Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Performance Modeling for Parallel Software Development and Tuning

Torsten Hoefler

Motivation

- **David Patterson**: *"A portable parallel program is an oxymoron"*
  - Certainly true, isn't it?
  - And it gets worse … why?

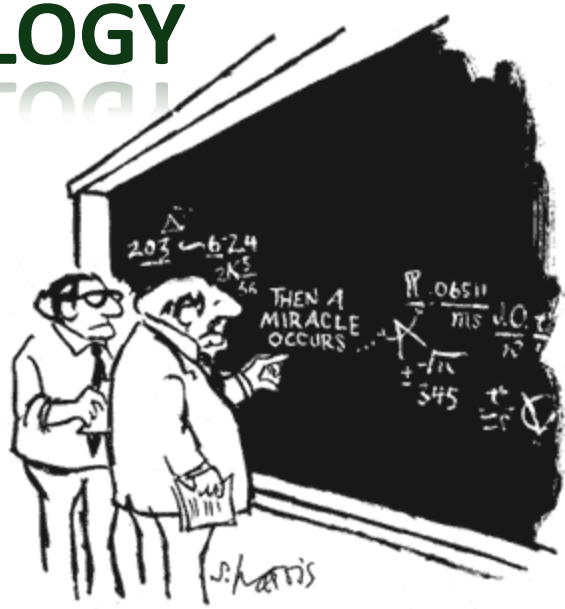
- Performance is complex
  - In parallel even more …
  - We need tools to understand and generalize!
- Even correctness is complex
  - Mainly fixed by models of computation/invariants
- We propose performance modeling for a change!

Performance Modeling

- Representing performance with **analytic** expressions
  - Not just series of points from benchmarks
  - **Algebraic** derivation to find sweet-spots
- Why performance modeling?
  - Extrapolation (scalability)
  - **Insight** into requirements
    - Message sizes, HW/SW Co-Design
  - Purchasing decisions based on models

- BUT: It's mostly used by computer scientists!
  - Our goal: enable application developers and domain scientists to use performance modeling

Our Simple Methodology



"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

- Combine **analytical** methods and **empirical** performance measurement tools
  - Programmer specifies expectation
    - E.g., $T = a+b*N^3$
  - Tools find the parameters
    - Empirically, e.g., least squares
  - *We derive the scaling analytically and fill in the constants with empirical measurements*
- Models must be as **simple** and **effective** as possible
  - Simplicity increases the insight
  - *Precision needs to be just good enough to drive action*

Details: Hoefler et al.: "Performance Modeling for Systematic Performance Tuning.", SC11, SotP

Other Philosophies

- Simulation:
  - Very accurate prediction, little insight
- Traditional Performance Modeling (PM):
  - Focuses on accurate predictions
  - Tool for computer scientists, not application developers

Performance Engineering

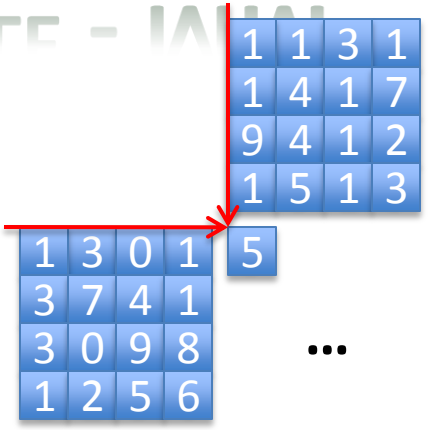
When and where should it be used?

- During the whole software development cycle
  - Analysis (pick the right algorithms)
  - Design (pick the right design pattern)
  - Implementation (choose implementation options)
  - Testing (test if performance expectations are met)
  - Maintenance (monitor performance)
- **Performance bugs** can be as serious and as **expensive** as correctness bugs!

Software Development - Example - MM

- Matrix multiplication (N^3 algorithm)

```
for(int i=0; i<N; ++i)
  for(int j=0; j<N; ++j)
    for(int k=0; k<N; ++k)
      C[i+j*N] += A[i+k*N] * B[k+j*N];
```

