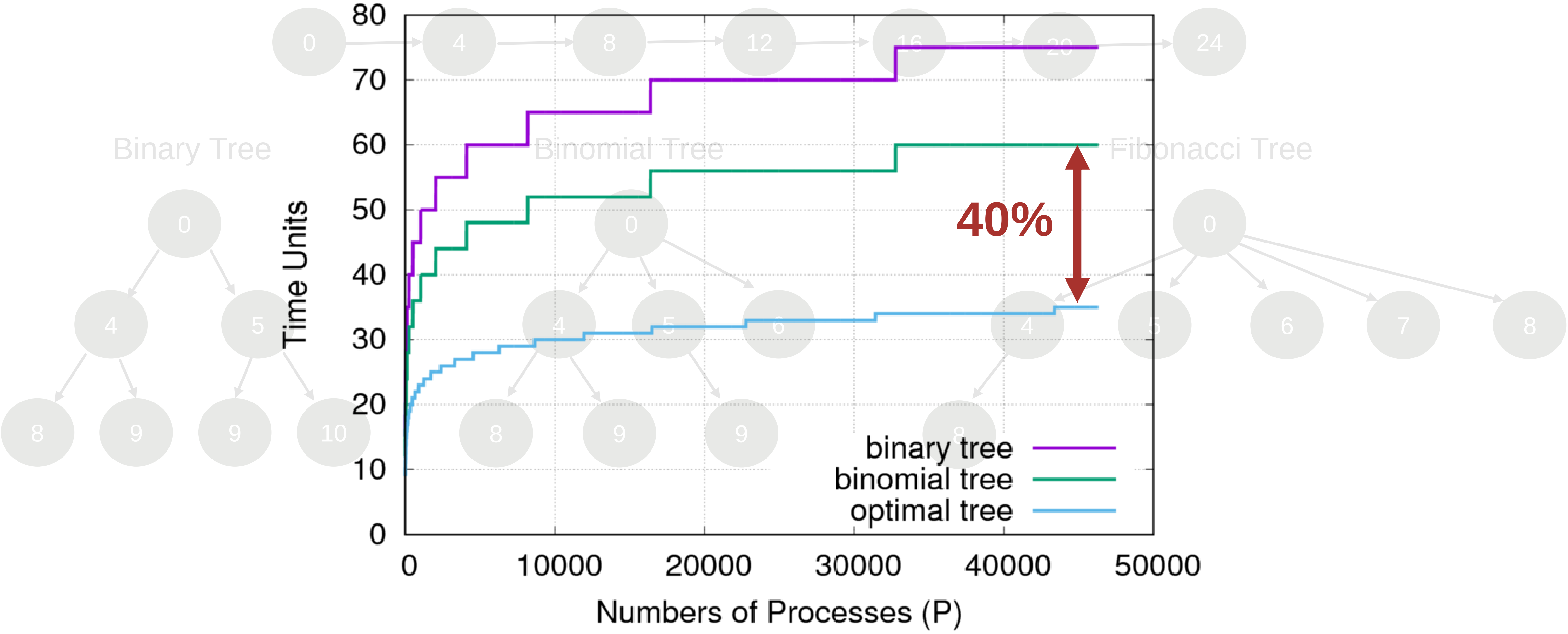




2) Design optimal algorithms – small broadcast in LogP

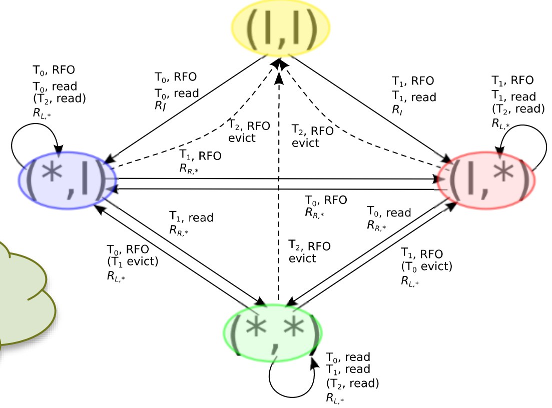


L=2, o=1, P=7

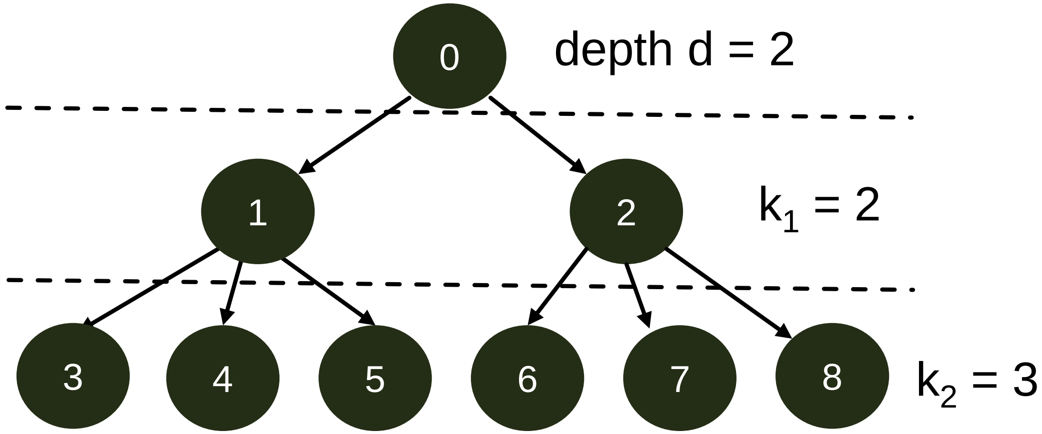


