Model-Driven, Performance-Centric HPC Software and System Design and Optimization

Torsten Hoefler

With contributions from: William Gropp, William Kramer, Marc Snir

Scientific talk at Jülich Supercomputing Center
April 8th Jülich, Germany
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!
Imagine …

• … you’re designing a hardware to achieve $10^{18}$ operations per second …
• … to run at least some number of scientific applications at scale …
• … and everybody agrees that the necessary tradeoffs make it nearly impossible …
• … where pretty much everything seems completely flexible (accelerators, topology, etc.) …
... and all you have (now) is ...

- ... how do you determine what the system needs to perform at the desired rate?
- ... how do you find the best system design (CPU architecture and interconnection topology)?
State of the Art in HPC – A General Rant 😊

- Of course, nobody planned ahead 😊
- Performance debugging is purely empirical
  - Instrument code, run, gather data, reason about data, fix code, lather, rinse, repeat
- Tool support is evolving rapidly though!
  - Automatically find bottlenecks and problems
  - Usually done as black box! (no algorithm knowledge)
- Large codes are developed without a clear process
  - Missing development cycle leads to inefficiencies
Performance Modeling: State of The Art!

• Performance Modeling (PM) is done ad-hoc to reach specific goals (e.g., optimization, projection)
• But only for a small set of applications (the manual effort is high due to missing tool support)
• Payoff of modeling is often very high!
  • Led to the “discovery” of OS noise [SC03]
  • Optimized communication of a highly-tuned (assembly!) QCD code [MILC10] → >15% speedup!
  • Numerous other examples in the literature

[SC03]: Petrini et al. “The Case of Missing Supercomputer Performance …”
[MILC10]: Hoefler, Gottlieb: “Parallel Zero-Copy Algorithms for Fast Fourier Transform …”
Performance Optimization: State of the Art!

- Two major “modes”:
  1. Tune until performance is sufficient for my needs
  2. Tune until performance is within X% of optimum
- Major problem: what is the optimum?
  - Sometimes very simple (e.g., Flop/s for HPL, DGEMM)
  - Most often not! (e.g., graph computations [HiPC’10])
- Supercomputers can be very expensive!
  - 10% speedup on Blue Waters can save millions $$$
  - Method (2) is generally preferable!

[HiPC’10]: Edmonds, Hoefler et al.: “A space-efficient parallel algorithm for computing Betweenness Centrality …”

8
Ok, but what is this “Performance” about?

- Is it Flop/s?
  - Merriam Webster “flop: to fail completely”
- HPCC: MiB/s? GUPS? FFT-rate?
  - Yes, but more complex
  - Many (in)dependent features and metrics
    - network: bandwidth, latency, injection rate, …
    - memory and I/O: bandwidth, latency, random access rate, …
    - CPU: latency (pipeline depth), # execution units, clock speed, …
- Our very generic definition:
  - Machine model spans a vector space (feasible region)
  - Each application sits at a point in the vector space!
Example: Memory Subsystem (3 dimensions)

- Each application has particular coordinates

![Diagram showing memory subsystem with axes for latency, bandwidth, and injection rate. The diagram illustrates different applications with specific coordinates: Application A is in the lower right quadrant, indicating regular mesh computations; Application B is in the upper left quadrant, indicating highly irregular mesh computations. Some graph or “informatics” applications are also illustrated.](image-url)
Our Practical and Simple Formalization

- Machine Model spans n-dimensional space $\Gamma = (p_1, p_2, \ldots, p_n)$
  - Elements are rates or frequencies (“operations per second”)
  - Determined from documentation or microbenchmarks
    - Netgauge’s memory and network tests [HPCC’07, PMEO’07]
- Application Model defines requirements $\tau = (r_1, r_2, \ldots, r_n)$
  - Determined analytically or with performance counters
  - Lower bound proofs can be very helpful here!
    - e.g., number of floating point operations, I/O complexity
  - Time to solution (“performance”): $\max_{0 < i \leq n} (r_i / p_i)$

[PMEO’07]: Hoefler et al: "Low-Overhead LogGP Parameter Assessment for Modern Interconnection Networks"
Should Parameter X be Included or Not?

- The space is rather big (e.g., ISA instruction types!)

Benchmark --- Full Simulation --- Model Simulation --- Model

Number of Parameters

Model Error

- Apply Occam’s Razor wherever possible!
  - Einstein: “Make everything as simple as possible, but not simpler.”
- Generate the simplest model for our purpose!
  - Not possible if not well understood, e.g., jitter [LSAP’10,SC10]
A Pragmatic Example: The Roofline Model

- Only considers memory bandwidth and floating point rate but is very useful to guide optimizations! [Roofline]
  - Application model is “Operational Intensity” (Flops/Byte)

The Roofline Model: Continued

• If an application reaches the roof: good!
• If not …
  • … optimize (vectorize, unroll loops, prefetch, …)
  • … or add more parameters!
    • e.g., graph computations, integer computations
• The roofline model is a special case in the “multi-dimensional performance space”
  • Picks two most important dimensions
  • Can be extended if needed!