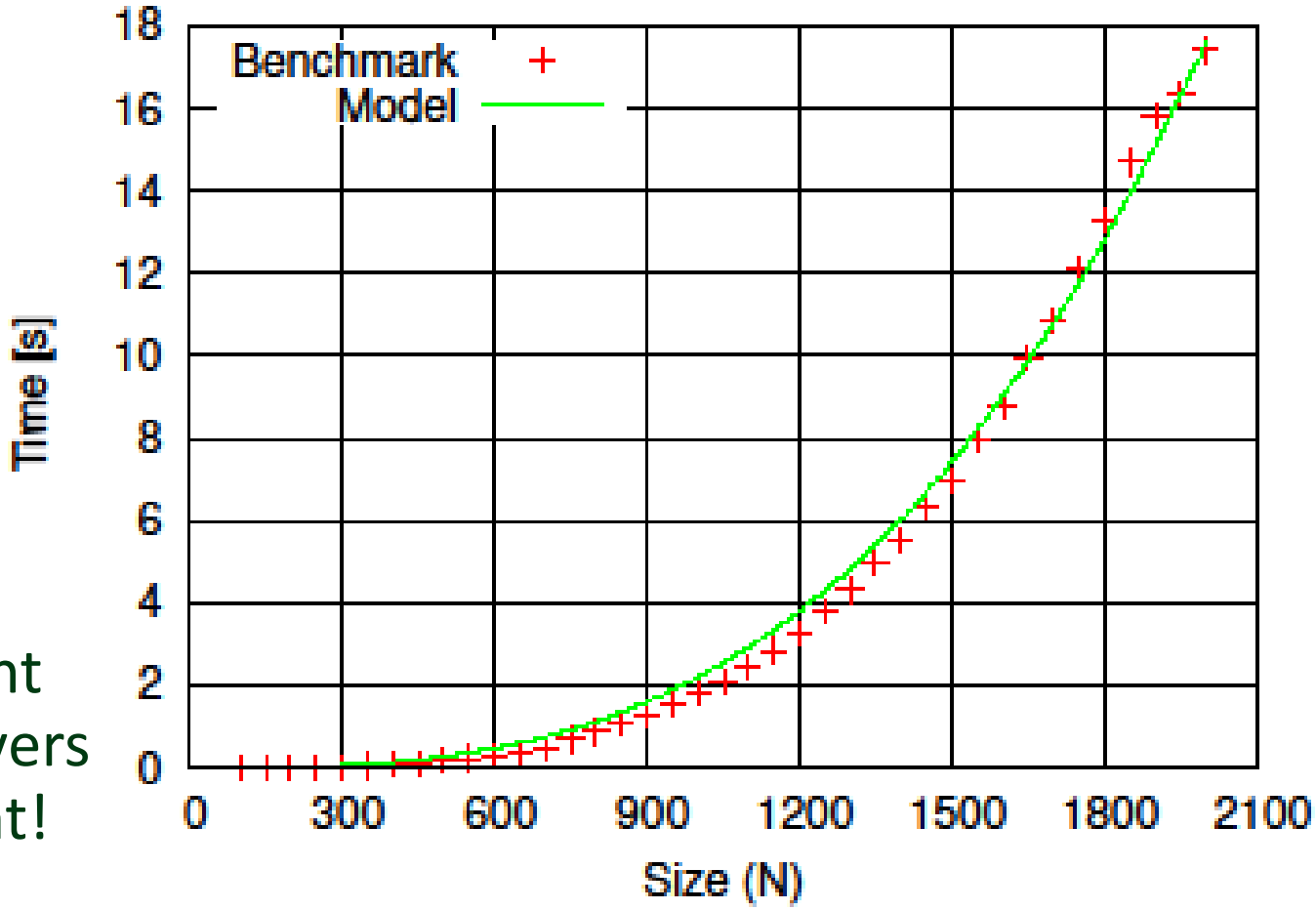


- Trivial (non-blocked) algorithm

- Analytic Model:
  - N^3 FP add/mult, $4N^3$ FP load/store, +int ops
  - How can we get to an execution time? → very hard!

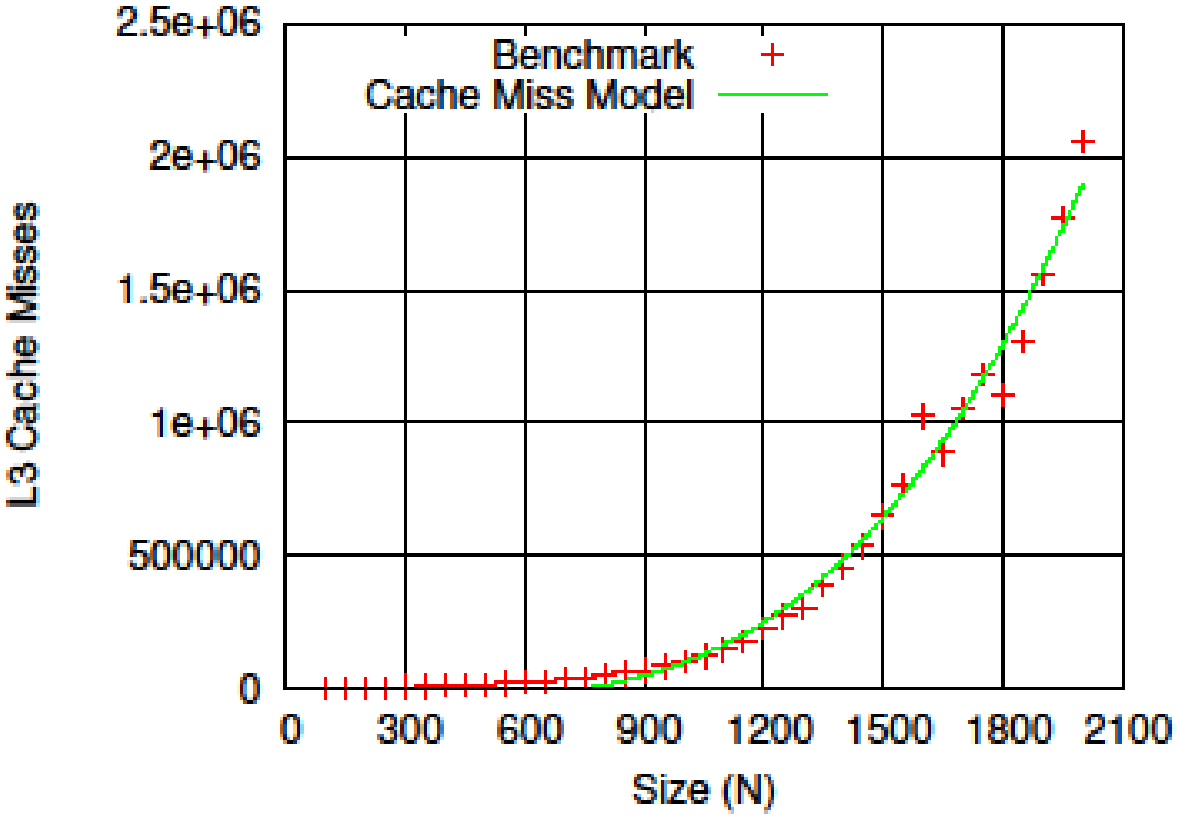
Semi-Empiric Model for MM

- $T(N) = tN^3$
- POWER7
  - t=2.2ns
  - 0.8% err
- Is that all?
  - Requirement Model delivers more insight!



