BLUE WATERS

SUSTAINED PETASCALE COMPUTING

Application Performance Modeling on Petascale and Beyond

Torsten Hoefler

GREAT LAKES CONSORTIUM

FOR PETASCALE COMPUTATION

Imagine …

- … you’re planning to construct a multi-million Dollar Supercomputer …
- … that consumes as much energy as a small [european] town …
- … to solve computational problems at an international scale and advance science to the next level …
- … with “hero-runs” of [insert verb here] scientific applications that cost $10k and more per run …

… and all you have (now) is …

- … then you better plan ahead! (same for Exascale)

What is Performance Modeling?

- Understand the resource usage of an application on a particular architecture
  - We focus mostly on time as a resource
  - Generate analytic expressions to estimate runtime
- Closely related to “Performance Engineering”
  - Often builds on empirical techniques
- Also cutting into complexity theory
  - More pragmatic (asymptotes often insufficient)
  - Complex (low-order terms cannot be dropped)

Execution-Time Modeling - Basics

- Set of performance-critical input variables
  - $X=\{x_1, x_2, \ldots, x_n\}$
  - e.g., size of the system, number of CPUs
- Application requirements model
  - Vector of requirements: $P(X) = f(x_1, x_2, \ldots, x_n)$
- System model
  - Vector of performance characteristics $C=\{c_1, c_2, \ldots, c_m\}$
    - Problematic if not all are independent (e.g., superscalar arch.)
- Performance Prediction
  - $T(X,C) = \min_{i=1..|C|}(p_i(X) * c_i)$

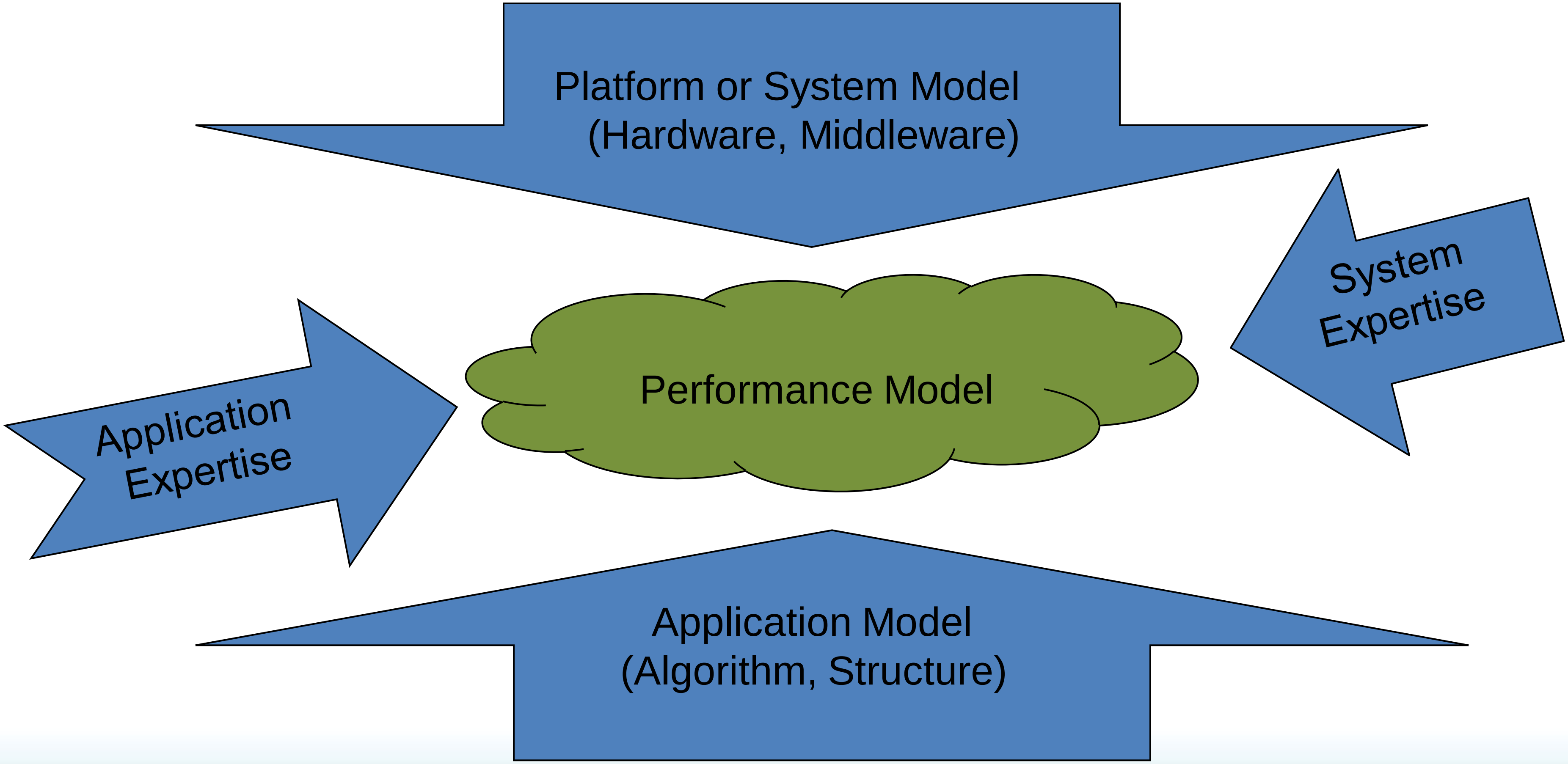
Performance Modeling – Quick Review

- Single CPU performance models (Davidson et al.)
  - Limited to (specific) relatively simple architectures
  - Investigate quality of compilers
- Cache models (Ding et al.)
  - Based on reuse-distance, good for BOE analysis
- Prediction based on convolution (Snavely et al.)
  - Model-driven - faster than detailed simulation
  - Less insight than analytic models