Caution: Resource Sharing and Parallelism

- Some dimensions might be “shared”
  - e.g., SMT threads share ALUs, cores share memory controllers, …
- Needs to be considered when dealing with parallelism (not just simply multiply performance)
  - Under investigation right now, relatively complex on POWER7
How to Apply this to Real Applications?

1. Performance-centric software development
   - Begin with a model and stick to it!
   - Preferred strategy, requires re-design

2. Analyze and model legacy applications
   - Use performance analysis tools to gather data
   - Form hypothesis (model), test hypothesis (fit data)
Performance-Centric Software Development

- Introduce Performance Modeling to all steps of the HPC Software Development Cycle:
  - Analysis (pick method, PM often exists [PPoPP’10])
  - Design (identify modules, re-use, pick algorithms)
  - Implementation (code in C/C++/Fortran - annotations)
  - Testing (correctness and performance! [HPCNano’06])
  - Maintenance (port to new systems, tune, etc.)

[HPCNano’06]: Hoefler et al.: “Parallel scaling of Teter's minimization for Ab Initio calculations”
[PPoPP’10]: Hoefler et al.: "Scalable Communication Protocols for Dynamic Sparse Data Exchange"
Tool 1: Performance Modeling Assertions

• Idea: The programmer adds model annotations to the source-code, the compiler injects code to:
  • Parameterize performance models
  • Detect anomalies during execution
  • Monitor and record/trace performance succinctly
• Has been explored by Alam and Vetter [MA’07]
  • Initial assertions and potential has been demonstrated!

Tool 2: Middleware Performance Models

- Algorithm choice can be complex
  - Especially with many unknowns, e.g.,
    - performance difference between reduce and allreduce?
  - scaling of broadcast, it’s not $O(S \log_2(P))$
- Detailed models can guide early stages of software design but such modeling is hard
  - See proposed MPI models for BG/P in [EuroMPI’10]
  - Led to some surprises!

[EuroMPI’10]: Hoefer et al.: “Toward Performance Models of MPI Implementations …”
Example: Current Point-to-Point Models

- Asymptotic (trivial): \( \Theta(S') \)
- Latency-bandwidth models: \( T = \alpha + S\beta \)
- Need to consider different protocol ranges
- Exact model for BG/P:

\[
T(S') = \begin{cases} 
4.5\mu s + 2.67\text{ns}/B \cdot S : & S \leq 256B \\
5.7\mu s + 2.67\text{ns}/B \cdot S : & 256B < S \leq 1024B \\
9.8\mu s + 2.67\text{ns}/B \cdot S : & 1024B < S 
\end{cases}
\]

- Used Netgauge/logp benchmark
- Three ranges: small, eager, rendezvous

[EuroMPI’10]: Hoeffer et al.: “Toward Performance Models of MPI Implementations …”
Example: Point-to-Point Model Accuracy

- Looks good, but there are problems!
Example: The not-so-ideal (but realistic) Case I

- Strided data-access (p2p model assumed stride-1)

Benchmark: Netgauge: one_one_dtype, 16 kiB MPI_CHAR data
Example: The not-so-ideal (but realistic) Case II

- Matching queue overheads (very common)

![Graph showing latency vs. number of outstanding requests]

- R requests: $T_{match}(R) \leq 100\, ns \cdot R; \quad T(13) \geq 1.3\, \mu s$
- Benchmark: Netgauge/one_one_req_queue

[EuroMPI’10]: Hoefler et al.: “Toward Performance Models of MPI Implementations …”
Example: The not-so-ideal (but realistic) Case III

- Congestion is often ignored
  - Very hard to determine but worst-case can be calculated (assuming rectangular 3D Torus on BG/P)
- Effective Bisection Bandwidth
  - Average bandwidth of a random perfect matching
- Upper bound is congestion-less (see p2p model)
- Lower bound assumes worst-case mapping
  - Assume ideal adaptive routing (BG/P)
  - Congestion of $O\left(\sqrt[3]{P}\right)$ per link
Example: Worst-case vs. Average-case Congestion

- Average seems to converge to worst-case (large P)
- Benchmark: Netgauge/ebb
Tool 3: Modeling for Legacy Applications

- Current programming models don’t support performance modeling well
  - Performance analysis tools to gather data
  - Costly manual analysis
- Automatic modeling tools?
  - Detection of regions
    - changes in IPC
  - Example: MILC, detect five “critical regions”, same result as manual modeling