Requirements Model for MM

- Required floating point operations: $2N^3$ (verified)
- Cache misses?
  - Semi-analytic!
  - $C(N) = aN^3 - bN^2$
- POWER7
  - a=3.8e-4
  - a=2.7e-1



More uses of Models

- Performance Optimization
  - Identify bottlenecks and problems during porting
- System Design
  - Co-design based on application requirements
- System Deployment and Testing
  - Know what to expect, find performance issues quickly
- During System Operation
  - Detect silent (and slow) performance degradation

Our Process for Existing Codes

- Simple 6-step process:

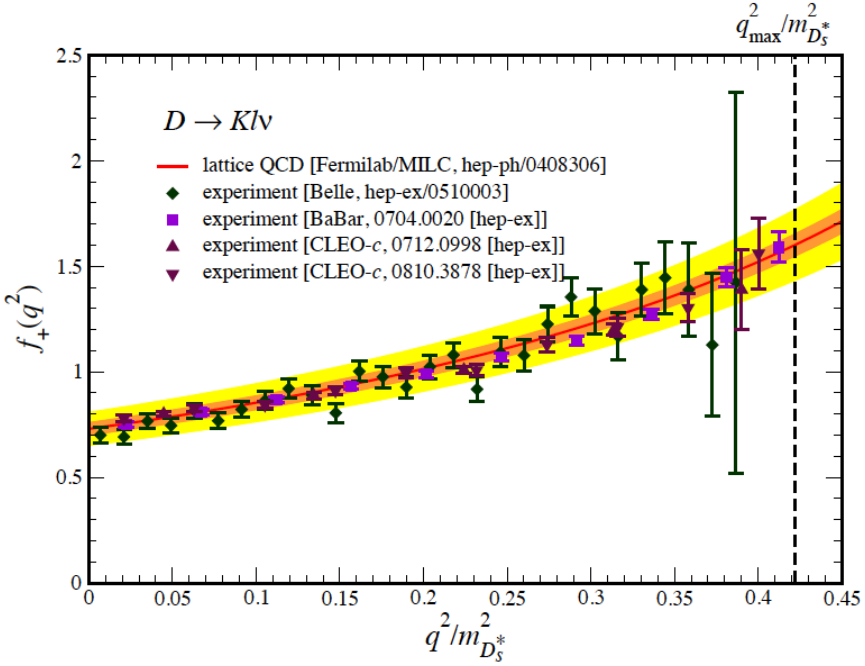
- Analytical steps (domain expert or source-code)
  1) identify **input parameters** that influence runtime
  2) identify most time-intensive **kernels**
  3) determine **communication pattern**
  4) determine **communication/computation overlap**

- Empirical steps (benchmarks/performance tools)
  1) determine **sequential baseline**
  2) determine **communication parameters**

Details: Hoefler et al.: "Performance Modeling for Systematic Performance Tuning.", SC11, SotP

All Steps By Example – MILC

- MIMD Lattice Computation
  - Gains deeper insights in fundamental laws of physics
  - Determine the predictions of lattice field theories (QCD & Beyond Standard Model)
  - Major NSF application
- Challenge:
  - High accuracy (computationally intensive) required for comparison with results from experimental programs in high energy & nuclear physics



Bernard, Gottlieb et al.: Studying Quarks and Gluons On Mimd Parallel Computers

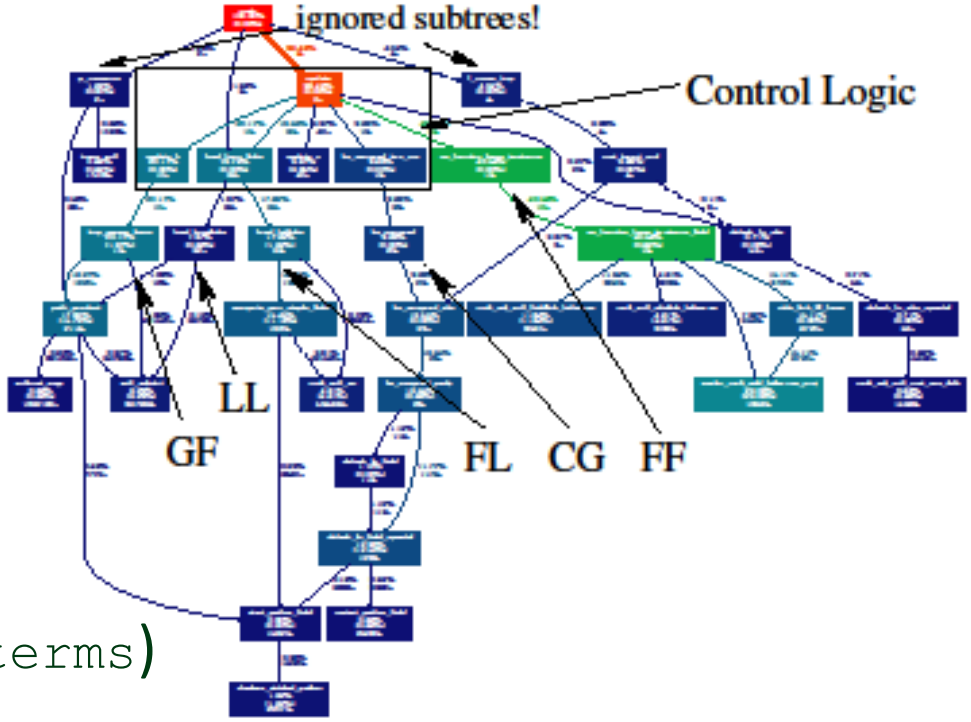
Step 1: Critical Parameters

- Best way: ask a domain expert!
  - Or: look through the code/input file format
- For MILC (thanks to S. Gottlieb):

