BLUE WATERS

SUSTAINED PETASCALE COMPUTING

February 28, 2011

Performance Modeling for Systematic Performance Tuning

William Gropp, *Torsten Hoefler*, Marc Snir

NCSA NSF IBM GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION

Imagine …

- … you're to optimize applications to run on a multi-hundred-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 …

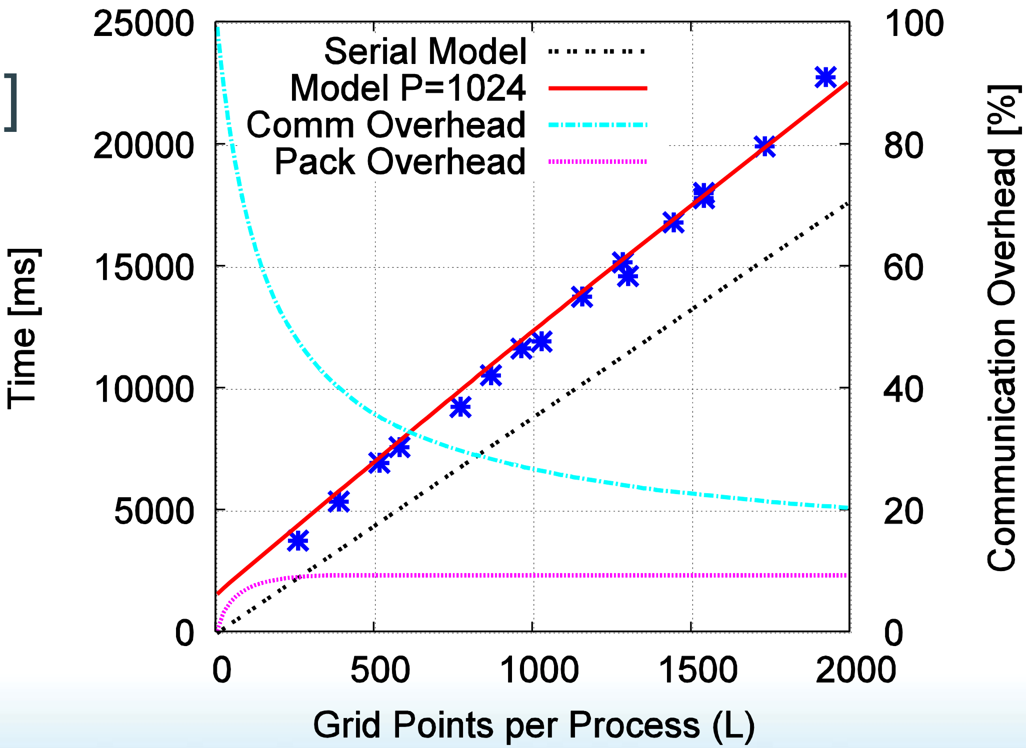
- … then you better plan ahead! (same for Exascale)

Model-guided Optimization - Motivation

- Parallel application performance is complex
  - Often unclear how optimizations impact performance
  - Especially at scale or different architectures!
- Big issue for applications on large-scale systems
  - Need to guide optimizations
- One of our models shows:
  - Local memory copies to prepare communication are significant
    - Relative importance grows at scale
  - Frequent communication synchronizations are critical
    - Importance increases with P

Model-guided Optimization - Potential

- Analytic model showed possible improvement of 12% by eliminating the pack before communicating
- Implemented and analyzed in [EuroMPI'10]
  - Demonstrated benefit of up to 18%
- Next bottleneck: CG phase
  - Investigating use of nonblocking collectives
  - Also model-driven



What is Performance Modeling

- Representing application performance with **analytic** expressions
  - Not just series of points from benchmarks
  - Enables derivation to find sweet-spots

- Why performance modeling?
  - Extrapolation (scalability in P or with input system)
  - Insight into requirements (message sizes etc.)
    - Guide system design and optimization
  - Expectations for porting to a different architecture

