BLUE WATERS

SUSTAINED PETASCALE COMPUTING

Performance Modeling for the Masses

Torsten Hoefler

Performance Modeling Panel at SC'11

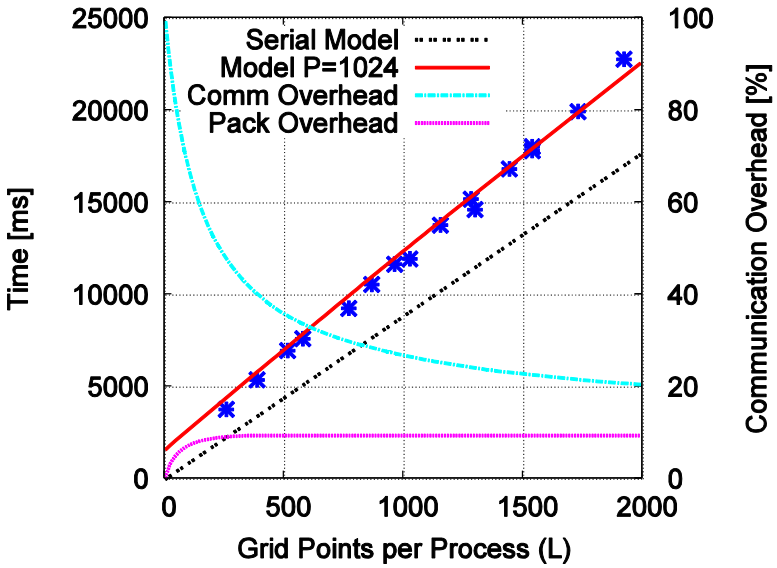
NCSA

NSF

GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION

Model-guided Optimization - Motivation

- Parallel application performance is complex
  - Often unclear how optimizations impact performance
- Issue for applications at large-scale
  - Models can guide optimizations
- One of our models 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

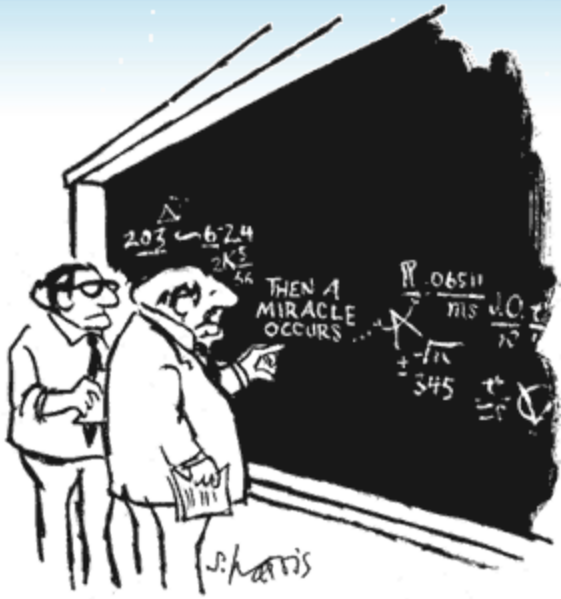


What is Performance Modeling

- Representing performance with **analytic** expressions
  - Not just series of points from benchmarks
  - Enables 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

- Combine analytical methods and 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*