Name	Description
P	number of PEs (intrinsic parameter)
nx, ny, nz, nt	size in x, y, z, t dimension
warms, trajecs	warmup rounds and trajectories (outer loop)
traj_between_meas	measurement "frequency"
steps_per_trajectory	number of "steps" in each trajectory
beta, mass1, ...	physics parameters that influence CG iterations
max_cg_iterations	limits the conjugate gradient iterations

Step 2: Find Kernels

- E.g., investigate call-tree or source-code
- Control logic
  - `update`
- MILC's kernels:
  - LL (`load_longlinks`)
  - FL (`load_fatlinks`)
  - CG (`ks_congrad`)
  - GF (`imp_gauge_force`)
  - FF (`eo_fermion_force_twoterms`)

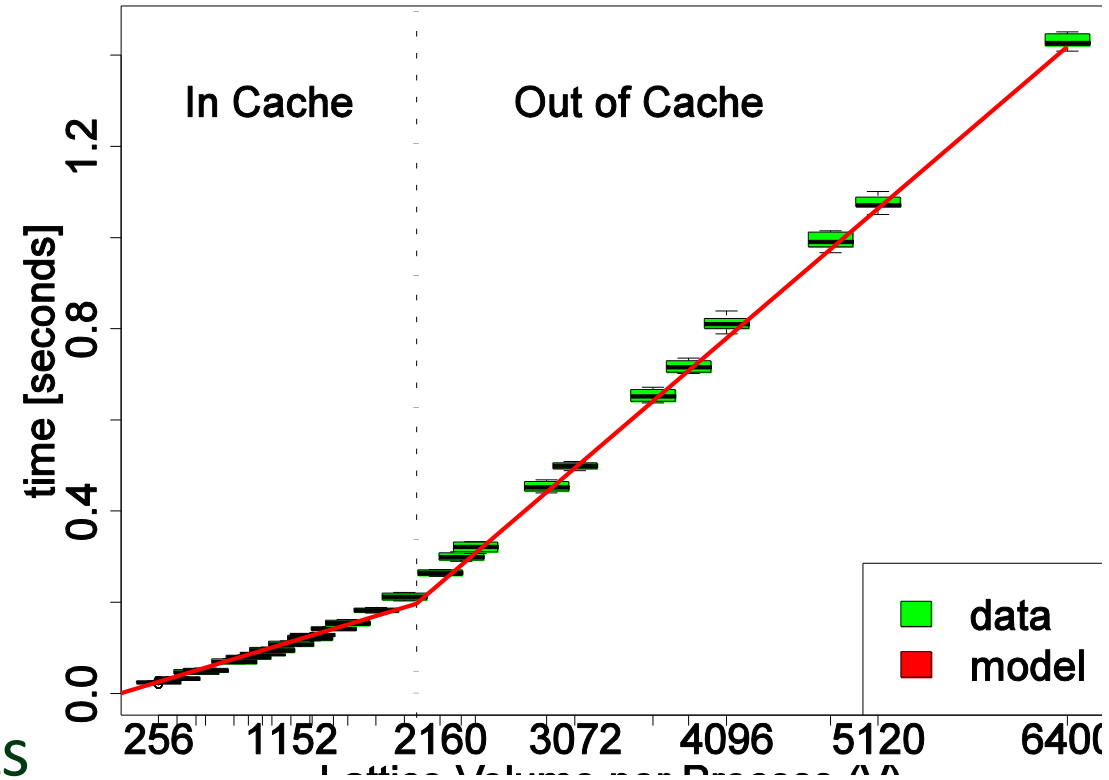


Step 4: Sequential Performance

- MILC "only" loops over the lattice → $\Theta(V)$
- T(V) = tV
  - Wait, it's not that simple with caches ☹
  - Small V fit in cache!
- $T(V) = t_1 * \min(s, V) + t_2 * \max(0, V-s)$
  - Cache holds s data elements
  - Three parameters for each kernel

An Example Kernel: GF (Gauge Force)

- Hopper (XE6):
  - t_1=81 µs, t_2=261 µs
  - s=1.500
- Kraken (XT-5):
  - t_1=74 µs, t_2=387 µs
  - s=1.500
- Surveyor (BG/P):
  - T_1=483 µs, t_2=567 µs
  - s=2000

