Memory hierarchies
(1/2)
Prof. Rich Vuduc <richie@cc>
Feb. 18, 2009
Single-processor (single-core) performance:
A motivating example
Architecture 101: Memory hierarchy.

- **Fast** = cache or RAM or "local"
- **Slow** = RAM or disk or "remote"

10–10^4 x
Processor-memory gap.

Memory hierarchies reflect and help deal with growing speed gap between processors & memory.
AMD Opteron X2 (2 x 2)
Linear algebra 101: Matrix multiply.

Compute $C \leftarrow C + A \cdot B$

3-nested loops (4 lines of code)
HPC 101: Cache-blocking is I/O optimal.

Computation:
\[ C \leftarrow A \cdot B \]

Naïve

Blocked

Theorem [Hong & Kung (’81)]:
A blocked algorithm minimizes slow memory transfers.

Tuning parameter: Block size
That was theory. What happens in practice?
“Naïve” code + best compiler

- Fast (fits in cache)
- Slow ~6x

Problem size (matrix dimension) vs. Speed (billions of ops/sec)
Blocked algorithm + best compiler
Best possible (1-core/1-thread peak)

Peak = 8 Gflop/s on 1-core of 4-core 2 GHz AMD Opteron 8350
Hand-tuned

~6.5x (+80%)
Aside: Naïve parallelization on 4-core system

Recall: 1-core peak

Naïve parallelization (4-threads, OpenMP): 1.5x
Goals for today & Friday

- Understand the theory’s applicability and limits
  - “Prove” Hong & Kung theorem
  - What is relationship to data movement in a parallel system?
- Learn the basics of the memory hierarchy
  - Caches, TLB, memory
  - Ideas extend to disk systems
A simple model of memory
A simple memory model

\[ m \equiv \text{No. words moved from slow to fast memory} \]
\[ f \equiv \text{No. of flops} \]
\[ \alpha \equiv \text{Time per slow memory op.} \]
\[ \tau \equiv \text{Time per flop} \]
\[ q \equiv \frac{f}{m} = \text{Flop-to-mop ratio} \quad \Leftarrow \text{Computational intensity} \]
\[ T = f \cdot \tau + m \cdot \alpha = f \cdot \tau \cdot \left( 1 + \frac{\alpha}{\tau} \cdot \frac{1}{q} \right) \]

\text{Machine balance}
Example: Dense matrix-vector multiply

```plaintext
// Implements y ← y + A · x

for i ← 1 to n do

   for j ← 1 to n do
      y_i ← y_i + a_{ij} · x_j
```
Example: Dense matrix-vector multiply

// Implements $y \leftarrow y + A \cdot x$
// Read $x$ (into fast memory)
// Read $y$
for $i \leftarrow 1$ to $n$ do
  // Read $a_{i,*}$
  for $j \leftarrow 1$ to $n$ do
    $y_i \leftarrow y_i + a_{ij} \cdot x_j$
// Write $y$ to slow memory
Example: Dense matrix-vector multiply

// Implements \( y \leftarrow y + A \cdot x \)
// Read \( x \) (into fast memory)
// Read \( y \)
for \( i \leftarrow 1 \) to \( n \) do
    // Read \( a_{i,j} \)
    for \( j \leftarrow 1 \) to \( n \) do
        \( y_i \leftarrow y_i + a_{ij} \cdot x_j \)
// Write \( y \) to slow memory

\( f = 2n^2 \)
\( m = 3n + n^2 \)
\( q \approx 2 \)
\( \frac{T}{f \cdot \tau} \approx 1 + \frac{\alpha}{\tau} \cdot \frac{1}{2} \)
Simplifying assumptions

- Ignored flop/mop parallelism within processor → drop arithmetic term
- Assumed fast memory large enough to hold vectors
- Assumed no-cost fast memory access
- Memory latency is constant, charged per word
  - Ignored cache lines / block transfers
  - Ignored bandwidth
Predictive accuracy of this model
Naïve matrix-matrix multiply

// Implements $C \leftarrow C + A \cdot B$

for $i \leftarrow 1$ to $n$ do

  for $j \leftarrow 1$ to $n$ do

    for $k \leftarrow 1$ to $n$ do

      $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$

      $f = ?$

      $m \geq ?$

      $\frac{T}{f \cdot \tau} \geq ?$
Naïve matrix-matrix multiply: Best case

// Implements $C \leftarrow C + A \cdot B$

for $i \leftarrow 1$ to $n$ do

    for $j \leftarrow 1$ to $n$ do

        for $k \leftarrow 1$ to $n$ do

            $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$

$$f = 2n^3$$

$$m \geq 4n^2$$

$$\frac{T}{f \cdot \tau} \geq 1 + \frac{\alpha}{\tau} \cdot \frac{2}{n}$$
### Naïve matrix-matrix multiply

