Multicore is dead; Long live Multicore!

Stephen Blair-Chappell
Intel Compiler Labs
(Ten) reasons why programming for multicore should be avoided. In this tongue-in-cheek session we take a head-in-the-sand approach to multicore programming. We present a number of anecdotal reasons why you should never program for multicore. Includes a rapid examination of several case studies.

Warning: Content may be subject to exaggeration and hyperbola - after attending this session you may never write a parallel program again.
1. Multicore is just a fad!

The Problem – Technology Shelf Life
Growth in Cores - A well rehearsed story

Multi-core ramp accelerates in 2006

- Today
  - Dual core
  - Quad core
  - Multi-socket solutions
- The Future
  - 6 & 8 cores
  - many-core
- R & D
  - 80 cores ...

The software industry goes parallel
Increase application performance & scalability

The Challenge
- Serial applications can not take advantage of multicore platforms.
- Number of processor cores is increasing.
- Remaining competitive requires parallelizing serial code or creating new parallel applications.

Performance cannot keep up with multi-core gains.

1. Multicore is just a fad!

Software & Services Group, Developer Products Division
Copyright© 2011, Intel Corporation. All rights reserved. *Other brands and names are the property of their respective owners.
1. Multicore is just a fad!
Silicon

1. Multicore is just a fad!  ##
1. Multicore is just a fad!

False
2. My Program will run just the same without any effort!

The Problem – No problems!
Vector Processing

Scalar Processing

\[
\begin{align*}
    & r1 & r2 \\
    \downarrow & & \downarrow \\
    & r3 & \\
\end{align*}
\]

\[
\text{add.d } r3, r1, r2
\]

Vector Processing

\[
\begin{align*}
    & v1 & v2 \\
    \downarrow & & \downarrow \\
    & v3 & \\
\end{align*}
\]

\[
\text{addvec.d } v3, v1, v2
\]

The life of a program instruction

1. Instruction read from memory
2. Instruction fed to Decoder
3. Micro-ops (uops) generated
4. uops queued in RS
5. uops dispatched
6. Results sent to ROB
7. Instruction marked - all uops executed
8. Instruction sent for retirement

3. The CPU automatically makes things parallel - so I don't need to.

2. My Program will run just the same without any effort!
## Layers of Optimisation

<table>
<thead>
<tr>
<th>Optimisation</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristics</td>
<td>Direct Sound</td>
</tr>
<tr>
<td>Libraries</td>
<td>IPP</td>
</tr>
<tr>
<td>Soft\Hard RT</td>
<td>Win32\RTX</td>
</tr>
<tr>
<td>Code Generation</td>
<td>Intel Compiler</td>
</tr>
<tr>
<td>Multicore</td>
<td>Core 2 \ i7</td>
</tr>
<tr>
<td>ILP</td>
<td>Execution Units</td>
</tr>
<tr>
<td>SIMD</td>
<td>SSE\AVX</td>
</tr>
</tbody>
</table>

*2. My Program will run just the same without any effort!*
2. My Program will run just the same without any effort! ✓ True
3. The CPU automatically makes things parallel – so I don’t need to.

The Problem – Wrong Information
Cores

The life of a program instruction

1. Instruction read from memory
2. Instruction fed to Decoder
3. Micro-ops (uops) generated
4. uops queued in RS
5. uops dispatched
6. Results sent to ROB
7. Instruction marked — all uops executed
8. Instruction sent for retirement

Retirement

Execution Units

Reservation Station

Insl Fetch Branch Pred

Decoder

Memory Sub-system

Multiple Execution units

Cores, Cores & more Cores

• Today
  - Dual core
  - Quad core
  - Multi-socket solutions

• The Future
  - 6 & 8 cores
  - many-core

• R&D
  - 80 cores ...

Performance comes from multi-core gains – but through parallelism

3. The CPU automatically makes things parallel – so I don’t need to.
The New Programming Challenge

“Everyone's happy—except perhaps for the programmers, who must now write code with threads of instructions that must be executed together—in pairs, quartets, or even larger groupings.”

Samuel K. Moore / January 2011
3. The CPU automatically makes things parallel – so I don’t need to.