Our Methodology

- Combine analytical methods and performance measurement tools
  - Programmer specifies parameterized expectation
    - E.g., $T = a+b*N^3$
  - Tools find the parameters with benchmarks
    - E.g., least squares fitting
  - *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.*

Different Philosophies

- Simulation:
  - Very accurate prediction, little insight
- Traditional Performance Modeling (PM):
  - Focuses on accurate predictions
  - Tool for computer scientists, not application developers
- Our view: PM as part of the software engineering process
  - PM for design, tuning and optimization
  - PMs are developed with algorithms and used in each step of the development cycle
  - ➢ Performance Engineering

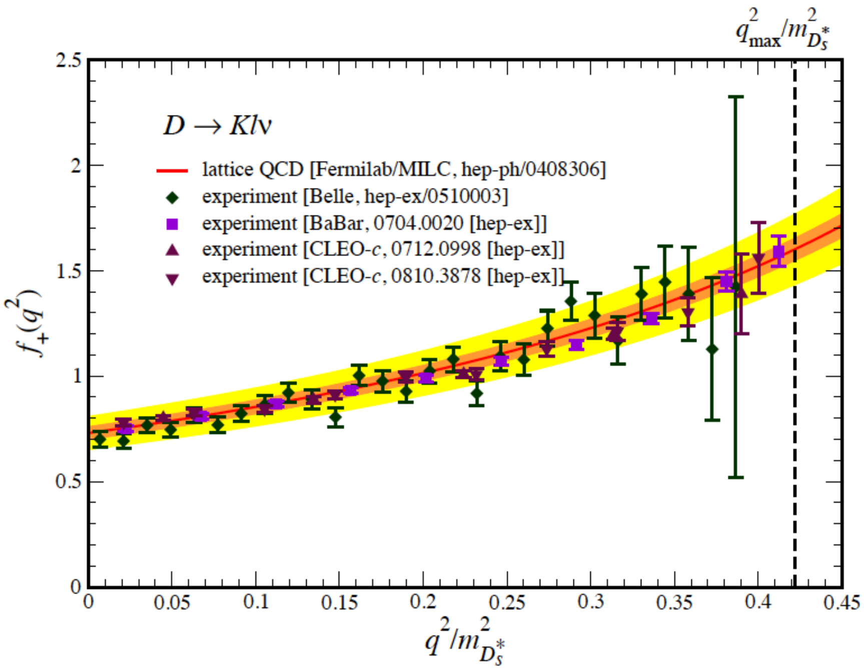
Our Process for Existing Codes

- Simple 6-step process:

- Analytical steps (domain expert or source-code)
  - Step 1: identify input parameters that influence runtime
  - Step 2: identify most time-intensive code-blocks
  - Step 3: determine communication pattern
  - Step 4: determine communication/computation overlap
- Empirical steps (benchmarks/performance tools)
  - Step 1: determine sequential baseline
  - Step 2: communication parameters

An 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 – Quick Model Walkthrough

