Brief Announcement:
Communication Bounds for Heterogeneous Architectures

Grey Ballard, James Demmel, Andrew Gearhart

UC Berkeley

SPAA
June 5, 2011

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227). Additional support comes from Par Lab affiliates National Instruments, NEC, Nokia, NVIDIA, and Samsung.
Communication-avoiding algorithms move as little data as possible, since these are the slowest and energy-hungriest operations any computer performs.

We present a general model for heterogeneous architectures.

We extend previous work on communication lower bounds to our heterogeneous model.

We provide a communication-optimal heterogeneous algorithm for matrix multiplication.
Previous Models of Computation

- Machine models assume “fast” and “slow” types of memory access
- We wish to asymptotically minimize “slow” traffic
  - goal of the models is to capture the design space of likely best algorithms...while leaving specific parameter selection to autotuners
Previous Models of Computation

- Machine models assume “fast” and “slow” types of memory access
- We wish to asymptotically minimize “slow” traffic
  - goal of the models is to capture the design space of likely best algorithms...while leaving specific parameter selection to autotuners

**Sequential**: Single processor separated from memory via a small cache
- fast = cache, slow = DRAM

**Distributed**: A group of processors connected on a network
- fast = local, slow = remote
For this work, we assume that each processor has an independent link to a shared global memory and that system parameters are constant.

Processor $i$ has parameters:
- $M_i$: local memory size (words)
- $\gamma_i$: processing rate (sec/flop)
- $\alpha_i$: latency (sec/msg)
- $\beta_i$: inverse bandwidth (sec/word)

E.g., multicore CPU + GPU, 8 ARM cores + 2 Nehalem cores, ...
We model a parallel program’s total runtime $T$ by

$$T(\{F_i, W_i, L_i\}) = \max_{1 \leq i \leq P} \{\gamma_i F_i + \beta_i W_i + \alpha_i L_i\}$$

$\gamma_i = \text{sec/flop}$  \quad $\beta_i = \text{sec/word}$  \quad $\alpha_i = \text{sec/msg}$

$F_i = \text{flops}$  \quad $W_i = \text{words}$  \quad $L_i = \text{messages}$
We model a parallel program’s total runtime $T$ by

$$T(\{F_i, W_i, L_i\}) = \max_{1 \leq i \leq P} \{\gamma_i F_i + \beta_i W_i + \alpha_i L_i\}$$

$\gamma_i = \text{sec/flop}$  $\beta_i = \text{sec/word}$  $\alpha_i = \text{sec/msg}$

$F_i = \text{flops}$  $W_i = \text{words}$  $L_i = \text{messages}$

Previous bounds for homogeneous model [HK81, ITT04, BDHS11]:

- assuming (3 nested loops) linear algebra computations

$$W_i \geq \frac{F_i}{8\sqrt{M_i}} \quad L_i \geq \frac{F_i}{8M_i^{3/2}}$$
Heterogeneous Runtime Lower Bound

Applying these bounds to each processor we obtain a runtime lower bound for any partition of $G$ flops to $P$ processors:

$$T \geq \min_{\sum F_i = G} \max_{1 \leq i \leq P} \left\{ \gamma_i F_i + \beta_i \frac{F_i}{8\sqrt{M_i}} + \alpha_i \frac{F_i}{8M_i^{3/2}} \right\}$$
Heterogeneous Runtime Lower Bound

Applying these bounds to each processor we obtain a runtime lower bound for any partition of $G$ flops to $P$ processors:

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

which can be simplified by solving the linear program:

$$T \geq \frac{G}{\sum \frac{1}{\delta_j}} \text{ where } \delta_i = \gamma_i + \frac{\beta_i}{8 \sqrt{M_i}} + \frac{\alpha_i}{8 M_i^{3/2}}$$
Applying these bounds to each processor we obtain a runtime lower bound for any partition of $G$ flops to $P$ processors:

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

which can be simplified by solving the linear program:

$$T \geq \frac{G}{\sum \frac{1}{\delta_j}} \text{ where } \delta_i = \gamma_i + \frac{\beta_i}{8\sqrt{M_i}} + \frac{\alpha_i}{8M_i^{3/2}}$$

- this bound is attainable for sufficiently large dense matrix multiplication
- the solution to LP gives optimal partition of flops (insight for algorithm)

$$F_i = \frac{1}{\delta_i} \frac{1}{\sum \frac{1}{\delta_j}} G$$
Heterogeneous Algorithm: Matrix Multiplication

