Performance Modeling as the Key to Extreme Scale Computing
William Gropp
www.cs.illinois.edu/~wgropp

Performance is Key
• Parallelism is (usually) used to get more performance
How do you know if you are making good (not even best) use of a parallel system?
• Even measurement-based approaches can be (and often are) performed without any real basis of comparison
The key questions are
• Where is most of the time spent?
• What is the achievable performance, and how do I get there?

Tuning A Parallel Code
Typical Approach
• Profile code: Determine where most time is being spent
• Study code: Measure absolute performance, look at performance counters, compiler FLTP rates
• Improve code that takes a long time; reduce time spent
• Spent

How Do We Know if there is a Performance Problem?
• My application scales well!
So what!
• Is it efficient?
Making the scalar code more efficient decreases scalability
• How can we know?
To what do we compare?

Tuning A Parallel Code
Why is it hard to know?
Improvement you can obtain?
How do you know how much performance you know when you are done?
Why this isn’t the right approach:
• Spent in “unproductive” operations
• Improves code that takes a long time; reduce time
• Performance counters, compiler FLTP rates

Why is it hard to know?
How can we know?
To what do we compare?
• Decrease scalability
Making the scalar code more efficient?
• Is it efficient?
So what!
• My application scales well!

Parallel Computing
WmW.cs.illinois.edu/~wgropp
William Gropp
Performance Modeling as the Key to Extreme Scale Computing
Heart of Blue Waters: Two New Chips

Quad-chip Module (QCM)
- Four Power7 chips
- 32 cores, 128 threads
- ~1 TF peak performance
- 128 GBytes memory
- 1.128 TB/sec bandwidth

One Hub Chip
- 128 GBytes memory bandwidth
- 40 GBytes/s general purpose
- 128 GB/s to other QCMs in drawer
- 256 GB/s to other QCMs in same drawer
- 384 GB/s to other QCMs in drawers in other cabinets
- Same drawer same QCM connection
- 192 GBytes/s general purpose

IH Server Node (Drawer) and Supernode
- 4 IH Server Nodes
- Up to 32 Tflops
- 4 TBytes of memory
- 16 TB/s memory bandwidth
- 32 Hub Chips
- 36 TB/sec/Hub

Specifications:
- 8 Hub Chips
- 1 TByte of memory
- 300 IOPS
- 384 GB/s local bandwidth
- 320 GB/s to QCMs in other drawers in same supernode
- 40 GB/s to QCMs in other supernodes

Packaging:
- 2U Drawer
- 39" w x 72" d
- > 300 lbs.
- Fully water cooled

Logical View of Blue Waters

Interconnect
- Full direct connectivity
- Local links within drawers (336 GB/s)
Another Example System

Two-level (L, D) Direct-connect Network

Blue Waters = 320 Supernodes
Fully Interconnected with D Links
(320 BBs x 8 SNs/BB)
Result: Very high bandwidth

Hardware Latency
 attorneys
Local and Remote Links

320 node GPU Cluster
• #3 on Green500
• 238 node GPU Cluster
• Two-level (L, D) Direct-connect

Future

Logical View of Blue Waters
Interconnect

Logical View of Blue Waters
Interconnect

Logical View of Blue Waters
Interconnect

Logical View of Blue Waters
Interconnect
An Even More Radical System

• Rack Scale
  Processor: 128 Nodes, 1 (+) PF/s
  Memory:
  • 128 TB DRAM
  • 0.4 PB/s Aggregate Bandwidth

Why Performance Modeling?

• What is the goal?
  It is not precise predictions
  It is not precise predictions
  It is not precise predictions
  Performance modeling can identify

Performance Models

Different Philosophies for Performance Models

• Simulation:
  Very accurate prediction, little insight

• Traditional Performance Modeling (PM):
  Very accurate prediction, little insight
  To identify the critical resource
  To nonlinear change to the code
  To estimate the potential benefit of a nontrivial change
  Performance modeling can be used
  and if not, how to fix it

• What is the goal?