- Performance-critical parameters

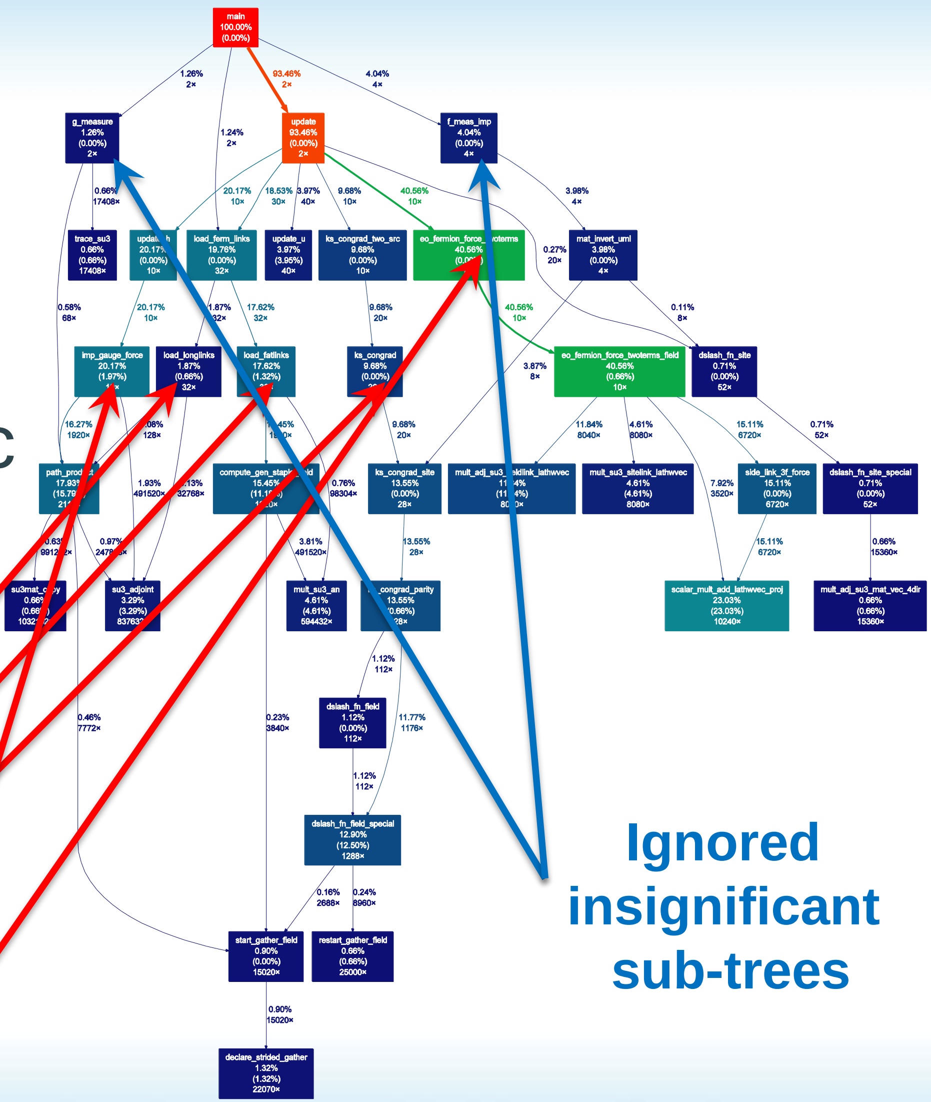
Name	simple	complex	comment
P	X		Number of processes
nx, ny, nz, nt	X		Lattice size in x,y,z,t
warms, trajecs	X		Warmup rounds and trajectories
traj_between_meas	X		Number of “steps” in each trajectory
beta, mass1, mass2, error_for_propagator		X	Physical parameters – influence convergence of conjugate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



