ON THE COMMUNICATION COMPLEXITY OF 3D FFTS AND ITS IMPLICATIONS FOR EXASCALE

Kent Czechowski, Chris McClanahan, Casey Battaglino, Kartik Iyer, P.-K. Yeung, and Richard Vuduc
Exascale-ability

**Today**

\[ N = 4096^3 \]

12.3\( \times 10^{12} \) Flops

1.1 TB of Data
Exascale-ability

**Tomorrow**

\[ N = 131,072^3 \]

\[ 0.574 \times 10^{18} \text{ Flops} \]

36 PB of Data

3D FFT
Performance Model
3D FFT Decompositions

- **Slab**: $P < n$
- **Pencil**: $n < P < n^2$
- **Cube**: $n^2 < P$

Problem size $N = n \times n \times n$
Pencil Decomposition

Slab \[ P < n \]

Pencil \[ n < P < n^2 \]

Cube \[ n^2 < P \]

Problem size \( N = n \times n \times n \)
Distributed 3D FFT - Performance Model

3D FFT Using the Pencil Decomposition of the Transpose Method

Computation Phase #1

Communication Phase #1

Computation Phase #2

Communication Phase #2

Computation Phase #3

1D FFT in the X-direction

X → Y Transpose

1D FFT in the Y-direction

Y → Z Transpose

1D FFT in the Z-direction
Distributed 3D FFT - Performance Model

Arithmetic Computation Time

\[ T_{\text{flops}} = 3 \times \frac{n^2}{P} \times \frac{5n \log n}{C_{\text{node}}} \]

Nodes: P
Compute Throughput: \( C_{\text{node}} \)

Memory Access Time

\[ T_{\text{mem}} \approx 3 \times \frac{n^2}{P} \cdot \frac{A \times n(\max(\log_Z n, 1.0))}{\beta_{\text{mem}}} \]

Cache Capacity: \( Z \)
Memory BW: \( \beta_{\text{mem}} \)
Distributed 3D FFT - Performance Model

Computation Phase

Each node computes $n^2/p$ 1D FFTs of size $n$

Arithmetic Computation Time

$$T_{\text{flops}} = 3 \times \frac{n^2}{P} \times \frac{5n \log n}{C_{\text{node}}}$$

Memory Access Time

$$T_{\text{mem}} \approx 3 \times \frac{n^2}{P} \cdot \frac{A \times n(\max(\log Z, n, 1.0))}{\beta_{\text{mem}}}$$

Nodes: $P$
Compute Throughput: $C_{\text{node}}$

Cache Capacity: $Z$
Memory BW: $\beta_{\text{mem}}$
Distributed 3D FFT - Performance Model

Computation Phase

Each node computes \( n^2/p \) 1D FFTs of size \( n \)

Arithmetic Computation Time

\[
T_{\text{flops}} = 3 \times \frac{n^2}{P} \times \frac{5n \log n}{C_{\text{node}}}
\]

Memory Access Time

\[
T_{\text{mem}} \approx 3 \times \frac{n^2}{P} \cdot \frac{4 \times n (\max(\log_Z n, 1.0))}{\beta_{\text{mem}}}
\]

\( \Theta(1 + (n/L)(1 + \log_Z n)) \)

Lower bound (Frigo 1999)

Nodes: \( P \)
Compute Throughput: \( C_{\text{node}} \)

Cache Capacity: \( Z \)
Memory BW: \( \beta_{\text{mem}} \)
Distributed 3D FFT - Performance Model

\[ T_{\text{net}} \approx 2 \times \frac{n^3}{P^\frac{2}{3} \beta_{\text{link}}} \]

Nodes: P  Network BW: \( \beta_{\text{link}} \)
Validation
3D FFT Software

Distributed 3D FFT Framework

Optimized 1D FFT Library

$p3dfft$
3D FFT Software

Distributed 3D FFT Framework + Optimized 1D FFT Library

p3dfft

{ FFTW ESSL MKL CUFFT }
Test Machines

**Hopper**
6,392 Nodes
Opteron 6100 CPU

- Processor Peak: 50.4 GF/s
- Cores: 6
- Memory BW: 21.3 GB/s
- Fast Memory: 6 MB
- Link BW: 10 GB/s