Traditional Performance Modeling (PM):

To make precise predictions

Simulated in development environment

Performance can identify

Performance Models

Why this sort of modeling

Performance benchmarks are not a performance model

Note that a series of measurements from application's algorithm and data structures

First, an analytic expression based on the code

Second, an analytic expression based on the application

Actually two different models

Performance Engineering
Our Methodology

- Combine analytical methods and performance measurement tools
- Programmer specifies parameterized expectation (e.g., \( T = a + b \cdot N^3 \))
- Estimate coefficients with appropriate benchmarks
- We derive the scaling analytically and fill in the constants with empirical measurements
- Focus on upper and lower bounds rather than precise constants
- Models must be as simple and effective as possible
- Simplicity increases the insight
- Precision needs to be just good enough to drive action
- Simplicity allows for deeper insight
- Models must be as simple and effective as possible
- Precision needs to be just good enough to drive action
- Simplicity increases the insight

Sparse Matrix-Vector Product

- Common operation for optimal (in floating-point operations) solution
- Sample code (common CSR format):
  ```
  for row=1,n
    \( y[i] = \text{sum} \) -= \text{sum} = \text{sum} += a \cdot x[j] for row=1,nnz
  ```
- Data structures are \([n,n], [n], [nnz], [nnz] \)
- Memory motion: \( \text{nnz} \cdot \text{sizeof(double)} + \text{sizeof(int)} + \text{n} \cdot \text{sizeof(double)} + \text{sizeof(int)} \)
- Computation: \( \text{nnz} \cdot \text{multiply-add (MA)} \)
- Roughly 12 bytes per MA
- Typical node can move 1-4 bytes/MA
- Maximum performance is 8-33% of peak
- Use STREAM benchmark to get sustained memory bandwidth
- Maximum performance is 8-33% of peak
- \( \text{nnz} \cdot \text{multiply-add (MA)} \)
- Assume a perfect cache (never load same data twice)

Realistic Measures of Peak Performance

- One vector: matrix size \( m = 90,708 \), nonzero entries \( nnz = 5,047,120 \)

Sparse Matrix Vector Product

- Sparse Matrix Vector Product

- Sparse matrix-vector multiply
- Precision needs to be just good enough to drive action
- Simplicity increases the insight
- Models must be as simple and effective as possible
- Precision needs to be just good enough to drive action
- Simplicity increases the insight
- Models must be as simple and effective as possible
- Focus on upper and lower bounds rather than precise constants
- We derive the scaling analytically and fill in the constants
- Estimate coefficients with appropriate benchmarks
- \( T = a + b \cdot N^3 \)
- Programmer specifies parameterized expectation
- Combine analytical methods and performance measurement tools

Simpler Performance Analysis

- An example: Sparse matrix-vector multiply
- Precision needs to be just good enough to drive action
- Simplicity increases the insight
- Models must be as simple and effective as possible
- Focus on upper and lower bounds rather than precise constants
- We derive the scaling analytically and fill in the constants
- Estimate coefficients with appropriate benchmarks
- \( T = a + b \cdot N^3 \)
- Programmer specifies parameterized expectation
- Combine analytical methods and performance measurement tools
The problem is so big!

Real applications are much larger – isn’t it hard to do this for the entire application?

Yes, but it doesn’t matter for runnable apps. Look at the parts that take the most time. Break the problem into digestible parts.

Note rapidly growing numbers of functional units.

Contributions to performance issues from:

How Good are Compilers at Vectorizing Codes?


Media Bench II Applications

<table>
<thead>
<tr>
<th>App</th>
<th>XLIC</th>
<th>ILIC</th>
<th>GCC</th>
<th>XLIC</th>
<th>ICC</th>
<th>GEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manual</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JPEG Enc</td>
<td>1.33</td>
<td>1.39</td>
<td>1.45</td>
<td>1.26</td>
<td>2.28</td>
<td>2.06</td>
</tr>
<tr>
<td>JPEG Dec</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.14</td>
<td>1.13</td>
</tr>
<tr>
<td>H263 Enc</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.25</td>
<td>2.28</td>
<td>2.06</td>
</tr>
<tr>
<td>H263 Dec</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.33</td>
<td>1.39</td>
<td>1.57</td>
</tr>
<tr>
<td>MPEG2 Enc</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.39</td>
<td>2.33</td>
<td>1.57</td>
</tr>
<tr>
<td>MPEG2 Dec</td>
<td>1.19</td>
<td>1.57</td>
<td>1.75</td>
<td>1.45</td>
<td>2.13</td>
<td>1.57</td>
</tr>
<tr>
<td>MPEG2 End</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MPEG4 Enc</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.44</td>
<td>2.28</td>
<td>2.06</td>
</tr>
<tr>
<td>MPEG4 Dec</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.12</td>
<td>-</td>
<td>1.18</td>
</tr>
</tbody>
</table>

Whole program speedup measured against:

Utilizing the Processor

- Note rapidly growing numbers of functional units. A has 4, access through "vector" instructions. B/C has 2 multiply-add units per core. B/C/P has 2 multiply-add units per core.

