Parallel Software 2.0

Wei Li
Senior Principal Engineer
Intel Corporation

CS243 Stanford 3/16/06
Agenda

Major Technological Change

Software Response

Parallel Software 2.0
Quiz

Moore’s law states which of the following roughly doubles every 2 years?

1. Frequency
2. Performance
3. Transistors
4. Transistor Density
MOORE'S LAW

from www.intel.com
Historical Driving Forces

Increased Performance via Increased Frequency

Shrinking Geometry

- **1946**: 20 Numbers in Main Memory
- **1971**: 14004 Processor, 2300 Transistors
- **2005**: 65nm, 1B+ Transistors

Increased Performance via Increased Frequency

Feature Size (um)

Frequency (MHz)


100000 10000 1000 100 10 1


0.01 0.1 1 10 100 1000 10000 100000
The Challenges

Power Limitations

Diminishing Voltage Scaling

Power = Capacitance $\times$ Voltage$^2 \times$ Frequency
also
Power $\sim$ Voltage$^3$
Energy:
The Next Frontier
Energy Efficient Performance – High End

**NASA Columbia**
- 2 MWatt
- 60 TFlops goal
- 10,240 cpus – Itanium II
- $50M
- 30,720 Flops/Watt
- 1,288 Flops/Dollar

**ASC Purple**
- 6 MWatt
- 100 TFlops goal
- 12K+ cpus – Power5
- $230M
- 7,066 Flops/Watt
- 467 Flops/Dollar

**Source:** NASA

**DATACENTER “ENERGY LABEL”**

**Computational Efficiency**

**Source:** LLNL
The Classic Tradeoff

Higher
Top Speed and Acceleration

OR

Increased
Range and Economy
The Real Challenge

Capabilities

Performance

Energy-Efficiency
Reducing Power with Voltage Scaling

- Power = Capacitance * Voltage^2 * Frequency
- Frequency ~ Voltage in region of interest
- Power ~ Voltage^3
- 10% reduction of voltage yields
  - 10% reduction in frequency
  - 30% reduction in power
  - Less than 10% reduction in performance

Rule of Thumb

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Frequency</th>
<th>Power</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>1%</td>
<td>3%</td>
<td>0.66%</td>
</tr>
</tbody>
</table>
Dual Core example of Voltage Scaling

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Frequency</th>
<th>Power</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>1%</td>
<td>3%</td>
<td>0.66%</td>
</tr>
</tbody>
</table>

Voltage scaling example:
- Voltage = 1
- Freq = 1
- Area = 1
- Power = 1
- Perf = 1

Voltage = -15%
- Freq = -15%
- Area = 2
- Power = 1
- Perf = ~1.8
Multiple cores deliver more performance per watt

- Many core is more power efficient
- Power ~ area
- Single thread performance ~ area^{**.5}

Power = \frac{1}{4}
Performance = \frac{1}{2}
Moore’s Law will provide transistors

Intel process technology capabilities

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Size</td>
<td>90nm</td>
<td>65nm</td>
<td>45nm</td>
<td>32nm</td>
<td>22nm</td>
<td>16nm</td>
<td>11nm</td>
<td>8nm</td>
</tr>
<tr>
<td>Integration Capacity</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>128</td>
<td>256</td>
</tr>
</tbody>
</table>

Use transistors for
• Multiple cores
• On-core memory (caches)
• New features (*Ts)

Multiple cores and caches address power and memory latency issues
The Dawn of Energy-Efficient Performance
Agenda

Major Technological Change

Software Response

Parallel Software 2.0
Multi-Core Platforms Demand Threaded Software

Biggest Performance Leap Since Out-of-Order Execution

Integer Performance at Introduction (normalized to 25MHz 486DX)

- Green: Single Threaded
- Orange: Multi Threaded

Pentium, Pentium II, Pentium III, Pentium 4, Pentium D, Core/1

1 Estimated based on preproduction measurements.
The Importance of Threading

• Do Nothing: Benefits Still Visible
  – Operating systems ready for multi-processing
  – Background tasks benefit from more compute resources

• Parallelize: Unlock the Potential
  – Native threads
  – Threaded libraries
  – Compiler generated threads
Multiple cores and Parallel Programming

- No change in fundamental programming model
- Synchronization and communication costs greatly reduced
  - Optimization choices may be different
  - Makes it practical to parallelize more programs
Threading for Multi-Core

- Architectural Analysis
- Introducing Threads
- Debugging
- Performance Tuning
Threading for Multi-Core

Introducing Threads

Introducing Threads

Debugging

Architecture Analysis

Call Graph

• Functional Structure
• Execution Times
• Counts

Intel® VTune™ Performance Analyzer
<table>
<thead>
<tr>
<th>Calls (341)</th>
<th>Execution Time (341)</th>
<th>Function (341)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>99.9% WinMainCRTStartup</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>99.9% WinMain</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>98.5% ParseArguments</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>98.5% Initialize</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>93.5% InitializeSG</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>97.3% LoadUSDFileInit</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>97.2% Load</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>97.2% Load</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>97.2% ExecuteReadX</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>60.3% ExecuteTransferX</td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>60.3% ProcessTransferOrderX</td>
<td></td>
</tr>
<tr>
<td>8,055</td>
<td>47.1% TransferX</td>
<td></td>
</tr>
<tr>
<td>16,212</td>
<td>34.4% ProcessGenericBlockX</td>
<td></td>
</tr>
<tr>
<td>8,085</td>
<td>33.8% ProcessModifierChainBlockX</td>
<td></td>
</tr>
<tr>
<td>12,113</td>
<td>31.8% ProcessBlockX</td>
<td></td>
</tr>
<tr>
<td>16,352,002</td>
<td>18.5% GetResourcePtr</td>
<td></td>
</tr>
</tbody>
</table>
Threading for Multi-Core