**Keeneland**
120 Nodes (3xGPUs per node)
Tesla M2070 GPU

- Processor Peak: 515 GF/s
- Cores: 448
- Memory BW: 144 GB/s
- Fast Memory: 2.7 MB
- Link BW: 2 GB/s
Artifacts
GPU vs CPU Performance

FFT Performance on Keeneland

Problem Size

Gflops/s

0
175
350
525
700

0
750
1500
2250
3000

GPU
CPU
GPU vs CPU Performance

FFT Performance on Keeneland

- **Gflops/s**
- **Problem Size**
- **GPU**
- **CPU**

20% Difference
Artifacts - PCIe Bottleneck
Projections
Predicting 2020 Technology

Component Performance Relative to 2010 Technology

Relative Performance

Year

Historical  Projected

1990  2000  2010  2020

(59x) Compute
(32x) Cache
(22x) Network BW
(10x) Memory BW
## Technology Extrapolation

<table>
<thead>
<tr>
<th>Year</th>
<th>CPU-Based</th>
<th>GPU-Based</th>
</tr>
</thead>
</table>
| 2010 | **Processor Peak:** 50.4 GF/s  
Memory BW: 21.3 GB/s  
Fast Memory: 6 MB  
Link BW: 10 GB/s  
79,400 Processors | **Processor Peak:** 515 GF/s  
Memory BW: 144 GB/s  
Fast Memory: 2.7 MB  
Link BW: 10 GB/s  
6,392 Processors |
| 2020 | **Processor Peak:** 3 TF/s  
Memory BW: 206 GB/s  
Fast Memory: 192 MB  
Link BW: 218 GB/s  
1.3 M Processors | **Processor Peak:** 30 TF/s  
Memory BW: 1.4 TB/s  
Fast Memory: 86.4 MB  
Link BW: 218 GB/s  
135,000 Processors |
3D FFTs at Exascale (2020, n=21000)

<table>
<thead>
<tr>
<th>GPU</th>
<th>CPU-1: Same Peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>131k sockets</td>
<td>1M sockets</td>
</tr>
<tr>
<td>Peak = 3.98 EF/s</td>
<td>Peak = 3.98 EF/s</td>
</tr>
<tr>
<td>Bisection = 1.12 PB/s</td>
<td>Bisection = 5.29 PB/s</td>
</tr>
</tbody>
</table>

Table:

- GPU:
  - 131k sockets
  - Peak = 3.98 EF/s
  - Bisection = 1.12 PB/s

- CPU-1: Same Peak:
  - 1M sockets
  - Peak = 3.98 EF/s
  - Bisection = 5.29 PB/s

Graph:

- Time (seconds)
- Comm.:
  - Memory
  - Network

- GPU:
  - Total time: 0.528 seconds
  - Communication time: 0.19 seconds

- CPU-1:
  - Total time: 0.112 seconds
  - Communication time: 0.116 seconds

Note on xPU-memory stacking:

One technology that could disrupt these analyses is xPU-memory die stacking, the leading proposed mechanism for enabling memory bandwidth to scale at the same rate as compute capacity [34]. Applying our model, which is based on the known I/O complexity estimates for the FFT, we can establish that a node’s computation and memory transfers will be balanced when $T_{mem} \leq T_{flops}$ [11, 33]. Given a chip with $\cdot$ cores, $C$ flop/s per core, and a shared cache of size $Z$, this balance constraint yields $\cdot \cdot \cdot O \cdot \cdot \cdot [11]$. In theory, stacked memory makes it possible to keep the left-hand side constant over time. Although $\cdot$ grows faster than $Z$, it enters into this inequality through the log and so will not decrease too quickly. Thus, stacked memory would keep the processing system balanced for FFTs. However, if it is not possible to keep $\cdot = \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \·
Performance vs Balance