TH, D. Moor: Energy, Memory, and Runtime Tradeoffs for Implementing Collective Communication Operations, JSFI 2015

Design algorithms – bcast in cache-to-cache model



Multi-ary tree example



Tree depth

Level size

$$\mathcal{T}_{tree} = \sum_{i=1}^{d} \mathcal{T}_C(k_i) = \sum_{i=1}^{d} (c \cdot k_i + b)$$

Tree cost

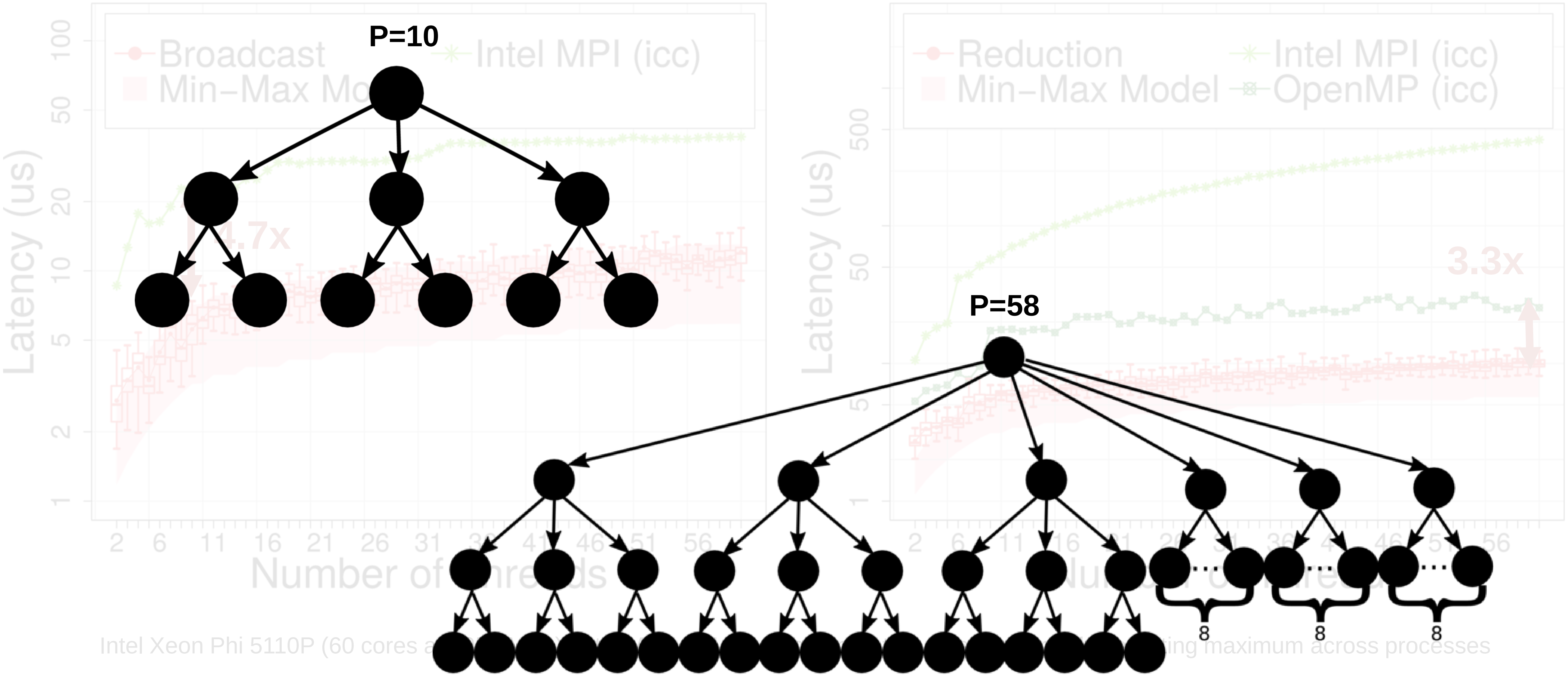
$$= \sum_{i=1}^{d} (R_R + R_L + c \cdot (k_i - 1))$$