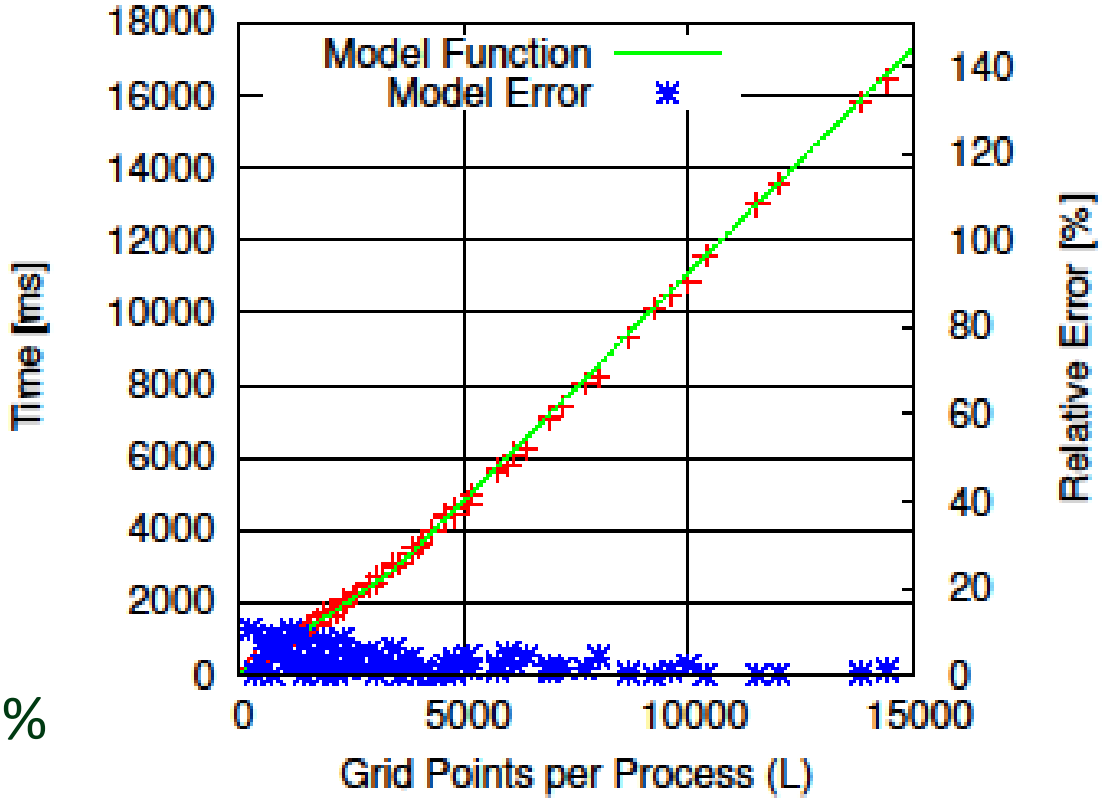


Data from Hopper

Bauer, Gottlieb, Hoefler: "Performance Modeling and Comparative Analysis of the MILC …, CCGRID 2012c

Complete Serial Performance Model

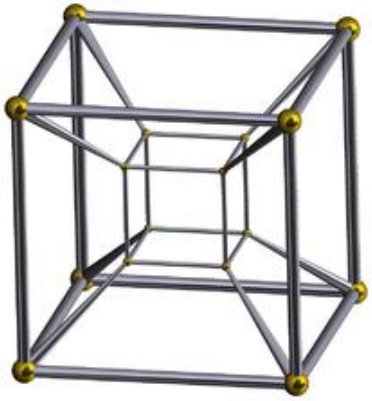
$$T_{serial}(V) = (\texttt{trajecs} + \texttt{warms}) \cdot \texttt{steps} \cdot [T(FF,V) + T(GF,V) + 3(T(LL,V) + T(FL,V))] + \left\lfloor \frac{\texttt{trajecs}}{\texttt{meas}} \right\rfloor [T(LL,V) + T(FL,V)] + \texttt{niters} \cdot T(CG,V)$$



- High predictability!
- Low variance
- Avg. model error <5%

Step 3: Communication Pattern

- 4d domain is cut in all dimensions (cubic)
  - 4d nearest-neighbor communication (8 neighbors)
- Allreduce to check CG convergence
  - One per iteration on full process set
- We counted messages and sizes
  - Separate for each kernel
  - See paper for sizes and full model equation!



kernel	$\#Messages$
FF	$(\texttt{trajecs} + \texttt{warms}) \cdot \texttt{steps} \cdot 1616$
GF	$(\texttt{trajecs} + \texttt{warms}) \cdot \texttt{steps} \cdot 828$
LL	$(3 \cdot \texttt{steps} \cdot (\texttt{trajecs} + \texttt{warms}) + \lfloor \frac{\texttt{trajecs}}{\texttt{meas}} \rfloor) \cdot 8$
FL	$(3 \cdot \texttt{steps} \cdot (\texttt{trajecs} + \texttt{warms}) + \lfloor \frac{\texttt{trajecs}}{\texttt{meas}} \rfloor) \cdot 288$

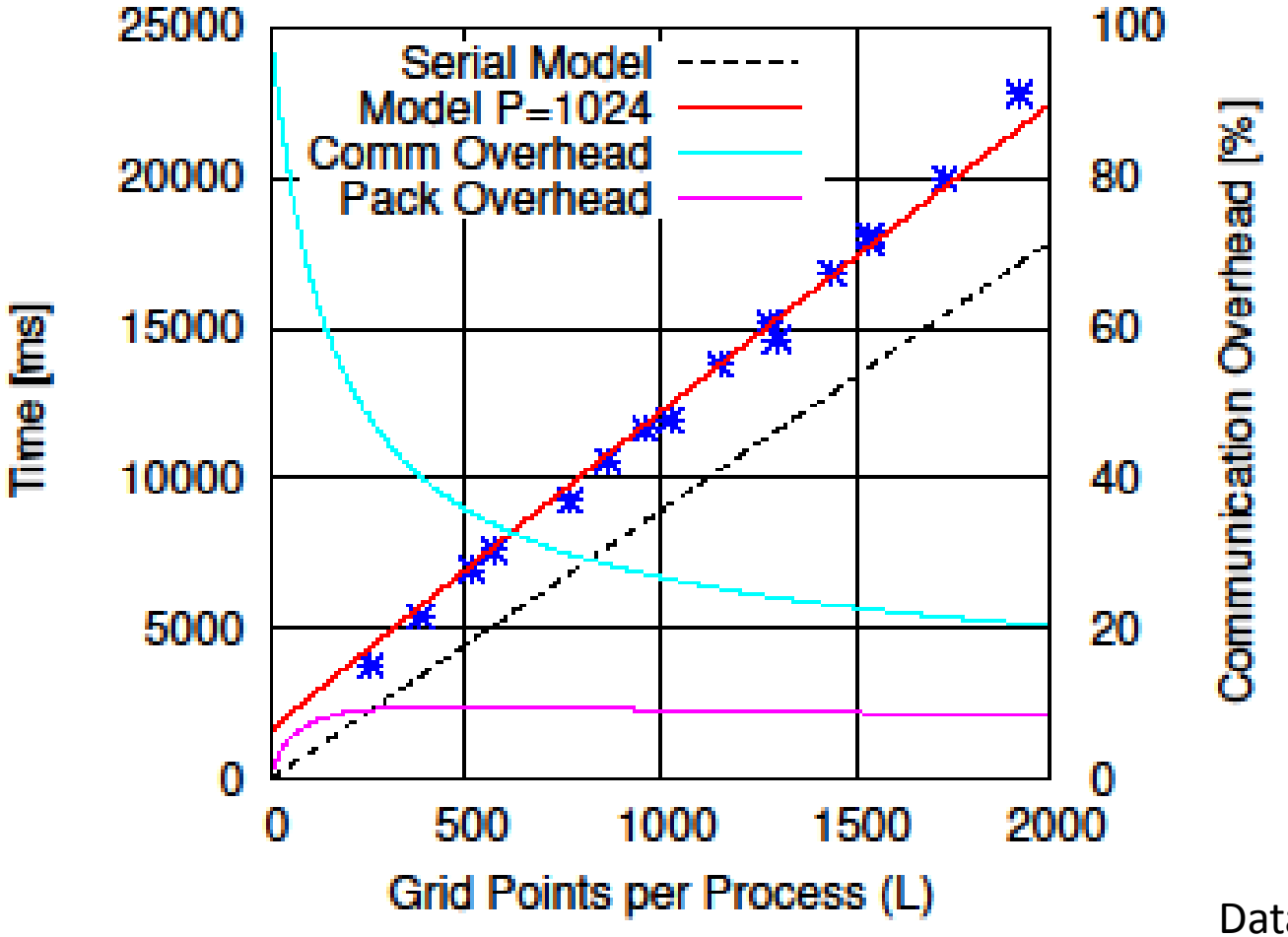
Bauer, Gottlieb, Hoefler: "Performance Modeling and Comparative Analysis of the MILC …, CCGRID 2012