False
Case Study 1
An Engine Simulator
The Simulation Environment

ECM under test

www.pishurlok.com
The Simulation Frames

- **Tick**
- **ADC Complete**
- **Interrupt Request**
- **Model**
- **Logger**
- **Script**

Frame 1: T2
Frame 2: T3
Frame 3: T4
Matlab design of the Engine Simulator
Results on 100k loop simulation

<table>
<thead>
<tr>
<th>CPU</th>
<th>No Auto-Vectorisation</th>
<th>With Auto-Vectorisation</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4</td>
<td>39.344</td>
<td>21.9</td>
<td>1.80</td>
</tr>
<tr>
<td>Core 2</td>
<td>5.546</td>
<td>0.515</td>
<td>10.77</td>
</tr>
<tr>
<td>Speedup</td>
<td>7.09</td>
<td>45.52</td>
<td>76</td>
</tr>
</tbody>
</table>
Vtune confirms reason for Speedup

<table>
<thead>
<tr>
<th>CPU EVENT</th>
<th>Without Vect</th>
<th>With Vect</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU_CLK_UNHALTED.CORE</td>
<td>16,641,000,448</td>
<td>1,548,000,000</td>
</tr>
<tr>
<td>INST RETIRED.ANY</td>
<td>3,308,999,936</td>
<td>1,395,000,064</td>
</tr>
<tr>
<td>X87_OPS RETIRED.ANY</td>
<td>250,000,000</td>
<td>0</td>
</tr>
<tr>
<td>SIMD_INST RETIRED</td>
<td>0</td>
<td>763,000,000</td>
</tr>
</tbody>
</table>

Reason for not parallelising

• OS did not support threads
  – Old RTOS
  – Incompatible runtime

• Already gained more than enough performance improvements
4. Parallel programming makes applications run slower!

The Issue – Granularity & Overhead
Granularity

4. Parallel programming makes applications run slower!

Software & Services Group, Developer Products Division

Copyright © 2011, Intel Corporation. All rights reserved. *Other brands and names are the property of their respective owners.

Optimization Notice
Overhead

4. Parallel programming makes applications run slower!
Overhead

4. Parallel programming makes applications run slower! ##

Software & Services Group, Developer Products Division
4. Parallel programming makes applications run slower!

False

True
5. No parallelism means no Errors. QED!

The Issue – Data Races and Deadlocks
Data Races and Deadlocks

- The Issue - Data races and Deadlocks

![Source Code Security Errors]

Run Static Security Analysis

Run Memory Analysis

Run Threading Analysis

Cleaner code!

5. No parallelism means no Errors. QED!  ##

Software & Services Group, Developer Products Division

Copyright © 2011, Intel Corporation. All rights reserved. *Other brands and names are the property of their respective owners.
5. No parallelism means no Errors. QED!

✓ True
6. Multicore programs don’t get faster on newer generations of CPU

The Issue – Scalability and Future Proofing
6. Multicore programs don’t get faster on newer generations of CPU

Scalability & Future Proofing

![Graph showing speedup vs. number of cores]

- Work no overhead
- Large Work with Overhead
- Small work with Overhead

Software & Services Group, Developer Products Division

Copyright © 2011, Intel Corporation. All rights reserved. *Other brands and names are the property of their respective owners.
6. Multicore programs don’t get faster on newer generations of CPU

False

True
Case Study 2

Financial Institution
Interactive Mode

Excel Front-end

Calculation engine

Database
Computer Farm
## Results

<table>
<thead>
<tr>
<th>Build</th>
<th>Option</th>
<th>Speed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Msvc</td>
<td></td>
<td>24</td>
<td>Goes to 22.5 with SSE2 changes</td>
</tr>
<tr>
<td>iCL</td>
<td>/02</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/PGO</td>
<td>16.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NO pgo</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>ADDING CILK FOR</td>
<td></td>
<td>25.5</td>
<td>CALLED 1 MILLION TIME</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cilk loop has 96 iterations</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sum product in each i elements</td>
</tr>
<tr>
<td>ARRAY NOTATION</td>
<td></td>
<td>19.5</td>
<td>USING REDUCE_ADD</td>
</tr>
<tr>
<td>Using MKL</td>
<td></td>
<td>17.2</td>
<td></td>
</tr>
<tr>
<td>With more MKL</td>
<td></td>
<td>16</td>
<td>WITH CDF (loop of 96)</td>
</tr>
<tr>
<td>SSE2</td>
<td></td>
<td>14.4</td>
<td></td>
</tr>
<tr>
<td>/pgo</td>
<td></td>
<td>13.8</td>
<td></td>
</tr>
</tbody>
</table>
Reason for not parallelising

- Code construct needed some heavy lifting/reconstructing
- All class objects instantiated at start of program
  - Done for performance reasons
  - Means lots of global\shared
- Potentially suitable loops not doing enough work
  - see how adding Cilk slowed down the code
7. We, only let our parallel expert do this, and he’s on holiday.