$$\mathcal{T}_{sbcast} = \min_{d,k_i} \left(\mathcal{T}_{fw} + \sum_{i=1}^{d} (c \cdot k_i + b) + \sum_{i=1}^{d} \mathcal{T}_{nb}(k_i + 1) \right)$$

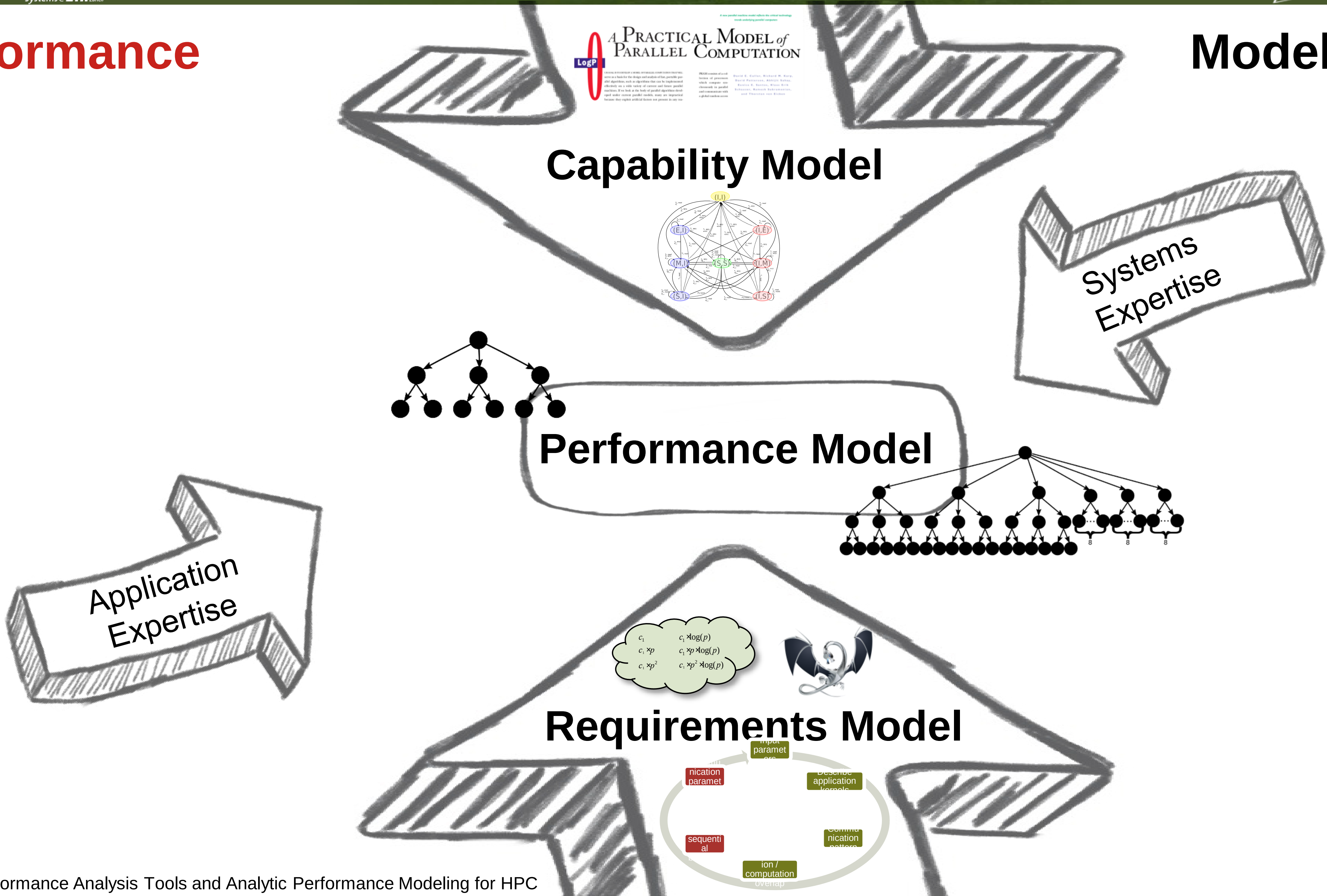
Reached threads

$$N \leq 1 + \sum_{i=1}^{d} \prod_{j=1}^{i} k_j, \quad \forall i < j, k_i \leq k_j$$