Step 6: Communication Parameters

- Two options:
  - **Semi-analytic** – fit measurements to get effective latency and bandwidth
    - Enables to check if they match expectations
  - **Analytic** – derive parameters separately (e.g., documentation or separate benchmark)
    - Often problematic if they do not match expectations
- We did both! "Measure" impact of topology!
  - Uses **analytic** LogGP parameters (measured by Netgauge [1])
  - Observe **effective** bandwidth and latency **semi-analytically**!

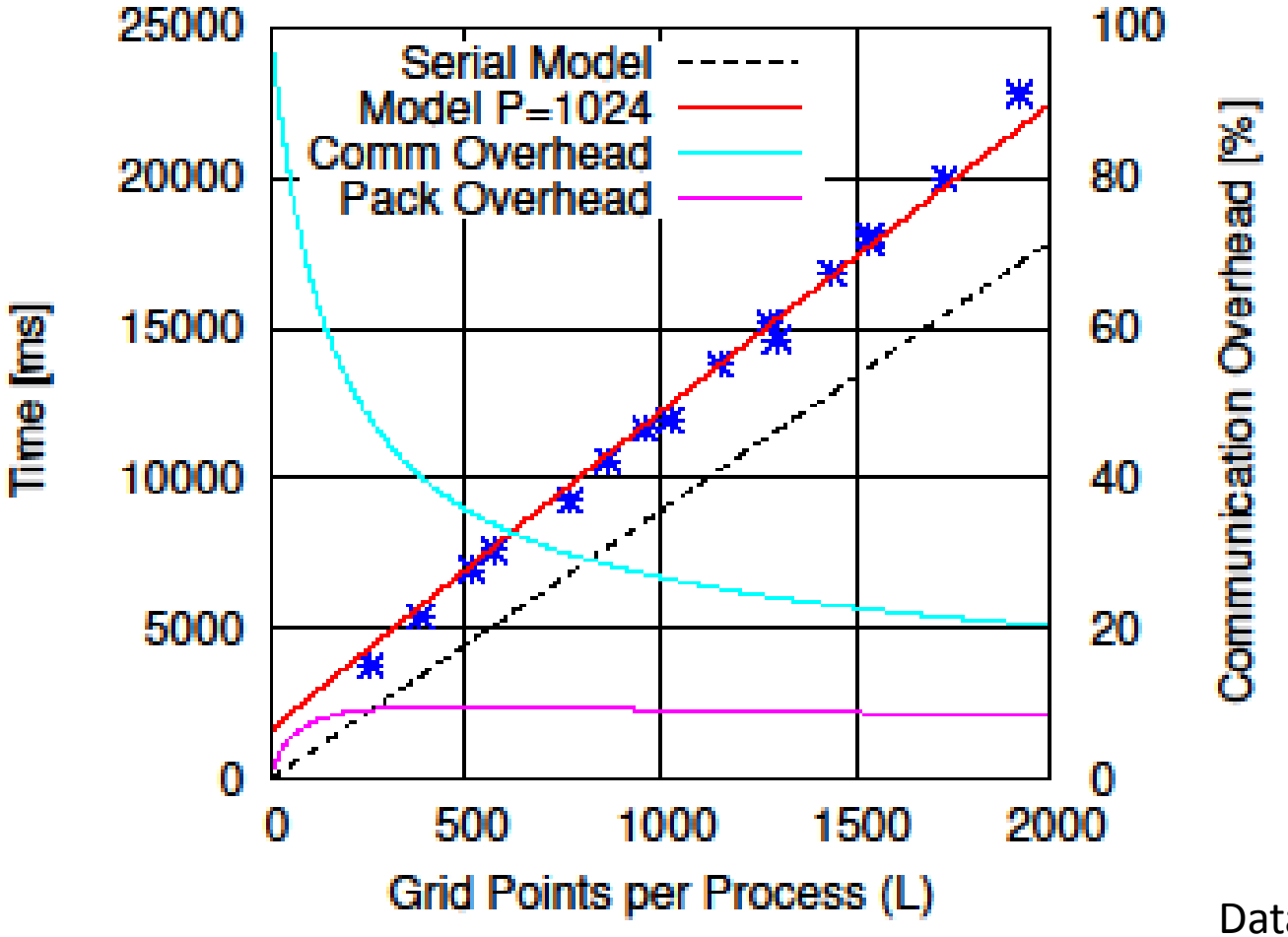
[1] Hoefler et al.: Low-Overhead LogGP Parameter Assessment for Modern Interconnection Networks