Other 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

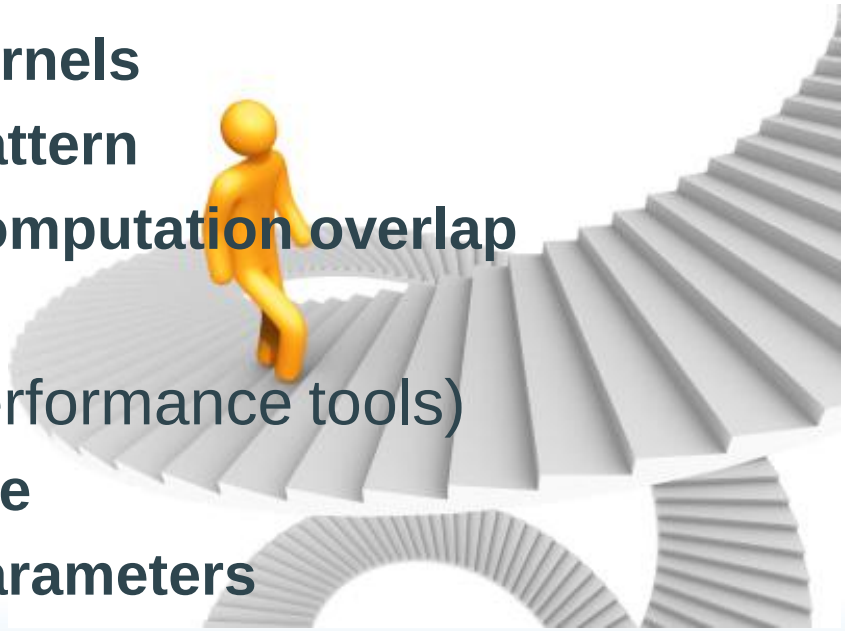
When and where should it be used?

- 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 expensive as correctness bugs!

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**

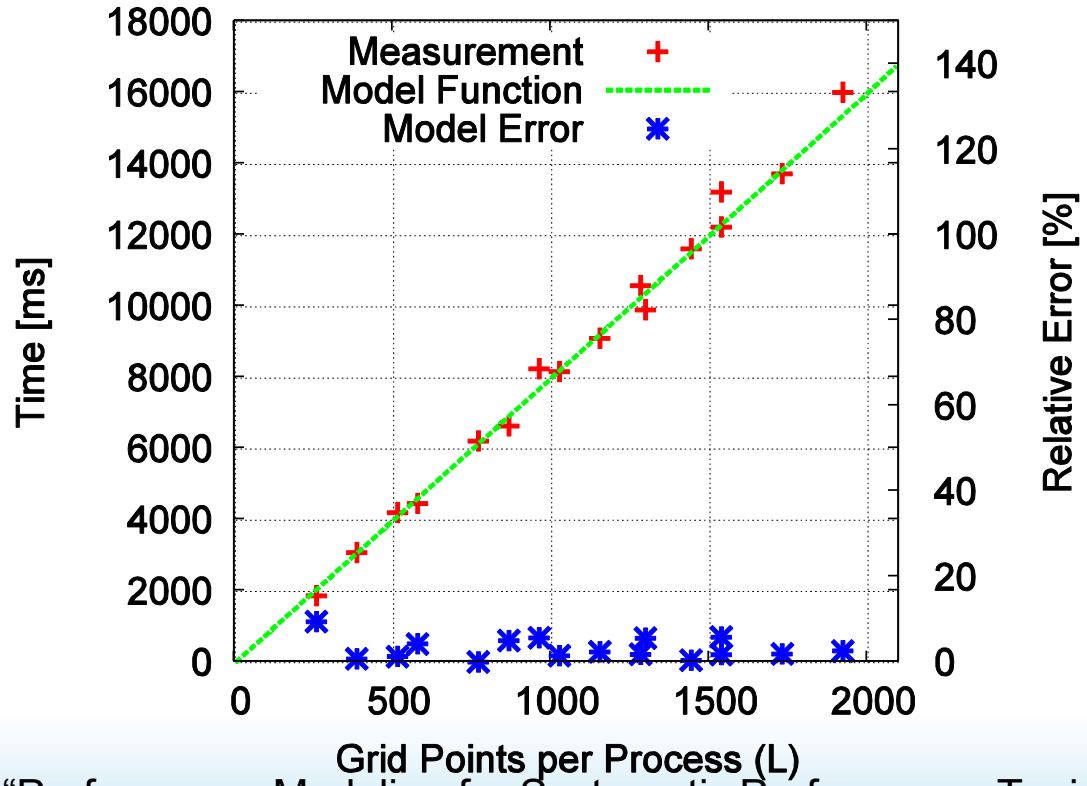


- Empirical steps (benchmarks/performance tools)
  1) determine **sequential baseline**
  2) determine **communication parameters**