But the problem is so big!
Processes and Memory

• For many computations, sustained memory performance is the limiting resource as in sparse matrix-vector multiply.

• What is the appropriate sustained rate?
  Memory bus bandwidth is nearly irrelevant—it is the sustained rate that is usually important.

• What about other ways to increase effective sustained performance, such as prefetch?
  Prefetch hardware can detect regular accesses and prefetch data, making use of otherwise idle memory bus time. However, the hardware must be presented with enough independent data streams so that the prefetch hardware can detect regular accesses. Prefetching performance is usually important only if the hardware is not already doing it.

Performance Ratio Compared to CSR Format

• S-CSR format is better than CSR format for all (on Power 6) or most (on Power 4 and 5) matrices.

Combining With Other Optimizations

• We can further modify the S-CSR format to match the requirements for vectorization.
  • We can use OSKI to optimize "within the loops" to optimize vectorization requirements for and S-BCSR to modify the S-CSR format.

Streamed Compressed Sparse Row (S-CSR) Format

• S-CSR format partitions the sparse matrix into blocks along rows with size of \(bs\). Zeros are added in to keep the number of elements the same in each row.
  For the sample matrix in the following figure, \(NNZ = 29\). Using a block size of \(bs = 4\), it generates four equal length streams R, G, B and P. This new design only adds 7 zeros every 4 rows. Design only adds 7 zeros every 4 rows.

For the sample matrix in the following figure, \(NNZ = 29\). Using a block size of \(bs = 4\), it generates four equal length streams R, G, B and P. This new design only adds 7 zeros every 4 rows.

• As in sparse matrix-vector multiply, performance is the limiting resource for many computations, sustained memory.
HPC users typically believe that their code “owns” all of the cores all of the time. The reality is that it was never true, but they did have all of the cores the same fraction of time when there was one core/node.

We can use a simple performance model to check the assertion and then use measurements to identify the problem and suggest fixes.

Consider a simple Jacobi sweep on a regular mesh, with every core having the same amount of work. How are run times distributed?

Data copies and MPI datatypes

Impacted choice of communication algorithm (many independent/communicating links)

For Blue Gene, must model the real system, not abstractly

Parameters?

What are the correct bandwidth and real system parameters?

How relevant is ping-pong bandwidth and real systems?

Application is MILC, a lattice QCD code

Application is model-guided optimization

MILC was the only case where performance improvements on the Blue Gene were greater than 10-16%.

Recent results give 10-16% performance improvements by eliminating the pack before communicating the pack.

Careful mixed strategies are even better.

Pure dynamic scheduling adds overhead, but is better.

Careful mixed strategies are even better.

Recent results give 10-16% performance improvements on large, scalable systems.

Thanks to Vivek Kale

Model-guided Optimization

Process and the Network

Investigating use of non-blocking collectives in a model-driven ICC

Processes and SMP nodes

Processes and SMP nodes

Having many cores available makes everyone think that they can use them to solve other problems (even if there are none).

Having many cores available makes everyone think that they can use them to solve other problems (even if there are none).

The reality is that was never true, but they did have all of the cores all of the time.

"owns" all of the cores all of the time.

HPC users typically believe that their code
AMG Performance Model

One common case is a halo exchange...

- One common culprit is the mapping of processes to physical topology.
- We have used modeling to determine that a certain kind of random mapping is often preferable for Blue Waters.
- Avoiding hot-spots on two-level direct networks.

Not Just Collectives

- Often the real problem is imbalance.
- Hypothesis is that there is load imbalance (more, less, etc.).
- Test using FPMPI2.