Modeling Parallel Applications – Quick Review

- Classification-based modeling (Schulz et al.)
  - Neural networks
  - Does not allow for extrapolation
- Regression-based (Lowenthal, Schulz, de Supinsky)
  - Least-squares fitting of low-order polynomials
    - Or fitting on a log-scale
- Manual application modeling (Kerbyson et al.)
  - Requires deep understanding of applications

Manual Performance Modeling from 10.000 Feet



Platform or System Model
(Hardware, Middleware)

System Expertise

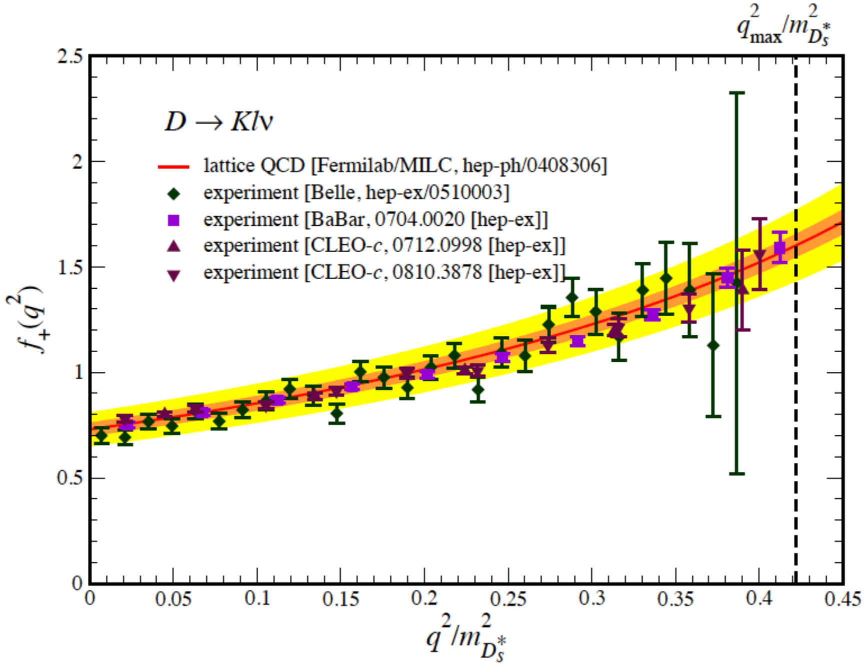
Application Expertise

Performance Model

Application Model
(Algorithm, Structure)

An Application Modeling 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



MILC - Performance-critical Parameters

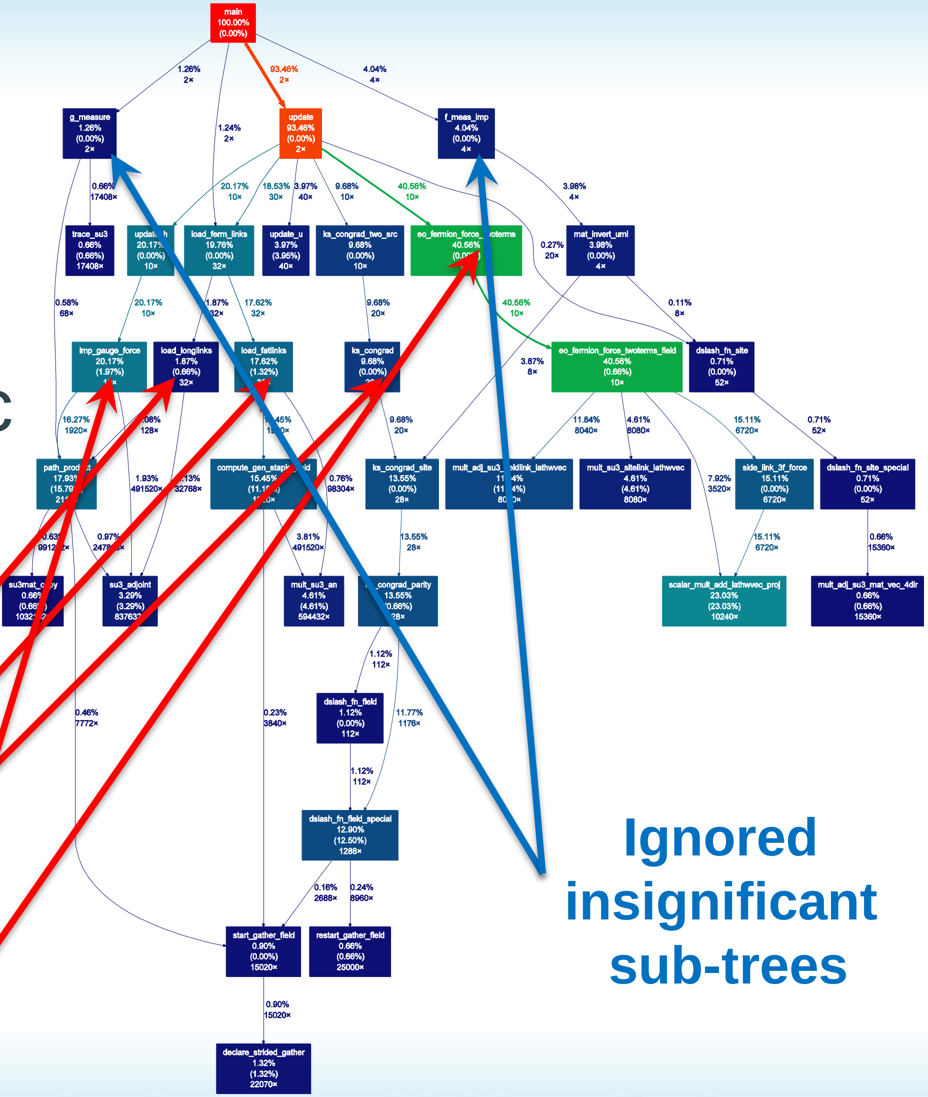
Name	simple	complex	comment
P	X		Number of processes
nx, ny, nz, nt	X		Lattice size in x,y,z,t
warms, trajecs	X		Warmup rounds and trajectories
traj_between_meas	X		Number of “steps” in each trajectory
beta, mass1, mass2, error_for_propagator		X	Physical parameters – influence convergence of conjugate gradient
max_cg_iterations		X	Limits CG iterations per step

