Communication [Lower] Bounds for Heterogeneous Architectures

Julian Bui
Outline

• Problem
• Goal
• MV multiplication w/i constant factor of the lower bound
• Matrix-matrix multiplication w/i constant factor of the lower bound
  – With Examples!
• If there’s time: derivation of proofs of lower bounds for $N^2$ and $N^3$ problems
Problem

• $N^2$ problems like MV multiply
  – Dominated by the communication to read inputs and write outputs to main memory

• $N^3$ problems like matrix-matrix multiply
  – Dominated by communication between global and local memory (Loomis-Whitney)
Architecture
Goal

• Let’s create a model based on the communication scheme to minimize the overall time

• Execution time becomes a function of processing speed, bandwidth, message latency, and memory size.
Optimization Function

\[ T \geq \min_{\sum F_i = G} \max_{1 \leq i \leq P} \gamma_i F_i + \beta_i \max \left\{ I_i + O_i, \frac{F_i}{8\sqrt{M_i}} \right\} + \alpha_i \max \left\{ \frac{I_i + O_i}{M_i}, \frac{F_i}{8M_i^{3/2}} \right\} \]

- **Time** won’t be faster than the fastest heterogeneous element (could be a GPU, for example).
- **Inv. Bandwidth**
- **Latency to shared memory**
- **Overall execution time**
- **The slowest processor is the weakest link**
- **Num. FLOPS times computation speed**
- **Comm. cost of reading inputs and writing outputs**
- **Num. bytes & msgs passed between global and shared memory**
Heterogeneous MV multiply

$A \cdot x = y$

Figure 3: HGEMV splitting
Heterogeneous, recursive MM

Algorithm 2 Heterogeneous matrix-matrix multiplication

Require: Matrices $A, B \in \mathbb{R}^{n \times n}$, stored in block-recursive order, $n$ is a power of two

1. Measure $a_i, \beta_j, \gamma_i, M_i$ and set $\delta_i$ according to equation (9) for each $1 \leq i \leq P$
2. for $i = 1$ to $P$ do
3.   Set $F_i$ according to equation (10) where $G = n^3$
4.   Set $k_i$ to be the largest integer such that $3(n/2^{k_i})^2 \geq M_i$
5.   Convert $F_i/G$ into octal and round to $k_i^{th}$ digit: $0.d_1^{(i)}d_2^{(i)} \ldots d_{k_i}^{(i)}$
6. end for
7. Initialize $S = \{ A \cdot B \}$
8. for $j = 1$ to $\max k_i$ do
9.   Subdivide all problems in $S$ into 8 subproblems according to square recursive GEMM
10. Assign $d_j^{(i)}$ subproblems to proc$_i$ and remove subproblems from $S$
11. end for
12. for all proc$_i$ parallel do
13.   Compute assigned subproblems using square recursive GEMM
14. end for

Ensure: Matrix $C = AB$, stored in block-recursive order

Solve the optimization equation for each processor, then assign it an appropriate amount of work based in units of work that are multiples of 8 sub MM multiplication blocks

Recursively solve each sub-block using a divide and conquer method
Recursive, Parallel MM

\[
\begin{array}{cc}
C & D \\
E & F \\
\end{array} \times \begin{array}{cc}
G & H \\
I & J \\
\end{array} = \\
\begin{array}{cc}
CG & CH \\
EG & EH \\
\end{array} + \begin{array}{cc}
DI & DJ \\
FI & FJ \\
\end{array}
\]
\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16 \\
\end{array}
\times
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16 \\
\end{array}
= 
\begin{array}{cccc}
\end{array}
\]
Review

\[
\begin{array}{cc}
C & D \\
E & F \\
\end{array}
\times
\begin{array}{cc}
G & H \\
I & J \\
\end{array}
=
\begin{array}{cc}
CG & CH \\
EG & EH \\
\end{array}
+
\begin{array}{cc}
DI & DJ \\
FI & FJ \\
\end{array}
\]
So we need \( EH + FJ \)
CG = 11 * 11 = 121
CH = 11 * 12 = 132
EG = 15 * 11 = 165
EH = 15 * 12 = 180
DI = 10 * 7 = 70
DJ = 10 * 8 = 80
FI = 14 * 7 = 98
FJ = 14 * 8 = 112

CG = 9 * 3 = 27
CH = 9 * 4 = 36
EG = 13 * 3 = 39
EH = 13 * 4 = 52
DI = 10 * 7 = 70
DJ = 10 * 8 = 80
FI = 14 * 7 = 98
FJ = 14 * 8 = 112