Halo Exchange on BG/P and Cray XT4

- 2048 doubles to each neighbor.
- BG/P 4 Neighbors 8 Neighbors
  - Irecv/Send Irecv/Send Irecv/Send Irecv/Send
  - World 208 328 184 237 Even/Odd 219 327 172 243
  - Cart_create 301 581 242 410

- Cray XT4 4 Neighbors 8 Neighbors
  - Irecv/Send Irecv/Send Irecv/Send Irecv/Send
  - World 311 306 331 262 269
  - Even/Odd 257 247 279 212 206
  - Cart_create 265 275 266 236 232

MPI Communication is too Slow

- How often do you hear "MPI Communication is too Slow"?

Lets look at a single process sending to its neighbors. Based on our performance model, we expect the rate to be roughly twice that for the halo (since this test is only sending, not sending and receiving). We see in Table 1:

<table>
<thead>
<tr>
<th>System</th>
<th>X14 SN</th>
<th>X14</th>
<th>X13</th>
<th>X13 SN</th>
<th>X13</th>
<th>X12</th>
<th>X12 SN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG/L</td>
<td>1808</td>
<td>1811</td>
<td>1710</td>
<td>1704</td>
<td>1703</td>
<td>1702</td>
<td>1701</td>
</tr>
<tr>
<td>BG/L VN</td>
<td>1604</td>
<td>1603</td>
<td>1602</td>
<td>1601</td>
<td>1600</td>
<td>1600</td>
<td>1600</td>
</tr>
<tr>
<td>BG/P VN</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
</tr>
<tr>
<td>BG/P</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
</tr>
</tbody>
</table>

XT4 gives roughly double the halo rate. It should be possible to improve the halo exchange on the XT by shortening the message transfer time. We see in Table 1:

<table>
<thead>
<tr>
<th>System</th>
<th>X14 SN</th>
<th>X14</th>
<th>X13</th>
<th>X13 SN</th>
<th>X13</th>
<th>X12</th>
<th>X12 SN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG/L</td>
<td>1808</td>
<td>1811</td>
<td>1710</td>
<td>1704</td>
<td>1703</td>
<td>1702</td>
<td>1701</td>
</tr>
<tr>
<td>BG/L VN</td>
<td>1604</td>
<td>1603</td>
<td>1602</td>
<td>1601</td>
<td>1600</td>
<td>1600</td>
<td>1600</td>
</tr>
<tr>
<td>BG/P VN</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
</tr>
<tr>
<td>BG/P</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
</tr>
</tbody>
</table>

XT4 gives roughly double the halo rate. It should be possible to improve the halo exchange on the XT by shortening the message transfer time. We see in Table 1:

<table>
<thead>
<tr>
<th>System</th>
<th>X14 SN</th>
<th>X14</th>
<th>X13</th>
<th>X13 SN</th>
<th>X13</th>
<th>X12</th>
<th>X12 SN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG/L</td>
<td>1808</td>
<td>1811</td>
<td>1710</td>
<td>1704</td>
<td>1703</td>
<td>1702</td>
<td>1701</td>
</tr>
<tr>
<td>BG/L VN</td>
<td>1604</td>
<td>1603</td>
<td>1602</td>
<td>1601</td>
<td>1600</td>
<td>1600</td>
<td>1600</td>
</tr>
<tr>
<td>BG/P VN</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
<td>268</td>
</tr>
<tr>
<td>BG/P</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
<td>176</td>
</tr>
</tbody>
</table>

Ratios of a single sender to all processes sending (in rate):

- X14: 1.76
- X13: 1.07
- X12: 1.76
- X12 SN: 1.07

Summary

- Isn't this just a collection of tricks?
- Yes and no
  - Yes, a number of different approaches have been applied
  - No, the same quantitative approach, based on getting performance estimates for the resources under consideration and emphasizing a simple model that estimates bounds, is applied

Quantitative Thinking

- Must be based on having a hypothesis (model), not just measurements...
Why is Performance Modeling the Key to Extreme Scale?

- Measuring yesterday’s applications, even with today’s runtimes, is often irrelevant.
- Architectures are changing rapidly.
- Further reduces value of measurements on existing codes.
- Focus on achievable performance at scale.
- Architectures are changing rapidly.
- Only way to evaluate radical (and necessary?) benefit to a major change.
- Models permit quantitative evaluation of different approaches and a priori estimation of possible benefit to major change.
- Models permit quantitative evaluation of different approaches and estimation of performance of different codes.
- Only way to evaluate radical (and necessary?) benefit to a major change.

Thanks