- If parameters are more complex (e.g., input files) then the user has to distill them into singletons (domain specific)

MILC – Critical Blocks

- Identify sub-trees in call-graph with same performance characteristic
- Five blocks in MILC

Name	Function
LL	load_longlinks
FL	load_fatlinks
CG	ks_congrad
GF	imp_gauge_force
FF	eo_fermion_force


main 100.00% (0.00%)
g_measure 1.26% (0.00%) 2×
update 93.46% (0.00%) 2×
f_meas_imp 4.04% (0.00%) 4×
trace_su3 0.66% (0.66%) 17408×
update_h 20.17% (0.00%) 10×
load_ferm_links 19.76% (0.00%) 32×
update_u 3.97% (3.95%) 40×
ks_congrad_two_src 9.68% (0.00%) 10×
eo_fermion_force_twoterms 40.56% (0.00%)
mat_invert_uml 3.98% (0.00%) 4×
imp_gauge_force 20.17% (1.97%)
load_longlinks 1.87% (0.66%) 32×
load_fatlinks 17.62% (1.32%)
ks_congrad 9.68% (0.00%)
eo_fermion_force_twoterms_field 40.56% (0.66%) 10×
dslash_fn_site 0.71% (0.00%) 52×
path_product 17.93% (15.79%)
compute_gen_staple_field 15.45% (11.1%)
ks_congrad_site 13.55% (0.00%) 28×
mult_adj_su3_[illegible]eldlink_lathwvec
mult_su3_sitelink_lathwvec 4.61% (4.61%) 8080×
side_link_3f_force 15.11% (0.00%) 6720×
dslash_fn_site_special 0.71% (0.00%) 52×
su3mat_copy 0.66%
su3_adjoint 3.29% (3.29%)
mult_su3_an 4.61% (4.61%) 594432×
ks_congrad_parity 13.55% (0.66%) 28×
scalar_mult_add_lathwvec_proj 23.03% (23.03%) 10240×
mult_adj_su3_mat_vec_4dir 0.66% (0.66%) 15360×
dslash_fn_field 1.12% (0.00%) 112×
dslash_fn_field_special 12.90% (12.50%) 1288×
start_gather_field 0.90% (0.00%) 15020×
restart_gather_field 0.66% (0.66%) 25000×
declare_strided_gather 1.32% (1.32%) 22070×

Ignored insignificant sub-trees

Single CPU Model

- Analytic modeling is rather complex
  - We approximate a serial model with fitting a piecewise linear function
  - Volume V = nx*ny*nz*nt; Type B = {LL, FL, GF, CG, FF}
  - Cache holds s(B) data elements

$$T(\mathcal{B},V) = t_1(\mathcal{B}) \cdot min\{s(\mathcal{B}),V\} + t_2(\mathcal{B}) \cdot max\{0, V - s(\mathcal{B})\}$$

