Parallel Programming critical measures to avoid risk as multi-core chips proliferate into ordinary PCs

Frank Dehne

Carleton University
Ottawa, Canada
www.dehne.net

Sambamba Technologies
Ottawa, Canada
www.sambamba.ca
About the speaker

- Chancellor's Professor of Computer Science
- Research specialization:
  - parallel computing (multi-core, clusters, could)
  - parallel data warehousing & OLAP
  - parallel bioinformatics
- IEEE Technical Committee on Parallel Processing (Vice-Chair: 2003-2006)

- Fellow, IBM Center For Advanced Studies Canada
- President, SAMBAMBA Technologies, www.sambamba.ca
Multi-core processors

- Core 1: One or more levels of cache
- Core 2: One or more levels of cache
- Core 3: One or more levels of cache
- Core 4: One or more levels of cache

Main memory

multi-core chip
**Cloud computing**

**Amazon EC2 application**

**Elastic:** “Amazon EC2 enables you to increase or decrease capacity within minutes, not hours or days. You can commission one, hundreds or even thousands of server instances simultaneously.” (Amazon website)
“Auto Scaling allows you to automatically scale your Amazon EC2 capacity up or down according to conditions you define. With Auto Scaling, you can ensure that the number of Amazon EC2 instances you’re using scales up seamlessly...”

(Amazon website)
What about your application software...?

- Cloud infrastructure is elastic (use as many nodes as you like)
- Multi-core processors provide increasing numbers of cores in desktops, tablets, smartphones, ...
- What about your application software?
- Is your software elastic?
- Can your software make efficient use of large numbers of cloud nodes or processor cores?
What about your application software...?

Traditional software:  
Single-threaded

Multi-core and cloud software:  
Multi-threaded = parallel programming

Multi-Core
Why parallel software is needed

Cloud Computing:
• Traditional (single-threaded) software can not utilize the large number of processors available in today's cloud computing systems.
• Single-threaded software is unable to scale, i.e. gain performance with increasing number of cloud computing nodes.

Multi-Core Processors:
• Traditional (single-threaded) software can only utilize one single processor core.
• The performance of single-threaded software may actually decrease because multi-core processors use reduced clock speeds to save on energy consumption.
“Parallel programming, once an obscure niche, is the focus of increasing interest as “multicore” chips proliferate in ordinary PCs.”  
--- *The Economist*

“Finding ways to make it easy to write software that can take full advantage of the power of parallel processing … is rapidly becoming a problem for the whole industry.”  
--- *David Smith, Gartner*

“Parallel Computing ranks number 2 out of 7 Grand Challenges facing IT!”  
--- *Gartner*

“IT leaders should look ahead for the emerging technologies that will have a dramatic impact on their business..”  
--- *Ken McGee, Vice President and Gartner Fellow (elaborating on Gartner’s Grand Challenges facing IT)*
The move from single-threaded software to multi-threaded software

• sometimes easy
• sometimes hard
• sometimes very hard
n : problem size
p : number of proc.
T(p): parallel time
T_s : sequential time (optimal sequ. Alg.)

$$s(p) = \frac{T_s}{T(p)} :$$ speedup \((1 \leq s \leq p)\)
### Converting single-threaded software into multi-threaded software

<table>
<thead>
<tr>
<th>Application type</th>
<th>conversion to multi-core</th>
<th>conversion to cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Many small transactions (e.g. transaction processing systems)</td>
<td>easy</td>
<td>easy (except for data partitioning)</td>
</tr>
<tr>
<td>One large transaction, <strong>regular</strong> data access (e.g. image &amp; video processing, matrix based simulation)</td>
<td>medium</td>
<td>medium - hard</td>
</tr>
<tr>
<td>One large transaction, <strong>irregular</strong> data access (e.g. business analytics, stock market analysis, risk assessment, scheduling, transport networks, graph based simulation)</td>
<td>hard</td>
<td>very hard (sometimes impossible)</td>
</tr>
</tbody>
</table>
P-Completeness

Applications that are provably IMPOSSIBLE to parallelize (convert to multi-threaded):

- Circuit simulation
- Dept-first search (network analysis)
- Max-flow (operations research)
- Context-free grammar parsing (compilers)
Amdahl's Law

Let \( f, 0 < f < 1 \), be the fraction of a computation that is inherently sequential. Then the maximum obtainable speedup is \( s \leq \frac{1}{f + (1-f)/p} \).

Proof: \( T(p) \geq f \cdot T_s + (1-f) \cdot T_s / p \).

Hence \( s \leq T_s / [f \cdot T_s + (1-f) \cdot T_s /p] \)

\[ = 1 / [f+(1-f)/p]. \]

\[ f \rightarrow 0 : s(p) \rightarrow p \]
\[ f \rightarrow 1 : s(p) \rightarrow 1 \]
\[ f = 0.5 : s(p) = 2 \cdot [p/(p+1)] \leq 2 \]
\[ f = 1/k : s(p) = k / [1+(k-1)/p] \leq k \]
How parallelization works in practice

- Task partitioning
- Data partitioning
- Worry about:
  - work vs. span
  - data transfer overhead
  - synchronization
  - race conditions
  - deadlocks

\[
T_1 = \text{work} \\
T_\infty = \text{span}
\]

Speedup = \[
\frac{\text{work}}{\text{span}}
\]
Parallel Sort for Cloud Computing

- N data items
- P processors (cloud nodes)
- Connected via cloud network

- Task: Sort all data (e.g. business analytics, OLAP)
Parallel Sort for Cloud Computing
Parallel Sort for Cloud Computing

Algorithm:
- sort locally and create p-sample
Algorithm:

• send all p-samples to proc. 1
Parallel Sort for Cloud Computing

Algorithm:

- proc.1: sort all received samples and compute global p-sample
Parallel Sort for Cloud Computing

Algorithm:

- broadcast global p-sample
- bucket locally according to global p-sample
- send bucket \( i \) to \( \text{proc}.i \)
- resort locally
Lemma: Each proc. receives at most $2 \frac{n}{p}$ data items
Parallel Sort for Cloud Computing

Post-Processing: “Array Balancing”
Parallel Sort for Cloud Computing

- 5 data movements for $n/p > p^2$
- $O(n/p \log n)$ local comp.
Parallel Sort for Cloud Computing

Performance:

- $N = 10,000,000$ data items
- Between 1 and 28 processors (cloud nodes)
Converting single-threaded software into multi-threaded software

<table>
<thead>
<tr>
<th>Application type</th>
<th>conversion to multi-core</th>
<th>conversion to cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Many small transactions (e.g. transaction processing systems)</td>
<td>easy</td>
<td>easy (except for data partitioning)</td>
</tr>
<tr>
<td>One large transaction, <strong>regular</strong> data access (e.g. image &amp; video processing, matrix based simulation)</td>
<td>medium</td>
<td>medium - hard</td>
</tr>
<tr>
<td>One large transaction, <strong>irregular</strong> data access (e.g. business analytics, stock market analysis, risk assessment, scheduling, transport networks, graph based simulation)</td>
<td>hard</td>
<td>very hard</td>
</tr>
<tr>
<td></td>
<td>(sometimes impossible)</td>
<td>(sometimes impossible)</td>
</tr>
</tbody>
</table>
For more information...

Dr. Frank Dehne
Sambamba Technologies
frank@sambamba.ca

www.sambamba.ca