Our algorithm is based on the square recursive algorithm (8 subproblems)

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\cdot
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix}
=
\begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]
Heterogeneous Algorithm: Matrix Multiplication

Our algorithm is based on the square recursive algorithm (8 subproblems)

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix} =
\begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]
Heterogeneous Algorithm: Matrix Multiplication

Our algorithm is based on the square recursive algorithm (8 subproblems)

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix} =
\begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]

Algorithm Overview

- Use static scheduling based on measured machine parameters
- Use flop distribution given by linear program from lower bound
  - load balances given optimal communication costs
Heterogeneous Algorithm: Matrix Multiplication

Our algorithm is based on the square recursive algorithm (8 subproblems)

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix} =
\begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]

Algorithm Overview

- Use static scheduling based on measured machine parameters
- Use flop distribution given by linear program from lower bound
  - load balances given optimal communication costs
- Assign as many large subproblems to processors as possible
  - convert \( F_i / G \) to octal to determine how many subproblems to assign to processor \( i \) at each level of recursion
  - minimizes per processor bandwidth cost
- Use block-recursive data structure so that subproblems are contiguous
  - minimizes per processor latency cost
Future Work

- Implementation and evaluation of matrix multiplication algorithm
  - CPU+GPU, Intel development prototype, and others

- Algorithms for more complicated algorithms: LU and QR
  - must consider dependencies within algorithms

- Can a dynamic scheduling/work stealing approach provide more flexibility and still minimize communication costs?
  - necessary for heterogeneity in time

- What would an energy-optimal algorithm look like on a heterogeneous machine? How does this relate to the communication-avoiding paradigm?
Thank you!

Grey Ballard
ballard@cs.berkeley.edu
www.eecs.berkeley.edu/~ballard
References

Minimizing communication in linear algebra.

J. W. Hong and H. T. Kung.
I/O complexity: The red-blue pebble game.

D. Irony, S. Toledo, and A. Tiskin.
Communication lower bounds for distributed-memory matrix multiplication.
Heterogeneous Model: Considerations

PCIe bandwidth w/ pinned memory

Xeon E5405 (FSB-based, 8 cores) and GTX280 GPU
Heterogeneous Model: Considerations

PCIe bandwidth w/ pinned memory

Xeon E5530 (NUMA, 8 cores) and Tesla C2050 GPU

Transfer Size (doubles) vs. GB/s

- cores busy with stream benchmark
- cores idle
Heterogeneous Model: Considerations

PCle Latency (dirac)

each trial 10000 ping-pong

Time (sec)

Trial

Busy
Busy (avg)
Idle
## Heterogeneous Matrix Multiplication

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>.25</td>
<td>.25</td>
<td>.40625</td>
<td>.09375</td>
</tr>
<tr>
<td>Octal</td>
<td>.20</td>
<td>.20</td>
<td>.32</td>
<td>.06</td>
</tr>
</tbody>
</table>
## Heterogeneous Matrix Multiplication

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>.25</td>
<td>.25</td>
<td>.40625</td>
<td>.09375</td>
</tr>
<tr>
<td>Octal</td>
<td>.20</td>
<td>.20</td>
<td>.32</td>
<td>.06</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursion 1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Recursion 2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>
Heterogeneous Matrix Multiplication

Recursion 1

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>.25</td>
<td>.25</td>
<td>.40625</td>
<td>.09375</td>
</tr>
<tr>
<td>Octal</td>
<td>.20</td>
<td>.20</td>
<td>.32</td>
<td>.06</td>
</tr>
</tbody>
</table>

Recursion 1

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursion 1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Recursion 2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>
Heterogeneous Matrix Multiplication

Recursion 1

\[
\begin{array}{c}
A_{11} A_{21} \\
A_{12} A_{22}
\end{array}
\]

Recursion 2

\[
\begin{array}{c}
A_{11} A_{21} \\
A_{12} A_{22}
\end{array}
\]

Decimal
<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>.25</td>
<td>.25</td>
<td>.40625</td>
<td>.09375</td>
</tr>
</tbody>
</table>

Octal
<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>.20</td>
<td>.20</td>
<td>.32</td>
<td>.06</td>
</tr>
</tbody>
</table>

Recursion 1

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Recursion 2

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>