// Implements $C \leftarrow C + A \cdot B$

```
for i ← 1 to n do
    // Read row $a_{i,*}$
    for j ← 1 to n do
        // Read col $b_{*,j}$
        // Read $c_{i,j}$
        for k ← 1 to n do
            $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$
        // Write $c_{ij}$ to slow memory
```

\[
\begin{align*}
  f &= 2n^3 \\
  m &= n^3 + 3n^2 \\
  \frac{T}{f \cdot \tau} &\approx 1 + \frac{\alpha}{\tau} \cdot \frac{1}{2}
\end{align*}
\]
// Let $I, J, K = \text{blocks of } b \text{ indices}$

for $I \leftarrow \text{index blocks } 1 \text{ to } \frac{n}{b} \text{ do}$

for $J \leftarrow \text{index blocks } 1 \text{ to } \frac{n}{b} \text{ do}$

// Read block $C_{I,J}$

for $K \leftarrow \text{index blocks } 1 \text{ to } \frac{n}{b} \text{ do}$

// Read block $A_{I,K}$

// Read block $B_{K,J}$

$C_{I,J} \leftarrow C_{I,J} + A_{I,K} \cdot B_{K,J}$

// Write $C_{I,J}$ to slow memory
Blocked (tiled) matrix multiply

\[ m \approx \frac{n^3}{b} \quad \Rightarrow \quad q \approx b \]

\[ \frac{T}{f \cdot \tau} = 1 + \frac{\alpha}{\tau} \cdot \frac{1}{b} \]

*Subject to what constraints?*
Blocked (tiled) matrix multiply

\[ m \approx \frac{2n^3}{b} \implies q \approx b \]

\[ 3b^2 \leq Z \implies b \leq \sqrt[3]{\frac{Z}{3}} \]

\[ \frac{T}{f \cdot \tau} = 1 + \frac{\alpha}{\tau} \cdot \frac{1}{b} \]

\[ \geq 1 + \frac{\alpha}{\tau} \cdot \sqrt[3]{\frac{3}{Z}} \]
Architectural implications

<table>
<thead>
<tr>
<th>Arch.</th>
<th>≈ $\alpha / \tau$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ultra 2i</td>
<td>25</td>
<td>1.5 MB</td>
</tr>
<tr>
<td>Ultra 3</td>
<td>14</td>
<td>460 KB</td>
</tr>
<tr>
<td>Pentium 3</td>
<td>6.3</td>
<td>94 KB</td>
</tr>
<tr>
<td>P-3M</td>
<td>10</td>
<td>240 KB</td>
</tr>
<tr>
<td>Power3</td>
<td>8.8</td>
<td>180 KB</td>
</tr>
<tr>
<td>Power4</td>
<td>15</td>
<td>527 KB</td>
</tr>
<tr>
<td>Itanium 1</td>
<td>36</td>
<td>3.0 MB</td>
</tr>
<tr>
<td>Itanium 2</td>
<td>5.5</td>
<td>71 KB</td>
</tr>
</tbody>
</table>

"M" in bytes to 2 digits; assumes 8-byte (double-precision) words

\[
1 + \frac{\alpha}{\tau} \cdot \frac{1}{q} < 1.1
\]

\[
\Rightarrow Z \geq 300 \left(\frac{\alpha}{\tau}\right)^2
\]
I/O optimality
Optimal?

\[ b = O\left(\sqrt{Z}\right) \]
\[ \Rightarrow \quad m = O\left(\frac{n^3}{b}\right) = O\left(\frac{n^3}{\sqrt{Z}}\right) \]
Bounding amount of I/O possible
Bounding amount of I/O possible

- Consider a schedule in phases of exactly \( Z \) transfers each (except last)

- *Definition:* \( c(i,j) \) is **live** during phase \( p \) if ...
  - ... for some \( k \), we compute \( a(i,k) \times b(k,j) \);
  - *and* some partial sum of \( c(i, j) \) is either in cache or moved to main memory

- At most \( 2^Z \) live \( c(i, j) \) in phase \( p \)

- At most \( 2^Z \) distinct elements of \( A \) in cache during phase \( p \); same for \( B \)
  - Either in cache at beginning or moved to cache during phase
  - Let \( A_p \) be set of elements in cache during phase \( p \); same for \( B_p \)
How many multiplies in phase \( p \)?

- Let \( S_{p,+} \) = set of rows of \( A \) with \( Z^{1/2} \) or more elements in \( A_p \)
- Let \( S_{p,-} \) = set of rows of \( A \) with fewer
- \( |S_{p,+}| \leq 2Z^{1/2} \)
- Consider rows in \( S_{p,+} \):
  - Operation “\( a(i,:) \times B \)” touches each element of \( B \) only once
  - So, no. of scalar multiplies \( \leq |S_{p,+}| \times (2\times M) = 4Z^{3/2} \)
- For rows in \( S_{p,-} \), consider that “\( c(i,j) = \text{row} \times \text{col} \)”
  - Thus, \( (\# \text{ multiplies}) \leq (\text{no. live}) \times (\text{max row len}) \leq 2Z^{3/2} \)
Final bound on multiplies

Total no. of multiplies = $n^3$

No. of multiplies per phase $\leq 6Z^{\frac{3}{2}}$

No. of phases $\geq \left\lceil \frac{n^3}{6Z^{\frac{3}{2}}} \right\rceil$

Total no. of words transferred $\geq Z \cdot \left( \frac{n^3}{6Z^{\frac{3}{2}}} - 1 \right)$

$= \frac{n^3}{6\sqrt{Z}} - Z$
Can we do better? No.

- **Theorem [Hong & Kung (1981)]:** Any schedule of conventional mat-mul must transfer $\Omega(n^3 / \sqrt{Z})$ words between slow and fast memory, where $Z < n^2 / 6$.

- We did intuitive proof by Toledo (1999)

- Historical note: Rutledge & Rubinstein (1951—52)

- So cached block matrix multiply is **asymptotically optimal**.

\[ b = O \left( \sqrt{Z} \right) \implies m = O \left( \frac{n^3}{b} \right) = O \left( \frac{n^3}{\sqrt{Z}} \right) \]