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

Sequential Baseline

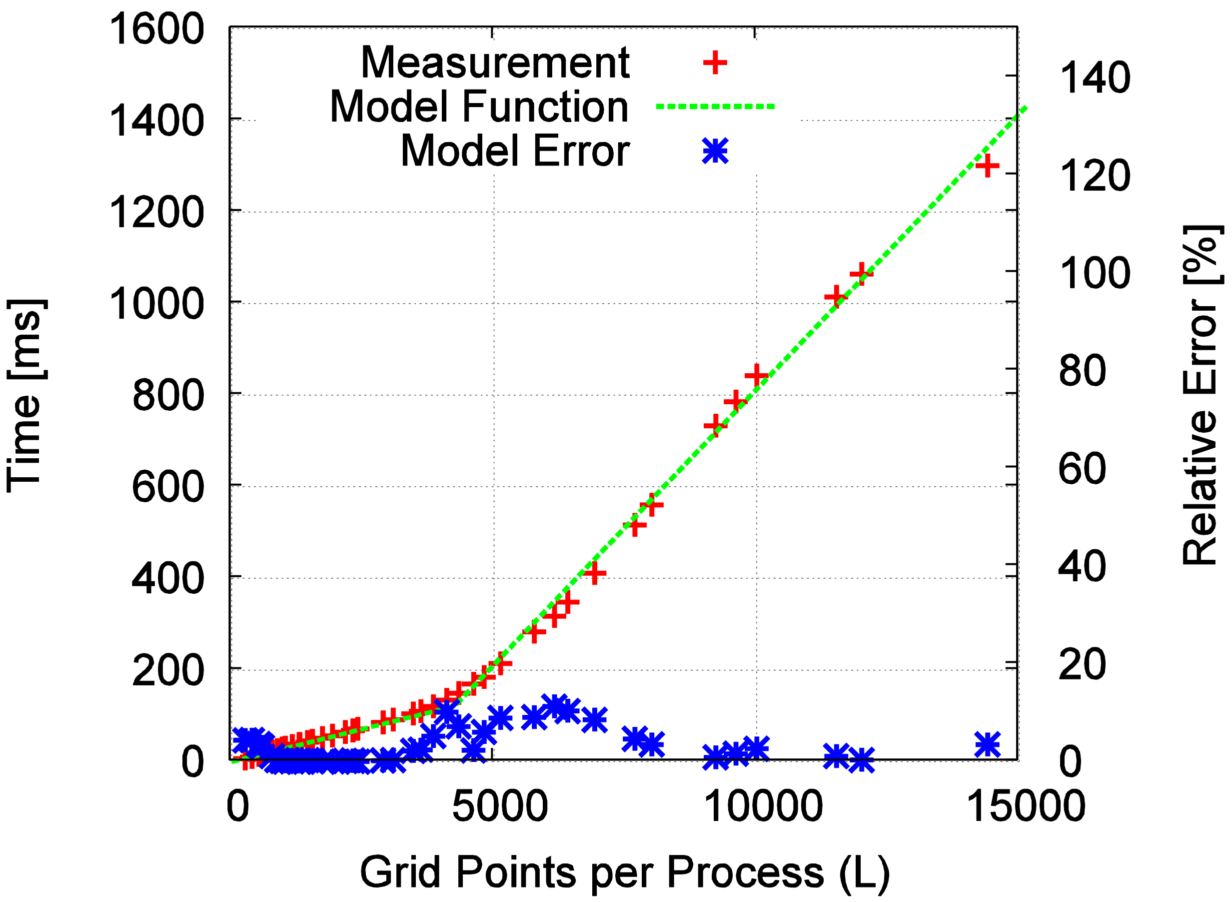
- Stepwise linear function to represent cache influence
  - Chose two steps, different CPUs might need more
  - Volume V = nx*ny*nz*nt; Type B = {LL, FL, GF, CG, FF}
  - Cache holds s(B) data elements