**CPU**
- **Processor**: 50.4 GFlop/s
- **Cache**: 6 MB
- **Memory**: 21.3 GB/s

**GPU**
- **Processor**: 515 GFlop/s
- **Cache**: 2.7 MB
- **Memory**: 144 GB/s

Flop/s : Byte/s = 2.3

Flop/s : Byte/s = 3.6

The GPU offers better performance, but is less balanced.
Two Costs: $T_{\text{memory}} + T_{\text{network}}$
Two Costs: $T_{\text{memory}} + T_{\text{network}}$
Two Costs: $T_{memory} + T_{network}$
Impact of Machine Balance

\[ T_{\text{mem}} \approx O \left( \frac{1}{R_{\text{peak}}} \cdot \frac{C_{\text{node}}}{\beta_{\text{mem}}} \cdot n^3 \log Z \cdot n \right) \]

\[ T_{\text{net}} \approx O \left( \frac{1}{R_{\text{peak}}} \cdot \frac{C_{\text{node}}^\kappa}{\beta_{\text{link}}} \cdot n^3 \right) \]

Number of processors: \( R_{\text{peak}} / C_{\text{node}} \)
Impact of Machine Balance

\[ T_{\text{mem}} \approx O \left( \frac{1}{R_{\text{peak}}} \cdot \frac{C_{\text{node}}}{\beta_{\text{mem}}} \cdot n^3 \log_Z n \right) \]

\[ T_{\text{net}} \approx O \left( \frac{1}{R_{\text{peak}}} \cdot \frac{C_{\text{node}}^\kappa}{\beta_{\text{link}}} \cdot n^3 \right) \]

Number of processors: \( \frac{R_{\text{peak}}}{C_{\text{node}}} \)
3D FFTs at Exascale (2020, n=21000)

Figure 5: We consider three extrapolated CPU-like systems vs. an extrapolated GPU-like system. CPU-1 has the same peak as GPU; CPU-2 computes the FFT in the same total time as GPU, assuming no overlap of communication; and CPU-3 also yields the same time as GPU, but assuming full overlap. In all cases, the CPU systems actually perform less network communication.

Note on xPU-memory stacking. One technology that could disrupt these analyses is xPU-memory die stacking, the leading proposed mechanism for enabling memory bandwidth to scale at the same rate as compute capacity [34]. Applying our model, which is based on the known I/O complexity estimates for the FFT, we can establish that a node's computation and memory transfers will be balanced when $T_{mem} \approx T_{flops}$ [11, 33]. Given a chip with $\rho$ cores, $C_{flop/s}$ per core, $\text{byte/s}$ bandwidth to the chip, and a shared cache of size $Z$, this balance constraint yields [11]

$$\rho \cdot C_{flop/s} \approx O\left(\log Z \cdot \rho\right).$$

In theory, stacked memory makes it possible to keep the left-hand side constant over time. Although $\rho$ grows faster than $Z$, it enters into this inequality through the log and so will not decrease too quickly. Thus, stacked memory would keep the processing system balanced for FFTs. However, if it is not possible to keep $\rho = \rho\left(\log Z\right)$, then stacked memory only delays rather than solves the processor imbalance problem.

7. CONCLUSIONS AND FUTURE WORK

One claim of this paper is that I/O-bus (PCIe) and network bandwidth will not be the true limiters of performance for parallel 3D FFTs. Instead, it is intra-node communication due to main memory bandwidth that will have the biggest impact at exascale. In fact, our key prediction is that if we ignore intra-node balance, as a na"ively extrapolated GPU-style design would do, then we will hurt overall system balance by not taking advantage of the better scaling of network bandwidth relative to memory bandwidth. Thus, more architectural emphasis on intra-node balance will have the biggest pay-off in the long-run for communication bandwidth-bound computations on high-end systems.

In terms of absolute bandwidth values, high-density (GPU-based) compute nodes extend the time until which network bandwidth will outpace main memory bandwidth, but do not fundamentally solve the problem. The most interesting solution for FFT-like computations, which include sorting, is most likely stacked memory if it can indeed deliver proportional scaling of memory bandwidth to core counts.

Interestingly, the most common weak-but-balanced processor designs are those of mobile processors. Our analysis hints strongly that leveraging the volume of production of such processors, combined with high-quality integrated networking, die stacking, and a balanced-throughput design, could be an excellent building block for an exascale system. This notion may, in fact, be part of an emerging view in other large-scale systems, such as data centers [4, 38].