The Analytic Parallel Model



Data from POWER5

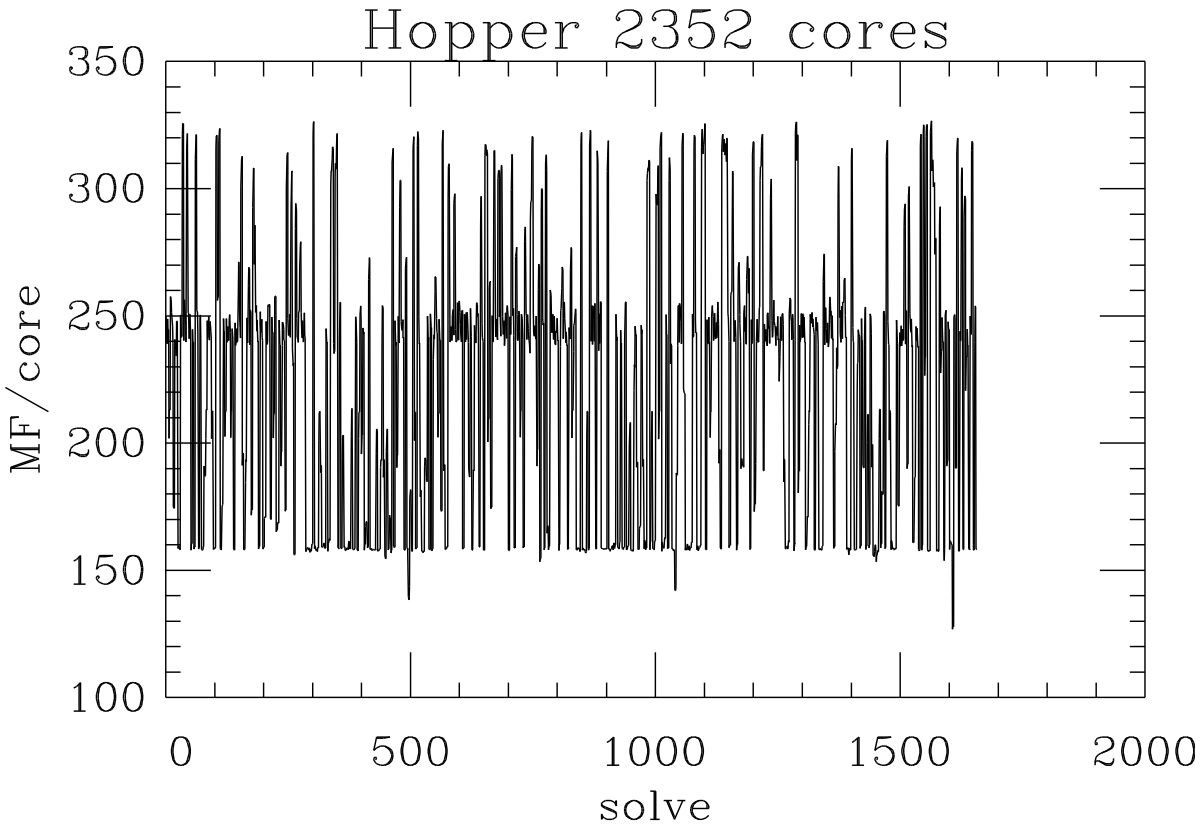
The Analytic Parallel Model



Data from POWER5

BUT WHAT IF …

- … you have a machine like this (from a user):



Graph by Steven Gottlieb, Indiana University

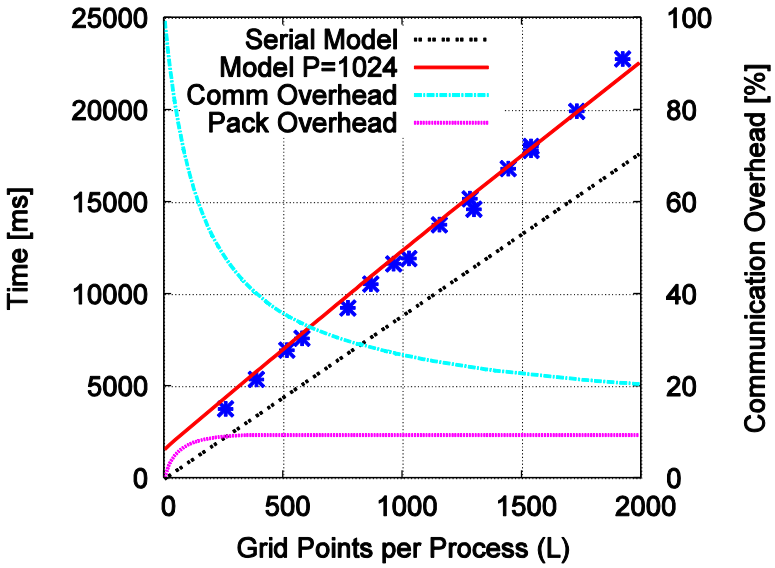
Statistical Modeling

- User functions as **expected performance**
  - Capture variance during measurements as deviation model → machine characteristic!
  - 99% network variations in our tests
- Effective latency and bandwidth (+variance) [1]:
  - BG/P (P=4096): 16.1 us (2%), 118 MiB/s (0.2%)
  - XT-5 (P=2048): 10.3 us (5%), 211 MiB/s (3.8%)
  - XE6 (P=8291): 41.5 us (4.8%), 232 MiB/s (1.7%)
  - IB (P=2048): 33.6 us (16%), 164 MiB/s (3%)
- Relatively low network variance leads to high performance variance → conjecture network noise [2]