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

Power7 MR

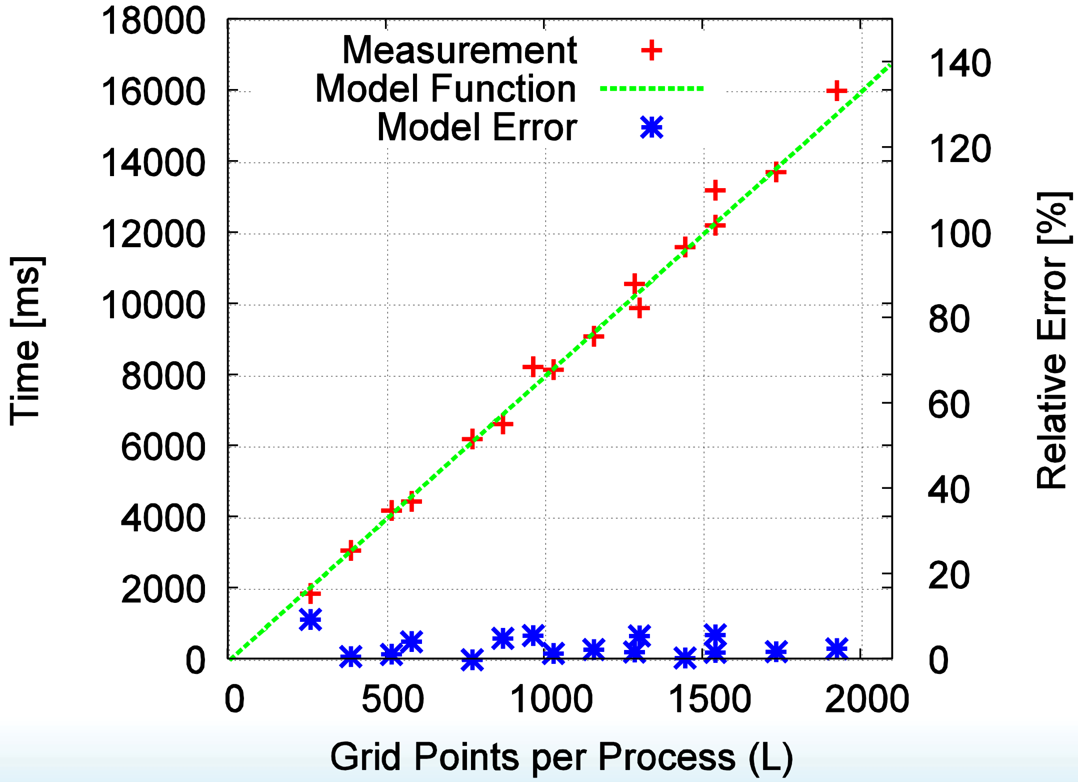
$\mathcal{B}$	$t_1(\mathcal{B})[\mu s]$	$t_2(\mathcal{B})[\mu s]$	$s(\mathcal{B})$
FF	62.4	92	3000
GF	27.8	48	4000
LL	0.425	0.68	4000
FL	11.4	20	3500
CG	0.239	-	∞