The Issue- Specialism
Moving to Parallel – a view from some developers

• Top 5 challenges
  – Legacy
  – Education
  – Tools
  – Fear of many cores
  – Maintainability
7. We, only let our parallel expert do this, and he’s on holiday.

✓ True
8. Writing parallel programs is expensive.

The Issue – Return on Investment
“Tip1: Just buy a faster machine!
First look at how much it will cost you to make your program parallel. If it will take say 2 months of coding, can you just buy a faster machine that will give you the speedup you want? Of course once you reach the limits of a machines speed, you are going to have to then do some parallelization.”

Dr Yann Golanski, York
8. Writing parallel programs is expensive.

False

True
9. There are too many choices – ask me again in a couple of years

The Issue – Standardisation & Perception
Standardisation

Intel® Parallel Building Blocks

- Intel® Cilk™ Plus
  Language extensions to simplify task and data parallelism
- Intel® Threading Building Blocks
  Widely used C++ template library for task parallelism
- Intel® Array Building Blocks
  Sophisticated C++ template library for data parallelism

Established Standards
- Message Passing Interface (MPI)
- OpenMP®

Compatible with Microsoft Visual Studio* and GCC*
Supports multiple operating systems and platforms

9. There are too many choices – ask me again in a couple of years

Software & Services Group, Developer Products Division

Copyright © 2011, Intel Corporation. All rights reserved. *Other brands and names are the property of their respective owners.
Perception

9. There are too many choices – ask me again in a couple of years # #

Software & Services Group, Developer Products Division

Copyright © 2011, Intel Corporation. All rights reserved. *Other brands and names are the property of their respective owners.

Optimization Notice

56
9. There are too many choices – ask me again in a couple of years

False

True
Case Study 3

VOIP Telephone Exchange
A Voice Over IP telephone Exchange
Goal

• Handle more concurrent calls
• By
  – Migrating to multicore
  – Improving the threading
  – Using VTune to profile
Results
Reason for not parallelising

- Already gained more than enough performance improvements
- Bug-fixed existing parallelism
Nine Reasons Why not to Program for Multicore

**Architectural**

- ✗ ✓ 1. Multicore is just a fad!
- ✗ ✓ 2. My Program will run just the same without any effort!
- ✗ ✓ 3. The CPU automatically makes things parallel – so I don’t need to.

**Programming Gotcha’s**

- ✗ ✓ 4. Parallel programming makes applications run slower!
- ✗ ✓ 5. No parallelism means no Errors. QED!
- ✗ ✓ 6. Multicore programs don’t get faster on newer generations of CPU

**Resource Issues**

- ✗ ✓ 7. We, only let our parallel expert do this, and he’s on holiday.
- ✗ ✓ 8. Writing parallel programs is expensive.
- ✗ ✓ 9. There are too many choices – ask me again in a couple of years
Nine Reasons Why not to Program for Multicore

Architectural
- 1. Multicore is just a fad!
- 2. My Program will run just the same without any effort!
- 3. The CPU automatically makes things parallel – so I don’t need to.

Programming Gotcha’s
- 4. Parallel programming makes applications run slower!
- 5. No parallelism means no Errors. QED!
- 6. Multicore programs don’t get faster on newer generations of CPU

Resource Issues
- 7. We, only let our parallel expert do this, and he’s on holiday.
- 8. Writing parallel programs is expensive.
- 9. There are too many choices – ask me again in a couple of years
Nine Reasons Why not to Program for Multicore

**Architectural**
1. Multicore is just a fad!
2. My Program will run just the same without any effort!
3. The CPU automatically makes things parallel – so I don’t need to.

**Programming Gotcha’s**
4. Parallel programming makes applications run slower!
5. No parallelism means no Errors. QED!
6. Multicore programs don’t get faster on newer generations of CPU

**Resource Issues**
7. We, only let our parallel expert do this, and he’s on holiday.
8. Writing parallel programs is expensive.
9. There are too many choices – ask me again in a couple of years
Thank you!

stephen.blair-chappell@intel.com
### Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options.” Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101
Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/software/products.


Copyright © 2010. Intel Corporation.

http://intel.com/software/products