[1]: Bauer, Gottlieb, Hoefler: "Performance Modeling and Comparative Analysis of the MILC …, CCGRID 2012
[2]: Hoefler, Schneider, Lumsdaine: "The Effect of Network Noise on […] Collective Communication, PPL 2009

Use-Case 1: Model-guided Optimization

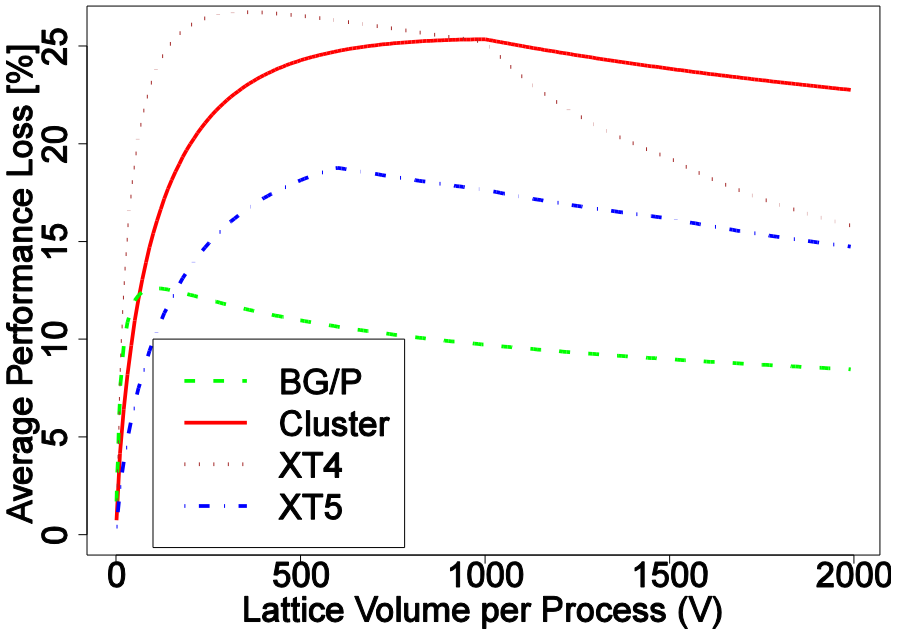
- Parallel application performance is complex
  - Often unclear how optimizations impact performance
- Issue for applications at large-scale
  - Models can guide optimizations
- The developed model shows:
  - Local memory copies to prepare communication are significant
    - Re-engineering resulted in 20% performance gain of a QCD code
  - Frequent communication synchronizations are critical
    - Importance increases with P – new algorithms in development



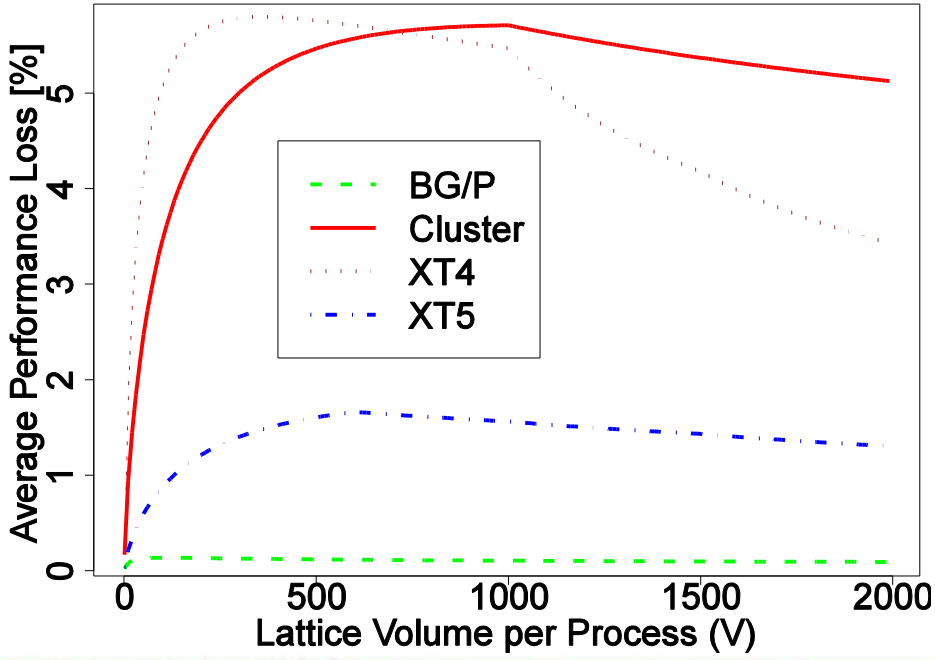
USE-CASE 2: ARCHITECTURAL OBSERVATIONS

- How important is topology (at 1024 cores)?
  - Compare LogGP analytic results with effective BWs

Performance degradation over ideal (uncongested) network



Expected performance degradation due to network variance



Collaboration Opportunities

- Autotuning – use model-driven tuning
  - Use user-specified functions to improve the surrogate function of the tuning search space
- Fault-tolerance – determine best checkpoint places
  - Maybe introduce application-specific checkpointing automatically or support the user to do so

- Application Modeling – done for many SPP apps, may be improved
  - Apply and develop automatic tools?
- Runtime Performance Models – provide performance guarantees
  - E.g., MPI implementations ☺ - provable performance?
- Investigate network topologies – different core counts
- Others: how to deal with the variance? Are stochastic models a good idea? Better job placement/scheduling? What else?

ACKNOWLEDGMENTS

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

NSF

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA ·

Sandia National Laboratories