An Introductory Exascale Feasibility Study for FFTs and Multigrid

Hormozd Gahvari
William Gropp

University of Illinois at Urbana-Champaign

April 22, 2010
Outline

1. Exascale basics
2. Studying application feasibility
3. FFT study
4. Multigrid study
5. Conclusions and directions for future work
Exascale Basics

- Exascale means $10^{18}$ operations per second
- Exascale machines expected to have between 100 million and 1 billion cores
- Use of new technologies and perhaps novel architectures also expected
- Big impact on applications anticipated
Main challenge: specific design and machine parameters are far from known, so no straightforward plugging numbers into performance models.

Instead, treat machine parameters like latency and bandwidth as variables and see what range of values for them would be feasible, i.e., what kind of machine would need to be built to enable exascale performance?

Model on following “hypothetical exascale machine:”

- $2^{28} \approx 268.5$ million cores
- Time per flop $t_c = 10^{-10}$ seconds
- Peak performance: 2.68 EFLOPS

Also vary problem sizes.
Use LogP performance model to model performance. Parameters are:

- $L$ – latency for communicating on one link
- $o$ – software overhead incurred in communication
- $g$ – gap between messages
- $P$ – number of processors

We use LogP rather than a more detailed model because:

1. A model that assumes more details about the architecture restricts the results to a certain class of machines
2. We are looking for bounds, not specific predictions (which we cannot make for a machine that has yet to be built!). LogP which ignores complicating factors like congestion can give us a good starting point

For each application, model performance and see the region in parameter space in which exascale performance is achieved.
Scalability challenge: requires collective communication
Past work has managed the cost by using either optimized collective communication routines or aggressive overlap of communication and computation
Is the communication cost still manageable at exascale?
We consider a 3D FFT on a cubic domain of $N = n^3$ points.

Two ways of partitioning: *slabs* (left), and *pencils* (right):

- 2D then 1D local FFTs (2 rounds)
  - One round communication
  - Min. computation time: *decades*

- 1D local FFTs (3 rounds)
  - Two rounds communication
  - Min. computation time: *milliseconds*

We consider only pencils decomposition. Assume $P = p \times p$
No overlap model:

\[ T = t_c \frac{N}{P} \log_2 N + 2(p - 1)(L + o) + 2(p - 2)g \]

Latency is treated as cost to send entire message, so no latency-hiding done here.

Overlap model: pipeline computation and communication using LogGP model, which extends LogP with an inverse bandwidth term \( G = \text{gap between units of data} \). Assuming computation and communication of one \( n \times \frac{n}{p} \) sheet at a time, we get this \( \left( \frac{n}{p} + 1 \right) \)-stage pipeline (only 3 stages shown here for simplicity):
Graph on left shows feasibility regions in $L$ and $g$ for two different problems under two different situations, one where software overhead was zero (dotted line) and the other where it was 1 ns (solid line). Graph on right shows feasibility regions for several problem sizes for the “real-world” case:

These graphs show that latency and gap have to be small unless problem is large
Overlap enables us to hide latency effectively, but will require GB/s bandwidth to do so. Gap constraint is also more restrictive:

![Feasibility Region for 3D FFT with Overlap, N = 1 x 10^{13}](image1)

![Feasibility Region for 3D FFT with Overlap, N = 1 x 10^{15}](image2)

![Feasibility Region for 3D FFT with Overlap, N = 1 x 10^{17}](image3)

![Feasibility Region for 3D FFT with Overlap, N = 1 x 10^{19}](image4)
Since computation grows superlinearly, a natural question to ask is how big the problem size can grow until it takes too long. It can get pretty big. Here are problem sizes at which FFT computation at the rate of one EFLOP takes at least...

<table>
<thead>
<tr>
<th>Time</th>
<th>No. Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ms</td>
<td>$5.87 \times 10^{13}$</td>
</tr>
<tr>
<td>1 s</td>
<td>$4.84 \times 10^{16}$</td>
</tr>
<tr>
<td>1 minute</td>
<td>$2.63 \times 10^{18}$</td>
</tr>
<tr>
<td>1 hour</td>
<td>$1.44 \times 10^{20}$</td>
</tr>
<tr>
<td>1 day</td>
<td>$3.24 \times 10^{21}$</td>
</tr>
<tr>
<td>1 week</td>
<td>$2.19 \times 10^{22}$</td>
</tr>
</tbody>
</table>
Another question to ask is, given the collective communication, the effect of the interconnect? Can give a performance upper bound as time required to (twice, since there are two communication rounds) move problem data across bisection bandwidth of network. If we treat individual link bandwidth as a variable, we can find a lower bound for it that corresponds to the upper bound being exascale:

<table>
<thead>
<tr>
<th>Interconnect</th>
<th>Bisection BW</th>
<th>$N = 2^{42}$</th>
<th>$N = 2^{59}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Mesh</td>
<td>$\sqrt{P}$</td>
<td>$1.72 \times 10^4$ GB/s</td>
<td>$1.23 \times 10^4$ GB/s</td>
</tr>
<tr>
<td>2D Torus</td>
<td>$2\sqrt{P}$</td>
<td>$8.63 \times 10^3$ GB/s</td>
<td>$6.14 \times 10^3$ GB/s</td>
</tr>
<tr>
<td>3D Mesh</td>
<td>$P^{2/3}$</td>
<td>$680$ GB/s</td>
<td>$484$ GB/s</td>
</tr>
<tr>
<td>3D Torus</td>
<td>$2P^{2/3}$</td>
<td>$340$ GB/s</td>
<td>$242$ GB/s</td>
</tr>
<tr>
<td>Fat-tree</td>
<td>$P/2$</td>
<td>$1.05$ GB/s</td>
<td>$0.75$ GB/s</td>
</tr>
<tr>
<td>Hypercube</td>
<td>$P/2$</td>
<td>$1.05$ GB/s</td>
<td>$0.75$ GB/s</td>
</tr>
</tbody>
</table>
Multigrid Feasibility Study

- Scalability challenge: while communication cost is constant, computation/communication ratio decreases as grids get coarser
- When there are less points than processors, some will sit idle unless special measures are taken
- Under what circumstances will such steps be necessary?
Consider using geometric multigrid applied in V-cycles to perform nearest-neighbor computation such as solution of Laplace equation.