Data collected with NCSA perfsuite/papi
Performance-centric Software Development

- Performance models allow to explain application performance
  - Find problems, not a solutions
  - Mostly a scientific exercise to understand
- Integrate modeling and the programming model to allow performance-centric design
  - Understand and avoid problems by design
  - Structured approach to “Performance Engineering”
Tool 1: Performance-transparent Abstractions

- **Abstractions** allow for performance portability and ease of programming!
  - How to choose an abstraction? What to expect?
  - Determine application requirements! → PM
  - e.g., nonblocking collectives, sparse collectives
- Trade-off between performance, portability, and programmability is most important!
- Performance must be **first class citizen** in HPC programming models (yet it isn’t!)

[PPL]: Balaji, Hoefler et al.: "MPI on Millions of Cores", [SciDAC’10] "MPI at Exascale"
[SC07]: Hoefler et al.: "Implementation and Performance Analysis of Non-Blocking Collective Operations for MPI"
[PPoPP’11,ISC’11]: Willcock, Hoefler et al. “Active Pebbles: Parallel Programming for Data-Driven Applications”
Tool 2: Model-driven Topology Mapping

- Can optimize performance significantly, nearly no impact on programmability (MPI-2.2 [CCPE])!
  - Computing a mapping is expensive!
  - Scalable algorithms in [ICS’11]

---

[ICS’11]: Hoefler and Snir: Generic Topology Mapping Strategies for Large-Scale Parallel Architectures

[CCPE]: Hoefler et al.: "The Scalable Process Topology Interface of MPI 2.2"
Tool 3 (Idea): Power-aware programming?

- Provide models and abstractions for power usage
  - Mostly data-movement centric
  - Flops-metric is not predictive for energy consumption
- But: performance and energy consumption correlate (finish faster = use less power)
  - detailed analysis for networks in [CiSE’10]

[CiSE’10]: Hoefler: “Software and Hardware Techniques for Power-Efficient HPC Networking”
[CAC’09]: Hoefler et al.: “A Power-Aware, Application-Based, Performance Study Of … Networks”
Tool 4: Model-guided System Design

- Systems and Applications need to evolve in parallel
  - Applications need to be ready when a machine goes online!
  - Co-design is attractive, models as “communication medium”
- Application-specific interconnection optimization:
  - Optimized general routing [IPDPS’11]
  - Application-specific routing
  - Novel topologies [HotI’10]
  - Reconfigurable architectures or topologies

[IPDPS’11]: Domke, Hoefler, Nagel: “Deadlock-Free Oblivious Routing for Arbitrary Topologies“
[HotI’10]: Arimilli, Hoefler et al.: "The PERCS High-Performance Interconnect"
Summarizing the Big Picture

• Develop performance modeling as a science discipline
  • Observation, measurement, hypothesis, test
  • Enables us to explain application performance

• Foster wide adoption of modeling techniques
  • Establish methodology, provide tool support
  • Static applications work, many open problems though

• Transform results into an engineering discipline
  • Not only explain performance but indicate how to program or tune code for best performance
References to Previous Work

[IPDPS'11]: Domke, Hoefler, Nagel: "Deadlock-Free Oblivious Routing for Arbitrary Topologies"
[PPL]: Balaji, Hoefler et al.: "MPI on Millions of Cores", [SciDAC'10] "MPI at Exascale"
[SIAM-CSE'10]: Gropp, Hoefler, Snir: "Performance Modeling for Systematic Performance Tuning"
[PROPER'10]: Hoefler: "Bridging Performance Analysis Tools and Analytic Performance Modeling"
[SC10]: Hoefler et al.: "Characterizing the Influence of System Noise … by Simulation" (Best Paper)
[CCPE]: Hoefler et al.: "The Scalable Process Topology Interface of MPI 2.2"
[HotI'10]: Arimilli, Hoefler et al.: "The PERCS High-Performance Interconnect"
[LSAP'10]: Hoefler et al.: "LogGOPSim – Simulating … Apps. in the LogGOPS Model" (Best Paper)
[PPoPP'10]: Hoefler et al.: "Scalable Communication … for Dynamic Sparse Data Exchange"
[PMEO'07]: Hoefler et al: "Low-Overhead LogGP Parameter Assessment …"
[HPCC'07]: Hoefler et al: "Netgage: A Network Performance Measurement Framework"
[SC07]: Hoefler et al.: "Implementation and Performance Analysis of Non-Blocking Collective Operations for MPI“
[HPCNano'06]: Hoefler et al.: “Parallel scaling of Teter's minimization for Ab Initio calculations”