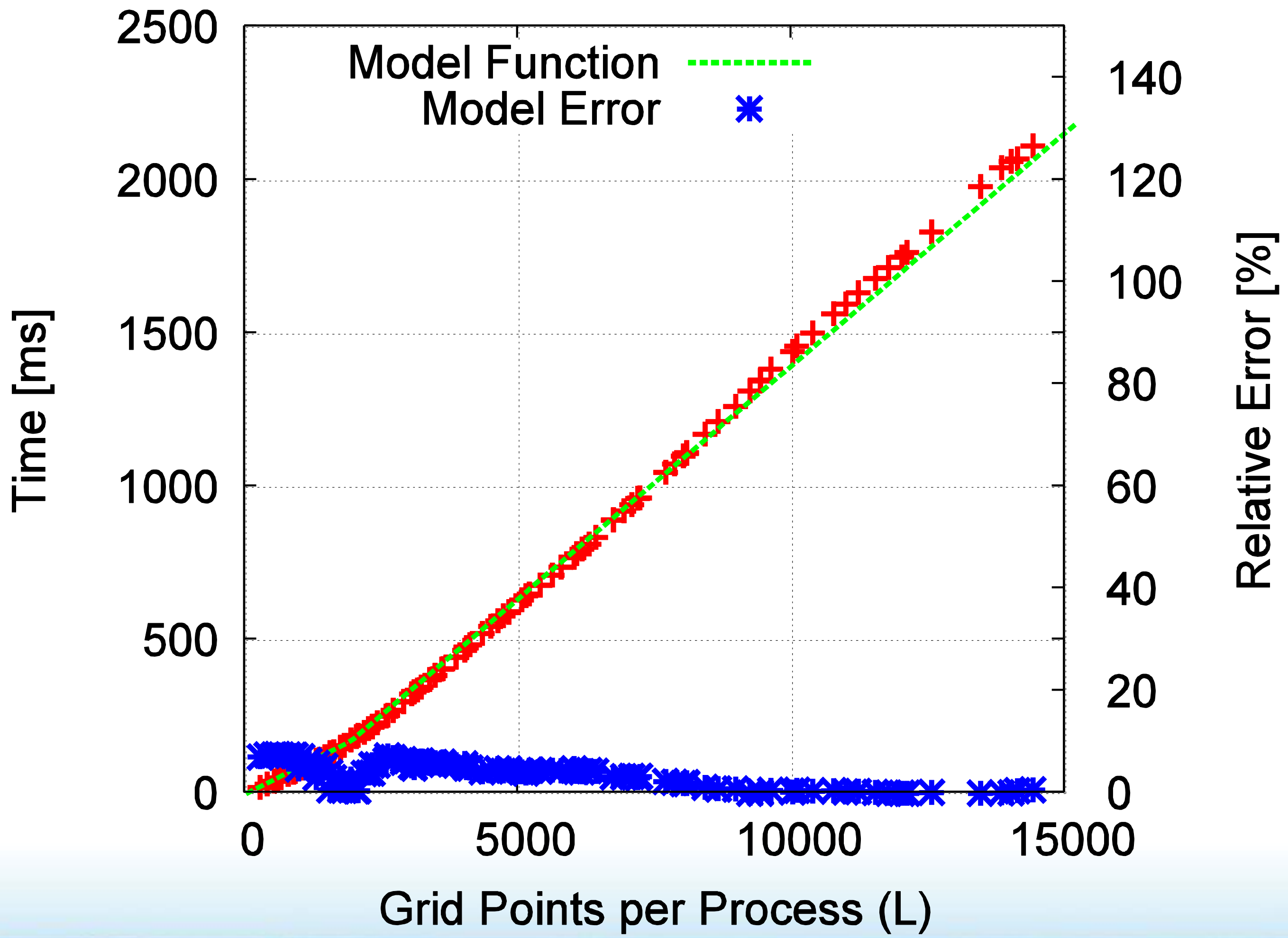
POWER 5+

$\mathcal{B}$	$t_1(\mathcal{B})[\mu s]$	$t_2(\mathcal{B})[\mu s]$	$s(\mathcal{B})$
FF	255	326	2500
GF	88	157	1900
LL	1.3	2.2	2500
FL	30	56	2000
CG	0.425	0.483	1200

POWER 7 MR

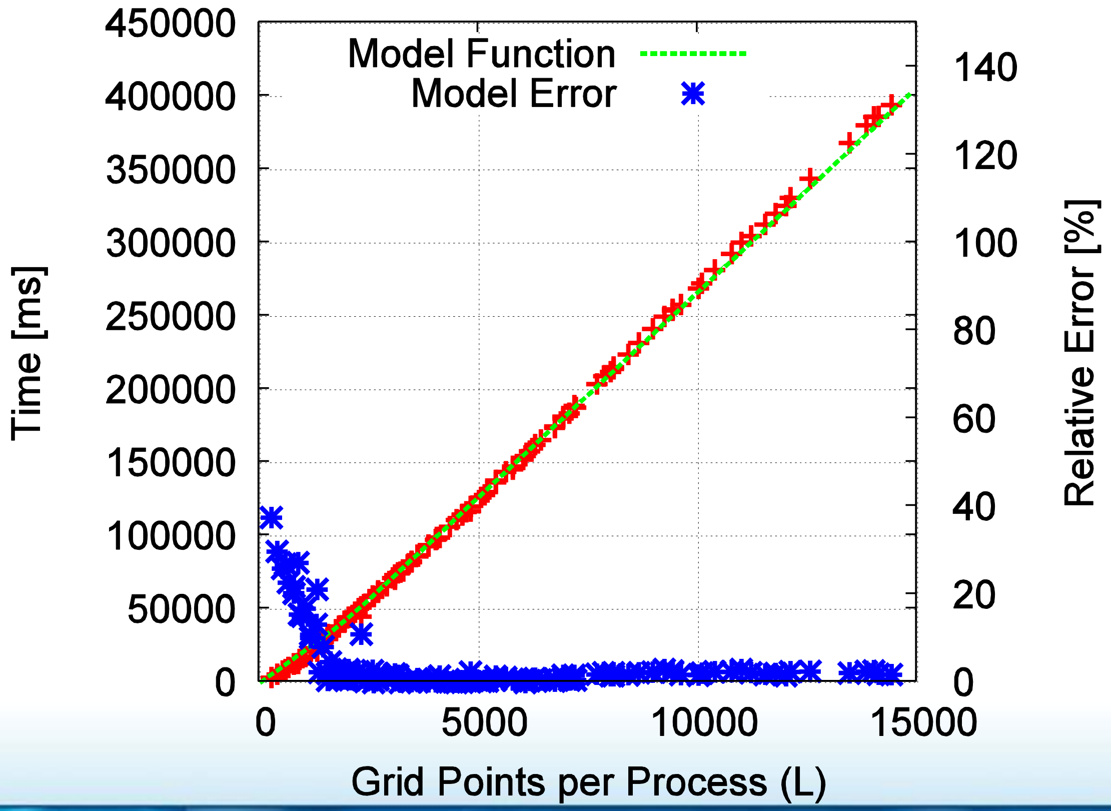
$\mathcal{B}$	$t_1(\mathcal{B})[\mu s]$	$t_2(\mathcal{B})[\mu s]$	$s(\mathcal{B})$
FF	62.4	92	3000
GF	27.8	48	4000
LL	0.425	0.68	4000
FL	11.4	20	3500
CG	0.239	-	∞