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

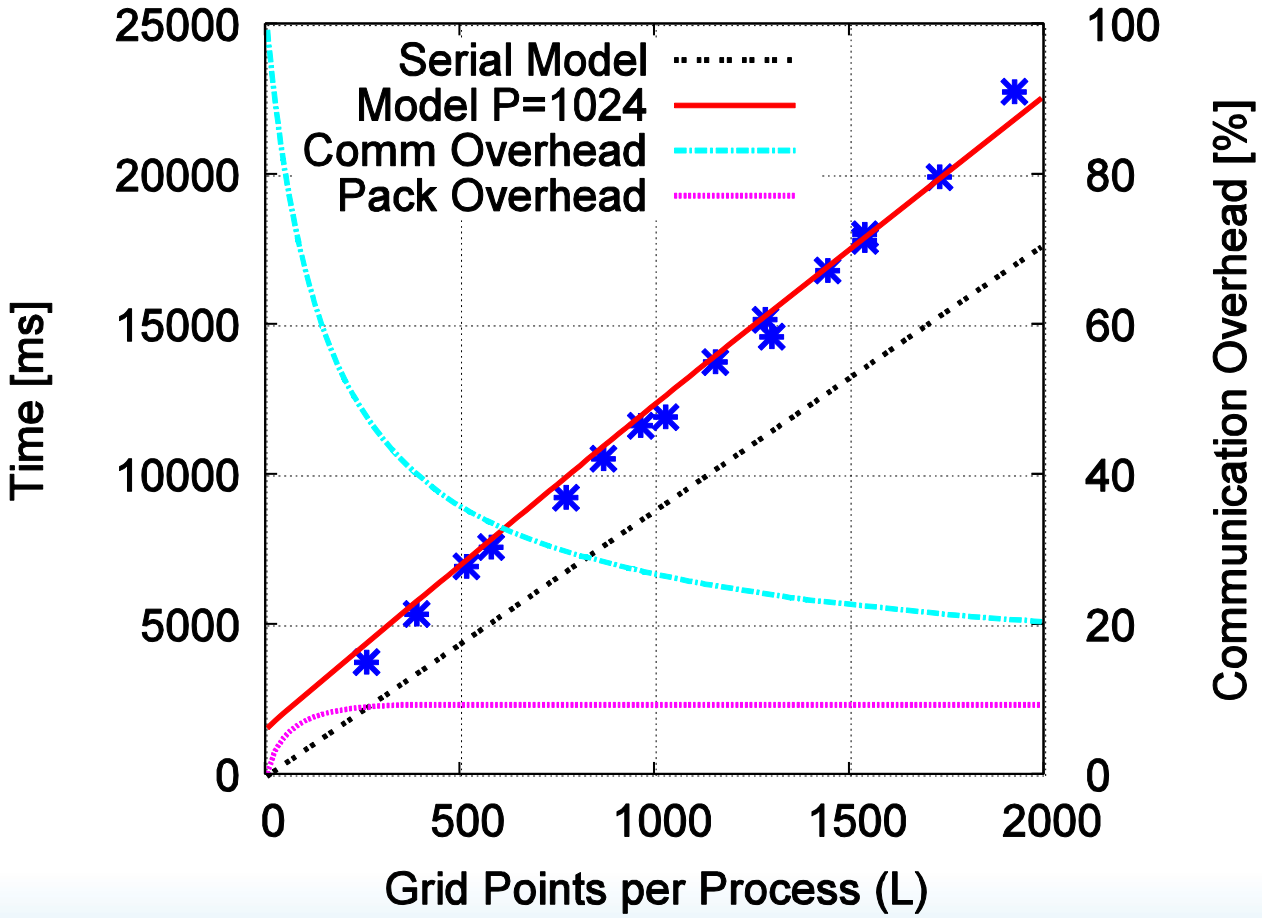
Example Serial Model: MILC

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



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

Example Parallel Model: MILC



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

Conclusions

- We advocate performance modeling as tool for
  - Increasing performance
  - Guide application design and tuning
  - Guide system design and tuning
  - **Throughout the whole software development process!**

- 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

All used images belong to the owner/creator!