Algorithmically, perhaps the only way forward for FFT type computations is through much more aggressive data or numerical (e.g., low-rank) compression [18]. We believe our basic modeling methodology and its level of detail could provide similar kinds of insights for other computations, particularly if enriched with additional parameters and an explicit accounting of power, energy, and even dollar costs. This style of analysis could be especially useful for understanding the performance of other large-scale systems, such as data centers [4, 38].
3D FFTs at Exascale (2020, n=21000)

<table>
<thead>
<tr>
<th>System</th>
<th>Sockets</th>
<th>Peak Speed</th>
<th>Bisection</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>131k</td>
<td>3.98 EF/s</td>
<td>1.12 PB/s</td>
</tr>
<tr>
<td>CPU–1: Same Peak</td>
<td>1M</td>
<td>3.98 EF/s</td>
<td>5.29 PB/s</td>
</tr>
<tr>
<td>CPU–2: Same Total</td>
<td>350k</td>
<td>1.04 EF/s</td>
<td>2.16 PB/s</td>
</tr>
<tr>
<td>CPU–3: Same Overlap</td>
<td>295k</td>
<td>2276 PF/s</td>
<td>1.93 PB/s</td>
</tr>
</tbody>
</table>

- **Comm.**
  - **Memory**
  - **Network**

- **Time (seconds):**
  - GPU: 0.19
  - CPU–1: 0.116
  - CPU–2: 0.274
  - CPU–3: 0.307

**Note on xPU-memory stacking.**

One technology that could disrupt these analyses is xPU-memory die stacking, the leading proposed mechanism for enabling memory bandwidth to scale at the same rate as compute capacity [34]. Applying our model, which is based on the known I/O complexity estimates for the FFT, we can establish that a node's computation and memory transfers will be balanced when $T_{\text{mem}} \leq T_{\text{flops}}$ [11, 33]. Given a chip with $C_{\text{flop/s}}$ per core, $C_{\text{byte/s}}$ bandwidth to the chip, and a shared cache of size $Z$, this balance constraint yields [11] $C \cdot C_{\text{flop}} \leq O(\log Z) \cdot C$.

In theory, stacked memory makes it possible to keep the left-hand side constant over time. Although $C$ grows faster than $Z$, it enters into this inequality through the log and so will not decrease too quickly. Thus, stacked memory would keep the processing system balanced for FFTs. However, if it is not possible to keep $C = \mathcal{O}(\log Z)$, then stacked memory only delays rather than solves the processor imbalance problem.

**7. CONCLUSIONS AND FUTURE WORK**

One claim of this paper is that I/O-bus (PCIe) and network bandwidth will not be the true limiters of performance for parallel 3D FFTs. Instead, it is intra-node communication due to main memory bandwidth that will have the biggest impact at exascale. In fact, our key prediction is that if we ignore intra-node balance, as a naively extrapolated GPU-style design would do, then we will hurt overall system balance by not taking advantage of the better scaling of network bandwidth relative to memory bandwidth. Thus, more architectural emphasis on intra-node balance will have the biggest pay-off in the long-run for communication bandwidth-bound computations on high-end systems.

In terms of absolute bandwidth values, high-density (GPU-based) compute nodes extend the time until which network bandwidth will outpace main memory bandwidth, but do not fundamentally solve the problem. The most interesting solution for FFT-like computations, which include sorting, is most likely stacked memory if it can indeed deliver proportional scaling of memory bandwidth to core counts.

Interestingly, the most common weak-but-balanced processor designs are those of mobile processors. Our analysis hints strongly that leveraging the volume of production of such processors, combined with high-quality integrated networking, die stacking, and a balanced-throughput design, could be an excellent building block for an exascale system. This notion may, in fact, be part of an emerging view in other large-scale systems, such as data centers [4, 38].

Algorithmically, perhaps the only way forward for FFT type computations is through much more aggressive data or numerical (e.g., low-rank) compression [18].