CG = 11 * 11 = 121
CH = 11 * 12 = 132
EG = 15 * 11 = 165
EH = 15 * 12 = 180
DI = 12 * 15 = 180
DJ = 12 * 16 = 192
FI = 16 * 15 = 240
FJ = 16 * 16 = 256
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td>DI</td>
<td>CG</td>
<td>DI</td>
<td></td>
</tr>
<tr>
<td>$9 \times 3 = 27$</td>
<td>$10 \times 7 = 70$</td>
<td>$11 \times 11 = 121$</td>
<td>$12 \times 15 = 180$</td>
<td></td>
</tr>
<tr>
<td>CH</td>
<td>DJ</td>
<td>CH</td>
<td>DJ</td>
<td></td>
</tr>
<tr>
<td>$9 \times 4 = 36$</td>
<td>$10 \times 8 = 80$</td>
<td>$11 \times 12 = 132$</td>
<td>$12 \times 16 = 192$</td>
<td></td>
</tr>
<tr>
<td>EG</td>
<td>FI</td>
<td>EG</td>
<td>FI</td>
<td></td>
</tr>
<tr>
<td>$13 \times 3 = 39$</td>
<td>$14 \times 7 = 98$</td>
<td>$15 \times 11 = 165$</td>
<td>$16 \times 15 = 240$</td>
<td></td>
</tr>
<tr>
<td>EH</td>
<td>FJ</td>
<td>EH</td>
<td>EH</td>
<td></td>
</tr>
<tr>
<td>$13 \times 4 = 52$</td>
<td>$14 \times 8 = 112$</td>
<td>$15 \times 12 = 180$</td>
<td>$15 \times 12 = 180$</td>
<td></td>
</tr>
</tbody>
</table>
CG = 9 * 3 = 27  
CH = 9 * 4 = 36  
EG = 13 * 3 = 39  
EH = 13 * 4 = 52  

DI = 10 * 7 = 70  
DJ = 10 * 8 = 80  
FI = 14 * 7 = 98  
FJ = 14 * 8 = 112  

CG = 11 * 11 = 121  
CH = 11 * 12 = 132  
EG = 15 * 11 = 165  
EH = 15 * 12 = 180  

DI = 12 * 15 = 180  
DJ = 12 * 16 = 192  
FI = 16 * 15 = 240  
EH = 15 * 12 = 180

\[
\begin{array}{cc}
C & D \\
E & F
\end{array}
\times
\begin{array}{cc}
G & H \\
I & J
\end{array}
+
\begin{array}{cc}
C & D \\
E & F
\end{array}
\times
\begin{array}{cc}
G & H \\
I & J
\end{array}
=\]
CG = 9 * 3 = 27
CH = 9 * 4 = 36
EG = 13 * 3 = 39
EH = 13 * 4 = 52
DI = 10 * 7 = 70
DJ = 10 * 8 = 80
FI = 14 * 7 = 98
FJ = 14 * 8 = 112
CG = 11 * 11 = 121
CH = 11 * 12 = 132
EG = 15 * 11 = 165
EH = 15 * 12 = 180
DI = 12 * 15 = 180
DJ = 12 * 16 = 192
FI = 16 * 15 = 240
EH = 15 * 12 = 180
\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16 \\
\end{array}
\times
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16 \\
\end{array}
= 
\begin{array}{cccc}
398 & 440 & 542 & 600 \\
\end{array}
\]
Derivation of Proofs

• For a MM, the a node can only do $O(N \times \sqrt{N})$ useful arithmetic operations per phase.
  – If a single processor accesses rows of matrix A and some of these rows have at least $\sqrt{N_A}$ elements, then the processor will touch at most $\sqrt{N_A}$ rows of matrix A.
  – Since each row of C is a product of a row in A and all of B, the number of useful multiplications on a single processor that involve rows of matrix A are bounded by $O(N_B \sqrt{N_A})$ or $O(N \sqrt{N})$. 
Derivation Cont’d

• Number of total phases = the total amt. of work per processor divided by the amt. of work a processor can do per phase
  \[ O\left(\frac{G}{M \cdot \sqrt{M}}\right) \]

• Total amt. of communication = # phases times memory used per phase (\(\sim M\))
  \[ \frac{G}{\sqrt{M}} \]
Derivation Cont’d

• Total # Words, $W = \frac{G}{\sqrt{M}}$
• $G / \sqrt{M} = W = G * M / (M * \sqrt{M})$
• $W \geq ((G / (M * \sqrt{M})) - 1) * M$
• $W \geq (G / \sqrt{M}) - M$