Consider both 2D and 3D versions of computation, with processors arranged in the appropriate mesh network.

Assume the points are distributed evenly among the processors, with an ideal point to processor mapping.

Assume Jacobi smoothing.
Use LogP model like with FFT, but with a slight modification. Treat $L$ as a per-link latency. Once there are fewer points than processors, communication will cross more links, and we want to capture this.

Other model assumptions:

- There are $N$ points, arranged in a $d$-dimensional grid
- Each processor communicates with $k$ neighbors ($k + 1$-point stencil)
- Number of points decreases by a constant factor $c$ in each dimension after coarsening
- We model one V-cycle
Break model into components:

- **smooth**$(n, l)$ – run smoother on $n$ points, with neighbors $l$ links away
- **coarsen**$(l)$ – perform one step of coarsening. Neighbors before coarsening are $l$ links away; this is the distance of communication
- **prolong**$(l)$ – perform one step of prolongation. Neighbors after prolongation are $l$ links away; this is the distance of communication

- Treat direct solve as smoother application and recurse as far as possible for simplicity
Smoothing time is:

\[ T_s = \sum_{i=0}^{\lfloor \log_{cd} \frac{N}{P} \rfloor} \text{smooth} \left( \frac{N}{c^{di} P}, L \right) + \sum_{i=\lfloor \log_{cd} \frac{N}{P} \rfloor + 1}^{\lfloor \log_{cd} N \rfloor} \text{smooth} \left( 1, c^i - \lfloor \log_{cd} \frac{N}{P} \rfloor L \right) \]

Coarsening time is:

\[ T_c = \sum_{i=0}^{\lfloor \log_{cd} \frac{N}{P} \rfloor} \text{coarsen} \left( L \right) + \sum_{i=\lfloor \log_{cd} \frac{N}{P} \rfloor + 1}^{\lfloor \log_{cd} N \rfloor - 1} \text{coarsen} \left( c^i - \lfloor \log_{cd} \frac{N}{P} \rfloor L \right) \]

Prolongation time is:

\[ T_p = \sum_{i=0}^{\lfloor \log_{cd} \frac{N}{P} \rfloor} \text{prolong} \left( L \right) + \sum_{i=\lfloor \log_{cd} \frac{N}{P} \rfloor + 1}^{\lfloor \log_{cd} N \rfloor - 1} \text{prolong} \left( c^i - \lfloor \log_{cd} \frac{N}{P} \rfloor L \right) \]
Applying LogP model gives us, for each component:

\[
\text{smooth}(n, l) = (k + 1)nt_c + k(l + o) + (k - 1)g
\]

\[
\text{coarsen}(l) = k(l + o) + (k - 1)g
\]

\[
\text{prolong}(l) = k(l + o) + (k - 1)g
\]

For our results, we will assume five smoother steps before coarsening and five smoother steps after prolongation back to that grid, with the number of grid points reduced by 2 in each dimension at each coarsening step.
We use performance models to come up with feasibility regions for two cases, 2D 5-point stencil and 3D 7-point stencil. Solid lines are for performance model as described earlier; dotted lines are when latency does not get bigger on coarse enough grids:

We see that a coarse grid penalty for latency makes it a big concern
Since multigrid is only linear time, problem size is not as much of a concern as with the FFT. Here are problem sizes at which multigrid computation at the rate of one EFLOP takes at least...

<table>
<thead>
<tr>
<th>Time</th>
<th>No. Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ms</td>
<td>$2.68 \times 10^{14}$</td>
</tr>
<tr>
<td>1 s</td>
<td>$2.68 \times 10^{17}$</td>
</tr>
<tr>
<td>1 minute</td>
<td>$1.61 \times 10^{19}$</td>
</tr>
<tr>
<td>1 hour</td>
<td>$9.66 \times 10^{20}$</td>
</tr>
<tr>
<td>1 day</td>
<td>$2.32 \times 10^{22}$</td>
</tr>
<tr>
<td>1 week</td>
<td>$1.62 \times 10^{23}$</td>
</tr>
</tbody>
</table>
A slowdown in computation on coarse grids could render exascale performance impossible for small enough problems, e.g. if machine peak is achievable only using hardware such as vector units. Model by adjusting computation rate $t_c$ accordingly. For vector unit of length 64, assuming varying latency:

Solid lines mean $t_c$ varies as well, while dotted lines have $t_c$ fixed. No solid line means that exascale performance is impossible.
Conclusions and Directions for Future Work

- There are substantial constraints to be satisfied to enable exascale performance for FFT and multigrid:
  - Latencies in the nanosecond to microsecond range for smaller FFT problems, or perhaps even smaller for multigrid
  - FFT needs bandwidth on the order of GB/s per process, and a mesh interconnect cannot provide enough bisection bandwidth
  - There is still room for the problem size to grow with the higher processor count, however

- Two main thrusts for future work:
  1. Continue the analysis presented here for other algorithms and applications, to see which ones are suited to exascale systems
  2. Build more depth, looking in more detail than was done here – network contention modeling for FFT and data movement techniques to handle coarse grids in multigrid