Example block: GF



Overall Serial (composed) MILC Model

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



Composing a Parallel Model

- First approximation
  - $T_{parallel} = T_{serial}(V/P) + T_{comm}(V/P)$
- Reality
  - Need to consider overlap ($-T_{overlap}$?)
  - Need to consider network congestion
    - Communication pattern
    - Collective operation times
  - Need to consider process-to-node mapping
  - Load imbalance and system noise

Application Communication Pattern

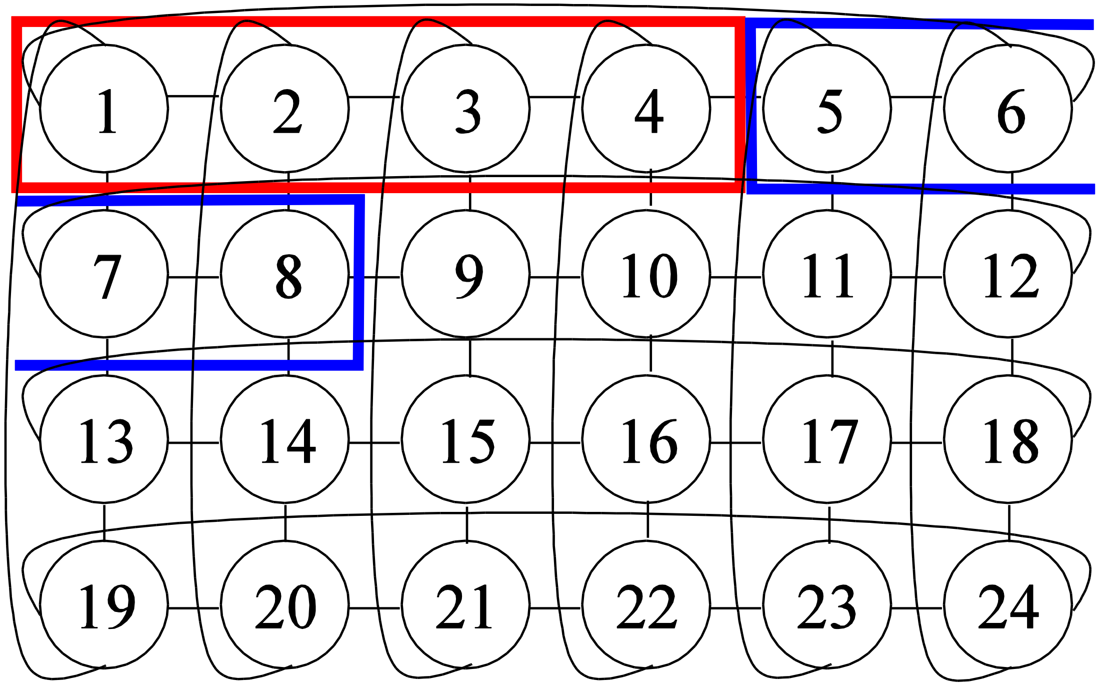
- Four-dimensional p2p communication topology
  - Prime-factor decomposition of P (→ square)
- Total number of p2p messages

Type	Number of Messages
FF	(trajecs + warms) · steps · 1616
GF	… (for LL, FL, CG)

  - Counted manually (profiling tools and source)
- Collective Communication
  - Single MPI_Allreduce per CG iteration