We believe our basic modeling methodology and its level of detail could provide similar kinds of insights for other computations, particularly if enriched with additional parameters and an explicit accounting of power, energy, and even dollar costs. This style of analysis could be especially...
Questions?
### Table 1: Processor architecture projections, from start-\(\times\)year 2007 to end-year 2020

<table>
<thead>
<tr>
<th>Parameter</th>
<th>2010 values</th>
<th>Doubling time (in years)</th>
<th>10-year increase factor</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak:</td>
<td>(C_{\text{CPU}})</td>
<td>50.4 GF/s</td>
<td>1.7</td>
<td>59.0(\times)</td>
</tr>
<tr>
<td></td>
<td>(C_{\text{GPU}})</td>
<td>515 GF/s</td>
<td></td>
<td>30 TF/s</td>
</tr>
<tr>
<td>Cores:</td>
<td>(\rho_{\text{CPU}})</td>
<td>6</td>
<td>1.87</td>
<td>40.7(\times)</td>
</tr>
<tr>
<td></td>
<td>(\rho_{\text{GPU}})</td>
<td>448</td>
<td></td>
<td>18k</td>
</tr>
<tr>
<td>Memory bandwidth:</td>
<td>(\beta_{\text{CPU}})</td>
<td>21.3 GB/s</td>
<td>3.0</td>
<td>9.7(\times)</td>
</tr>
<tr>
<td></td>
<td>(\beta_{\text{GPU}})</td>
<td>144 GB/s</td>
<td></td>
<td>1.4 TB/s</td>
</tr>
<tr>
<td>Fast memory</td>
<td>(Z_{\text{CPU}})</td>
<td>6 MB</td>
<td>2.0</td>
<td>32.0(\times)</td>
</tr>
<tr>
<td></td>
<td>(Z_{\text{GPU}})</td>
<td>2.7 MB(^b)</td>
<td></td>
<td>86.4 MB</td>
</tr>
<tr>
<td>Line size:</td>
<td>(L_{\text{CPU}})</td>
<td>64 B</td>
<td>10.2</td>
<td>2.0(\times)</td>
</tr>
<tr>
<td></td>
<td>(L_{\text{GPU}})</td>
<td>128 B</td>
<td></td>
<td>256 B</td>
</tr>
<tr>
<td>Link bandwidth:</td>
<td>(\beta_{\text{link}})</td>
<td>10 GB/s</td>
<td>2.25</td>
<td>21.8(\times)</td>
</tr>
<tr>
<td>Machine peak:</td>
<td>(R_{\text{peak}})</td>
<td>4 PF/s</td>
<td>1.0</td>
<td>1000(\times)</td>
</tr>
<tr>
<td>System memory:</td>
<td>(E)</td>
<td>635 TB</td>
<td>1.3</td>
<td>208(\times)</td>
</tr>
<tr>
<td>Nodes ((\frac{R_{\text{peak}}}{C})):</td>
<td>(P_{\text{CPU}})</td>
<td>79,400</td>
<td>2.4</td>
<td>17.4(\times)</td>
</tr>
<tr>
<td></td>
<td>(P_{\text{GPU}})</td>
<td>7,770</td>
<td></td>
<td>135,000</td>
</tr>
</tbody>
</table>
Distributed 3D FFT - Performance Model

3D FFT Using the Pencil Decomposition of the Transpose Method

Computation Phase #1
- 1D FFT in the X-direction

Communication Phase #1
- X → Y Transpose

Computation Phase #2
- 1D FFT in the Y-direction

Communication Phase #2
- Y → Z Transpose

Computation Phase #3
- 1D FFT in the Z-direction

\[
T_{\text{mem}} \approx 3 \times \frac{n^2}{P} \cdot \frac{A \times n(\max(\log_Z n, 1.0))}{\beta_{\text{mem}}}
\]

\[
T_{\text{net}} \approx 2 \times \frac{n^3}{P^{\frac{2}{3}} \beta_{\text{link}}}
\]

Cache Capacity (Z)
Nodes (P)
Memory BW (\(\beta_{\text{mem}}\))
Network BW (\(\beta_{\text{net}}\))
3D FFT on GPU Cluster

- Network: 36%
- Data Shuffle: 16%
- GPU<->CPU: 27%
- FFT Comp: 21%
FFT Performance on Hopper

![FFT Performance Graph](image)

- **Problem Size**
- **Time (s)**

Legend:
- **Communication**
- **Computation**