Example block: GF



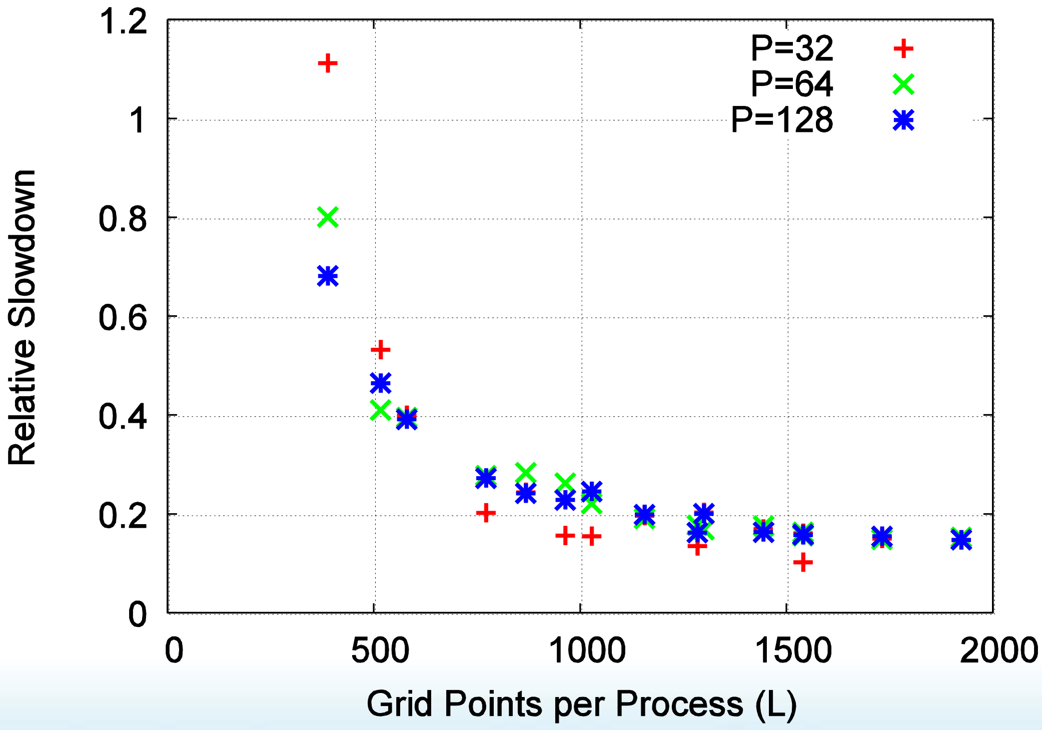
Overall (composed) MILC Model

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

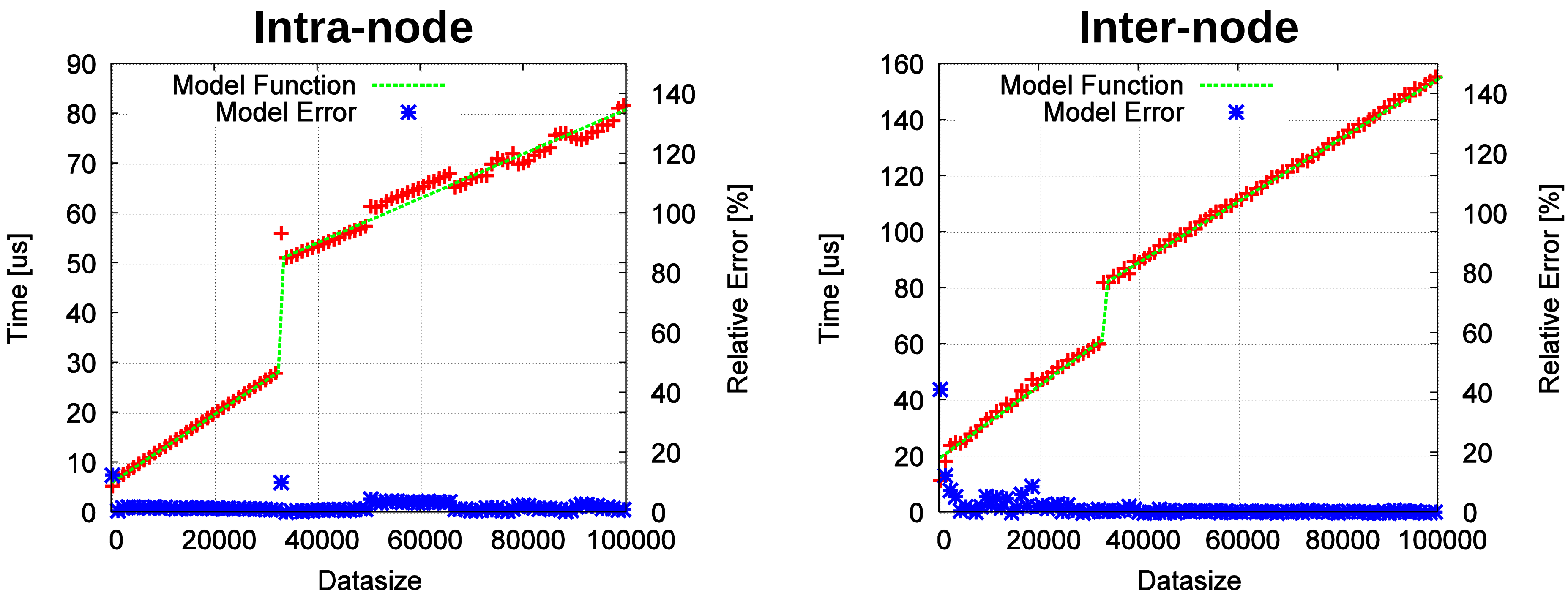


On-node Memory Contention

- Two cores share one memory controller
  - Congestion has complex performance effects
  - Empirical analysis
  - Assume fixed 20% slowdown