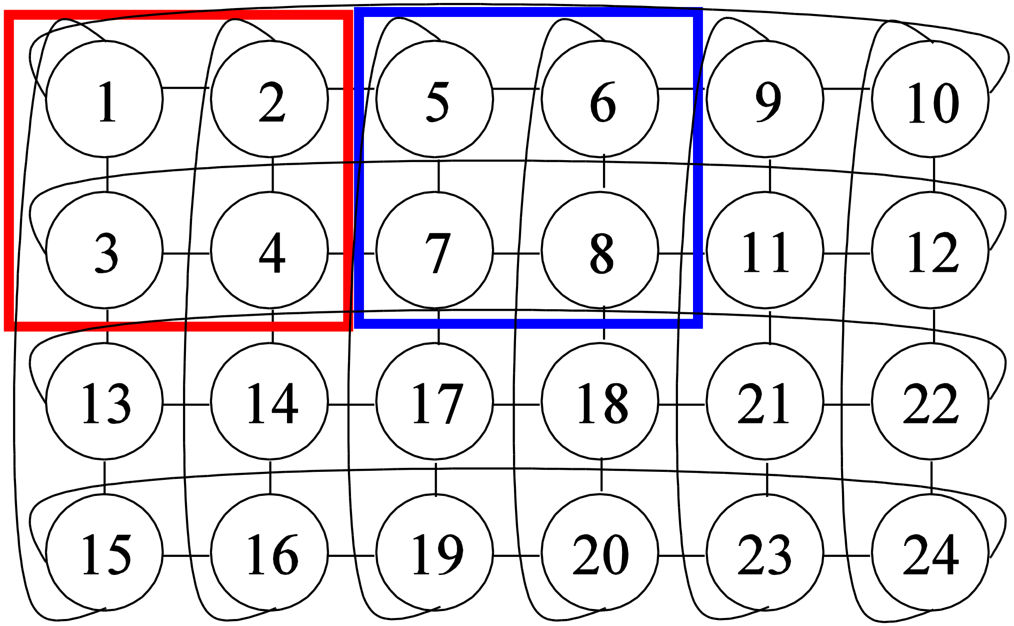
Process-to-Node Mapping – 2D Example

- Trivial linear default mapping
- With 4 processes per node:
  - 6 internal edges
  - 10 remote edges
- Wrap-around
  - Looses two internal edges
  - Unbalanced communication



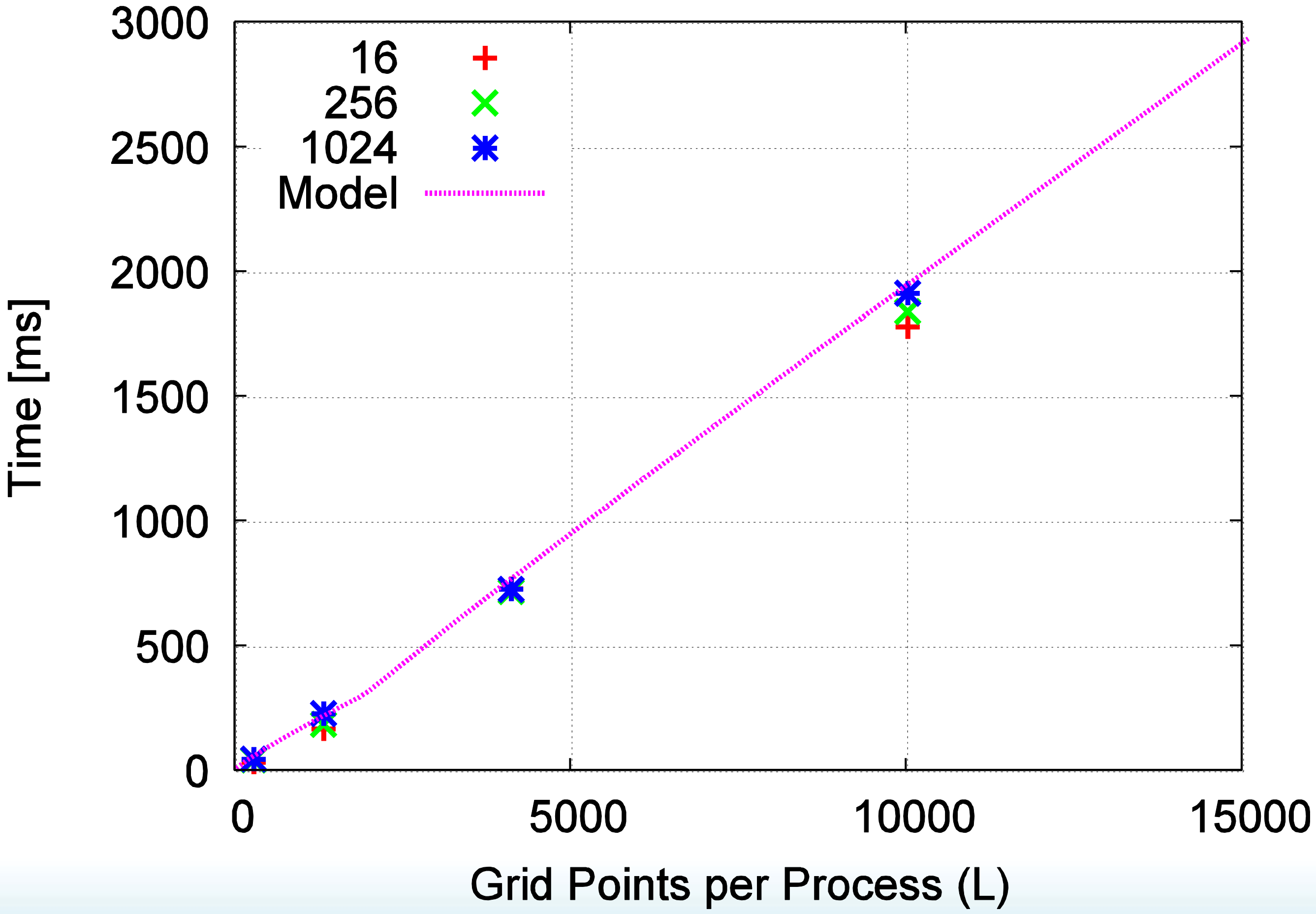
Optimized Process-to-Node Mapping

- Optimal mapping
  - cf. Lagrange multiplier
  - ~~6~~ 8 internal edges
  - ~~10~~ 8 remote edges