Intel® Compilers

OpenMP Loop Construct
- Creates one thread per core
- Assigns iterations to threads
24

```cpp
U32 uElapsedTime = uCurrentTime - uStartTime;
U32 i = 0;

// traverse each palette
for (i = 0; i < i1; i++)
    IFXRESULT = IFXRESULT_IFX_OK;

    // attempt to do palette updates

#ifndef TLP_IMPORT_FRONT

    // for each decoder in the component chain referenced by a palette entry,
    // traverse (i.e. decode) that decoder's content to the scene graph.
    IFXDECLAREREGLULARGLOBAL(IFXdecoderChainX, decoderChainX);
    IFXCHECKX(m_ppDecoderPalettes[i]->GetResourceId(uPaletteIndex, IID_IFXDecoderChainX, {void**} &decoderChainX));

    U32 wDecodedCount = 0;
    decoderChainX->GetDecoderCountX(uDecoderCount);

    // for the next decoder palette entry.
    for (j = 0; j < wDecodedCount; j++)
    {
        IFXDECLAREREGLULARGLOBAL(IFXdecoderX, decoderX);
        decoderX->GetDecoderX(j, decoderX);
        if (decoderX)
            // perform idling activities.
    }
#endif
```

```cpp
#pragma omp parallel for schedule(runtime)
for (indx = 0; indx < i1; indx++)
{
    U32 uPaletteIndex = pTable[index];
    // if a decoder has transferred all of its blocks and the read process has concluded
Introducing Threads

Architectural Analysis

Introducing Threads

Debugging

Performance Tuning

Thread Safety Issues

• Data Races
• Deadlocks

Intel® Thread Checker
26
Threading for Multi-Core

- Architectural Analysis
- Introducing Threads
- Debugging
- Performance Tuning

Find Contended Locks
- Most Overhead
- Largest Reduction in Parallelism

Intel® Thread Profiler
<table>
<thead>
<tr>
<th>Growing Momentum For Software Parallelization</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Activision (Ravensoft)</th>
<th>Pinnacle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adobe</td>
<td>Pixar (Renderman)</td>
</tr>
<tr>
<td>Algorithmics</td>
<td>Paradigm</td>
</tr>
<tr>
<td>Alias</td>
<td>PTC</td>
</tr>
<tr>
<td>Autodesk</td>
<td>Red Hat</td>
</tr>
<tr>
<td>Business Objects</td>
<td>SAP</td>
</tr>
<tr>
<td>Cakewalk</td>
<td>SAS</td>
</tr>
<tr>
<td>CodecPeople</td>
<td>Siebel CRM</td>
</tr>
<tr>
<td>Computer Associates</td>
<td>Signet</td>
</tr>
<tr>
<td>Corel (WordPerfect)</td>
<td>Skype</td>
</tr>
<tr>
<td>Cyberlink</td>
<td>SLB</td>
</tr>
<tr>
<td>Discreet</td>
<td>SnapStream</td>
</tr>
<tr>
<td>IBM</td>
<td>Sonic (Roxio)</td>
</tr>
<tr>
<td>id Software</td>
<td>Sony</td>
</tr>
<tr>
<td>Landmark</td>
<td>Steinberg</td>
</tr>
<tr>
<td>Macromedia</td>
<td>SunGard</td>
</tr>
<tr>
<td>Mainconcept</td>
<td>Sybase</td>
</tr>
<tr>
<td>Maxon</td>
<td>Symantec</td>
</tr>
<tr>
<td>mental images</td>
<td>Thomson</td>
</tr>
<tr>
<td>Microsoft (Office Suite)</td>
<td>THQ</td>
</tr>
<tr>
<td>Midway</td>
<td>Ubisoft</td>
</tr>
<tr>
<td>MSC</td>
<td>UGS</td>
</tr>
<tr>
<td>Novell SUSE</td>
<td>Valve</td>
</tr>
<tr>
<td>Oracle</td>
<td>Valve</td>
</tr>
<tr>
<td>Pegasus</td>
<td>Yahoo (Musicmatch)</td>
</tr>
</tbody>
</table>

Other names and brands may be claimed as the property of others.
Agenda

Major Technological Change

Software Response

Parallel Software 2.0
A New Era...

THE OLD

Performance Equals Frequency
Unconstrained Power
Voltage Scaling

THE NEW

Performance Equals IPC
Multi-Core
Power Efficiency
Microarchitecture Advancements

It is happening fast...
Multi-Core Trajectory

2006

Dual-Core

2007

Quad-Core
Future Architecture
Many More Cores

Parallel extension of IA
- Homogeneous array of cores
- Fixed-function units
- Coarse- and fine-grained data- and thread-level parallelism
- Global coherency hardware

Partitioned array
- Application domains
- Isolated communication traffic
- Fault tolerance
Parallel Software 2.0

• Ease of programming
  – Programming language, compiler, tools

• Ubiquitous
  – Consumer/wireless vs HPC/database
  – Home vs nuclear labs
  – More legacy applications

• Explosion of cores
  – 2X cores every 18 months
  – Scalable software

• Reliability

• User experience
  – vs. just raw performance

• Education
  – Mass vs elite