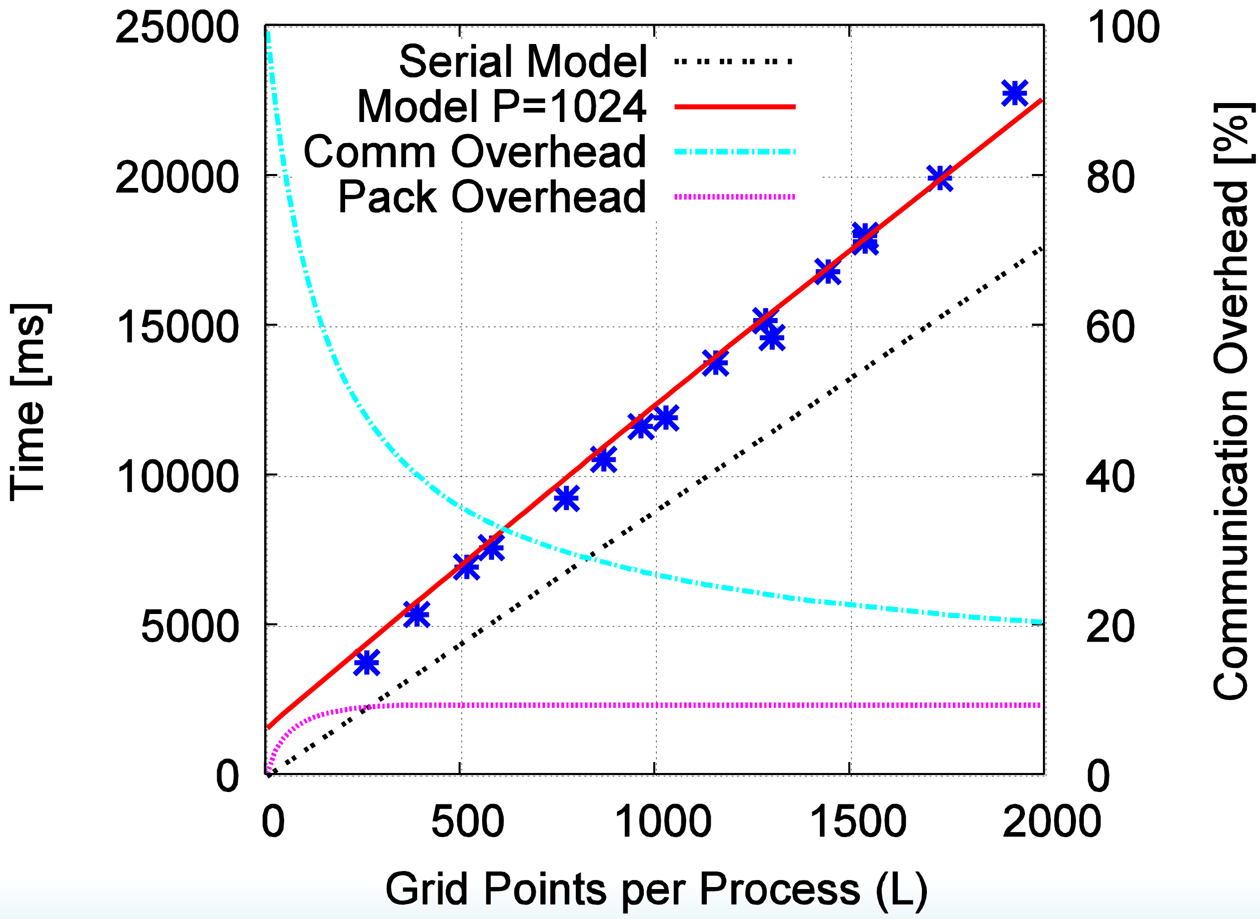


System Model: Communication Parameters



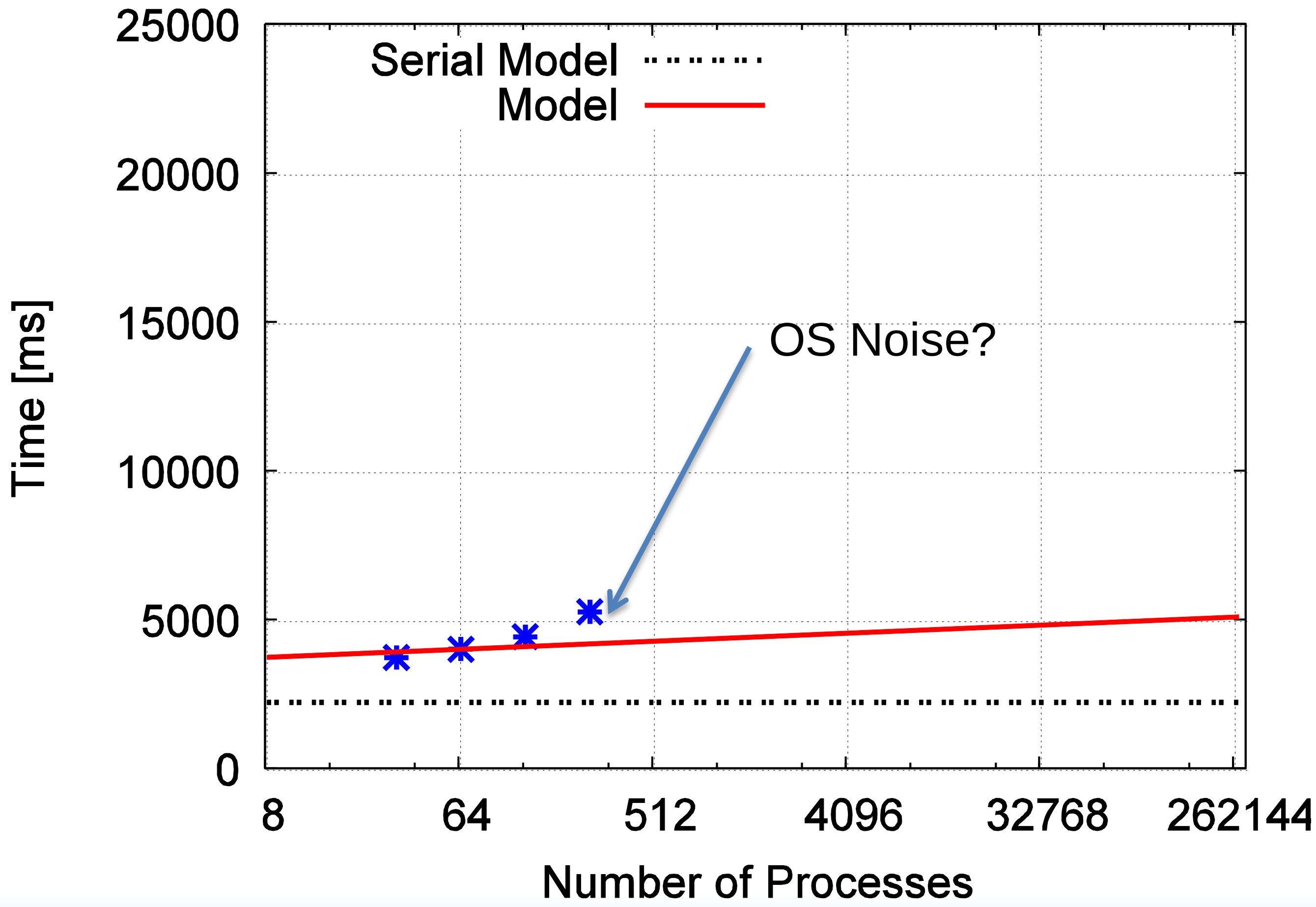
link	range	L	o	g	G
intra	$0 < S \leq 32768$	2.7	3.5	3.0	0.00068
intra	$S > 32768$	2.7	33.5	3.0	0.00045
...	...	...	...	...	...

Parallel Performance Model



Weak Scaling to 300.000 Cores

V=6^4



Conclusions

- We advocate performance modeling as tool for
  - Increasing performance
  - Guide application design and tuning
  - Guide system design and tuning

- Early results and key takeaways:
  - PM has been successfully applied to large codes
  - PM-guided optimization does not require high precision
  - Looking for insight with rough bounds is efficient