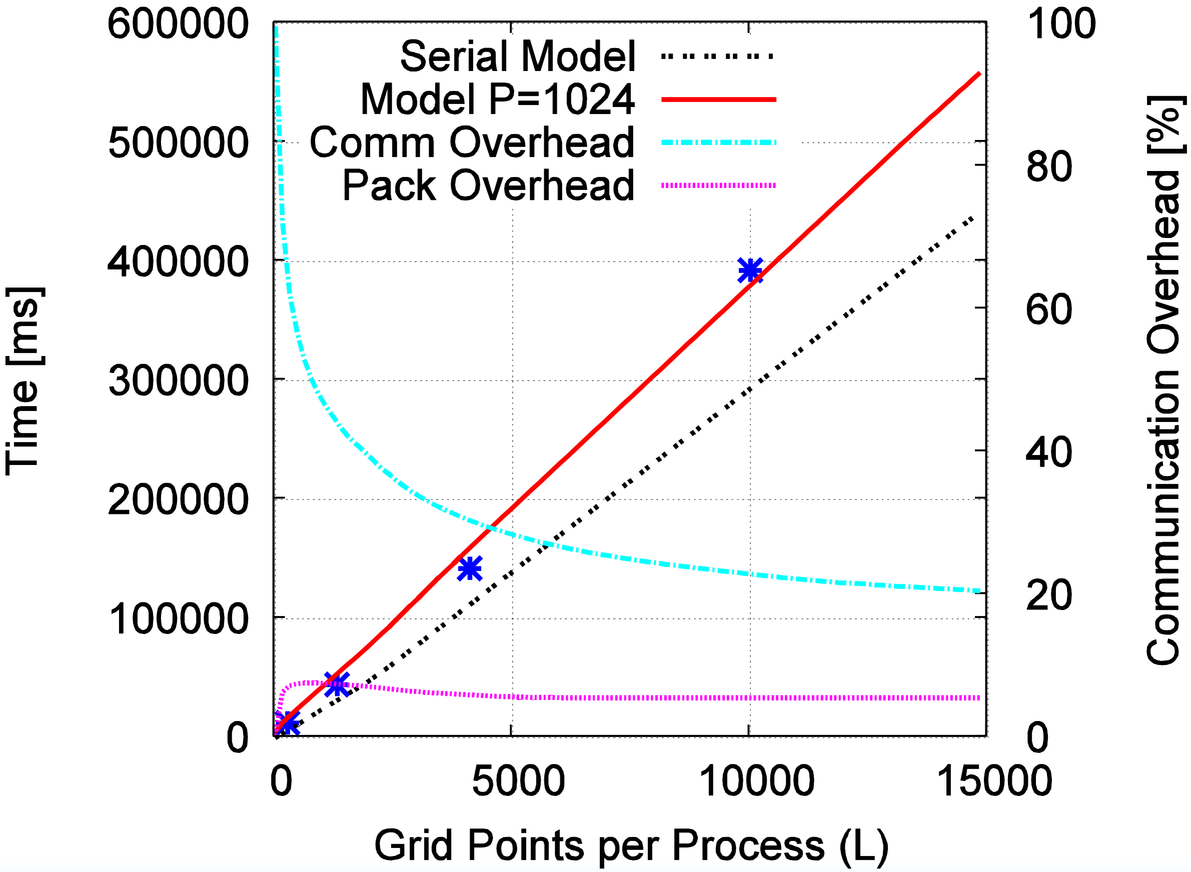
- Similar for 4d mapping
  - 16 cores, optimal sub-block: $\sqrt[4]{16} = 2 \cdot 2 \cdot 2 \cdot 2$
  - ½ remote edges