Measured results – small broadcast and reduction



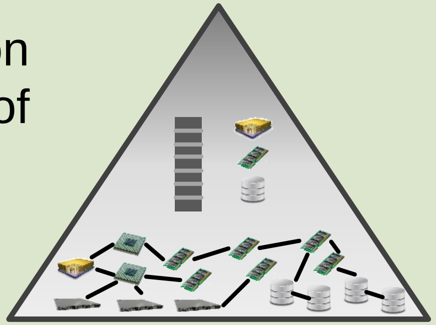
Performance Modeling

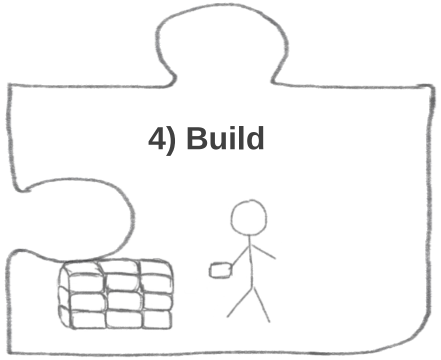


Part IV: Build

Abstraction is Key

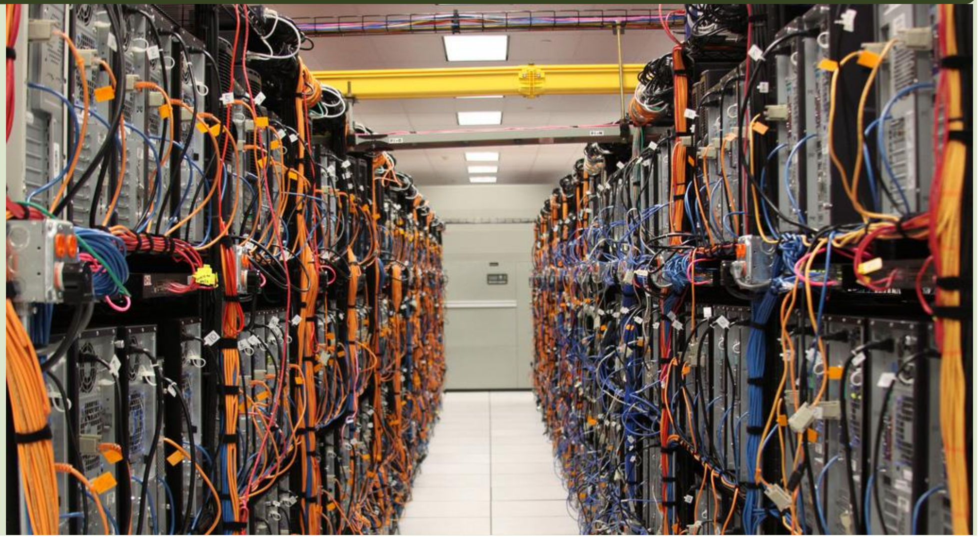
- Enables to focus on essential aspects of a system





Case study: Network Topologies

- **Observe**: optimize for cost, maintain performance:
  - router radix, number of cables, number of routers → cost
  - number of endpoints, latency, global bandwidth → capabilities
- **Model**: system as graph
- **Understand**: degree-diameter graphs
- **Build**: Slim Fly topology
- Result: non-trivial topology that outperforms all existing ones



M. Besta, TH: Slim Fly: A Cost Effective Low-Diameter Network Topology, ACM/IEEE Supercomputing 2014, SC14

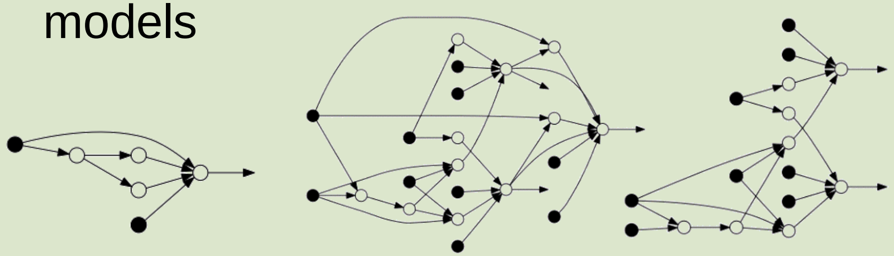
How to continue from here?

Transformation System

- User-supported, compile- and run-time

Parallel Language

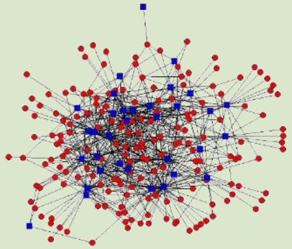
- Data-centric, explicit requirements models



DAPP

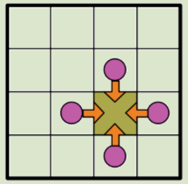
European Research Council
Established by the European Commission
Supporting top researchers from anywhere in the world

Performance-transparent Platforms

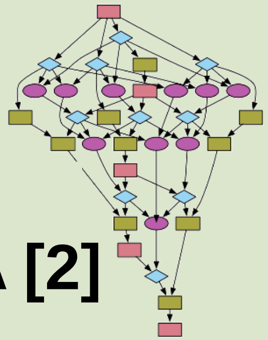


HTM [1]

MPI RMA



foMPI-NA [2]



NISA [3]

portals

[1]: M. Besta, TH: Accelerating Irregular Computations with Hardware Transactional Memory and Active Messages, ACM HPDC'15
[2]: R. Belli, TH: Notified Access: Extending Remote Memory Access Programming Models for Producer-Consumer Synchronization, IPDPS'15
[3]: S. Di Girolamo, P. Jolivet, K. D. Underwood, TH: Exploiting Offload Enabled Network Interfaces, IEEE Micro'16

Backup