Parallel Performance Example: GF

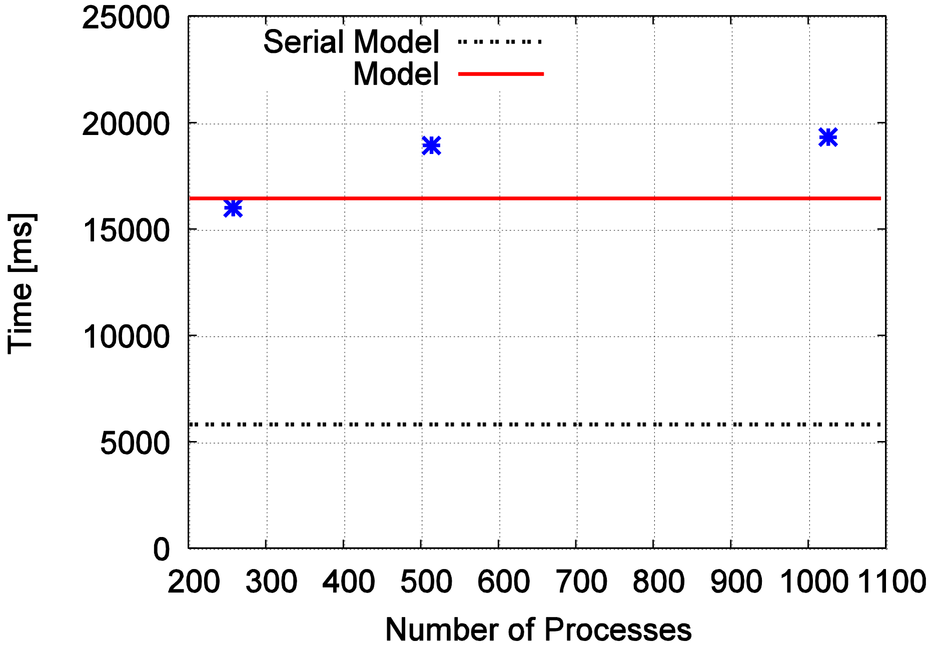


Parallel Performance Model

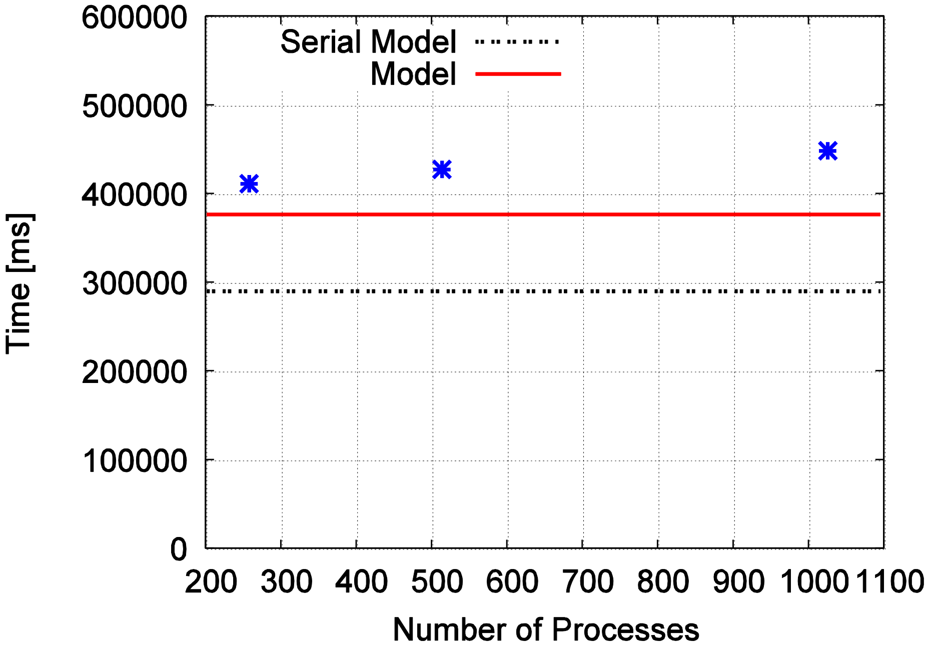
- Example: quantify maximum benefit of MPI datatypes!
  - cf. EuroMPI'10



Scaling with the Number of Processes



$V=6^4$



$V=10^4$

A Quick (Pre)View of Current Work on FFT

- Is IBM's single-core FFT (bandwidth) optimal?
  - Cf. Gropp's work on sparse solvers, Hong&Kung bounds
- Develop (bandwidth) optimal parallel FFT
  - Needs (at least) two-layer hybrid implementation
  - Requires optimal comp/comm overlap
  - Goal is to accurately model global behavior using LogGP
- Seeking for collaborations
  - Multiple implementations exist
  - Need to understand detailed performance characteristics

A Specific Example – 2D FFT

- Assuming 1D decomposed FFT of size N^2
  - $T_{FFT} = T_{comm} + T_{comp} + T_{trans}$
- Each process communicates $N^2(P-1)/P^2$ points
  - Linear alltoall: $T_{comm} = L+(P-1)\max\{g,o\}+(N^2(P-1)/P^2)G$
    - Is g, G, or L the dominating term?
    - BW global alltoall peak bandwidth: 0.8 PB/s
- We assume $T_{comm} > T_{comp}$ and $g > o$
  - i.e., $T_{FFT} = L+(P-1)g+(N^2(P-1)/P^2)G + T_{trans}$
  - Goal: minimize T_{FFT} and T_{trans} (and show optimality?)

Ideas for Automation/Collaboration – Tool support

- Model each function as a critical block
  - Automatic decomposition might lead to more blocks
  - User needs to provide asymptotic scaling function
    - e.g., T ~ nx*ny*nz*nt,
  - Statistical runs could automatically fit the parameters (could model cache linearly)
  - Similar techniques can be used for message counting
- Should investigate tool support for this!

More Ideas for Improvement and Collaboration

- Model-driven topology optimizations
  - Mapping (optimal?)
  - Renumbering (optimal?)
- Develop serial model
  - Reuse distance etc. (optimal?)
- Analytical model for system noise sensitivity
- Analytic modeling of irregular applications
  - AMR, load imbalance, etc.
  - Fully data-driven applications (e.g., graph-searches)

Acknowledgments & Discussion

- Performance Modeling should become routine!
  - Modeling during application and system design
  - Needs tool support
    - Existing, needs glue
- All ideas are influenced by
  - Marc Snir, Bill Gropp, Bill Kramer, and all aforementioned publications