Game Developers Conference 2009

Who Moved the Goalposts?
The rapidly changing world of CPU's

Leigh Davies
Senior Application Engineer
INTEL
March 2009
Leigh.Davies@Intel.com

Evolving Architectures
Today's uArchitectures

2005-06
TICK
Intel® Pentium® D, Xeon®, Core™ processors
65nm

2007-08
TICK
Intel® Core™ 2 processors
45nm

2009-10
TICK
SANDY BRIDGE processors
32nm

4 different u-architectures
3 different process technologies
Radically different topologies

Why Should I Care?

• Average game development takes 12-36 months.
• Core engine technology lasts even longer.

• Intel’s Tick-Tock model == new u-architecture every 2 years.
• Average game specification will span 3-4 different u-architectures.

• “Free Lunch” performance gains are over, best performance gains come from multi-threading and an understanding of the CPU topology being used.

• Design decisions made for 1 u-architecture can significantly impact other architectures degrading potential performance.
  • Lookup table vs. calculate on the fly

With great power comes great responsibility. More important than ever to test on different hardware.
**Intel® Pentium® 4 Processor**

- 42,000K - 124,000K Transistors
- 122mm²
- 3 instructions per clock cycle
- Pipeline Stages 31
- Intel® SSE2
- SIMD Units 2 x 64-bits

**Additions**

- 64 bit capable, Intel® SSE3
- SMT up to 2 threads per core

**Intel® Core™ Micro-architecture**

- 582,000K Transistors (Dual Core), 107mm²
- 820,000K Transistors (Quad Core) 214mm²
- 4 Instructions per clock cycle; 14 Stage Pipe, Micro and Macro Fusion
- Each core pair can access shared L2 cache
- Thread per core 1
- Up to 4 Cores
- SIMD Units 3* 128 bit Single cycle SSE

**45NM**

- New instructions SSE4
- SIMD performance enhancements
Intel® Core™ i7 architecture

- 731,000K Transistors (4 Core, 8 Thread)
- 263mm²
- Simultaneous Multi-Threading/Threads per core 2
- Up to 4 Cores in Desktop

- Additional Caching Hierarchy
- 4 instructions per clock cycle
- 16 Stage Pipe, Enhanced Micro and Macro Fusion
- Deeper Buffers
- SIMD Units 3* 128 bit Single cycle SSE
- Macrofusion™ in both 32-bit and 64-bit modes
Familiar performance tuning guidelines

Many new features introduced you get for free
- Better branch prediction + faster mispredict correction
- Improvements in unaligned loads + cache-line splits
- Improvements on store forwarding
- Memory bandwidth increase
- Reduced memory latency
- Etc...

No large differences in tuning guidelines
- Still use Intel® 64 and IA-32 Architecture Optimization
- Reference Manual:
  http://www.intel.com/products/processor/manuals/

Common Core

- Same core for server, desktop, mobile
- Each core supports SMT
- Architecture improvements over Core 2

Modular Uncore

- Differentiation in the “Uncore”
  - # of cores
  - # of memory channels
  - # of QPI links
  - Size of cache
  - Type of memory

Best performance comes from properly threading your game. Amdahl will get you every time.
Bigger better memory system

- 32kB L1 Data Cache
- 32kB L1 Inst. Cache
- Unified 256kB 8 Way L2 Cache 10 Cycle
- 32kB L1 Data Cache
- 32kB L1 Inst. Cache

Inclusive L3 Cache (35-40Cycles)
Provides benefit of an on-die snoop filter
Built to vary size with varied core counts
Built to easily increase L3 size in future parts

Relative Memory Latency
Stream Bandwidth - Mbytes/Sec (Triad)

Your users have SSE -- use it

<table>
<thead>
<tr>
<th>Intel® SSE</th>
<th>Intel® SSE2</th>
<th>Intel® SSE3</th>
<th>Intel® SSSE3</th>
</tr>
</thead>
</table>

- 70 instructions
  - Single-Precision Vectors
  - Streaming operations
- 144 instructions
  - Double-Precision Vectors
  - 128-bit vector integer
- 13 instructions
  - Complex Arithmetic
- 12 instructions
  - Decode

- 47 instructions
  - Video Accelerators
  - Graphics building blocks
  - Coprocessor Accelerators

95.72% of system on the Valve Steam* survey report SSE2+
All Dual core system support SSE3 or above.
Hardware improvements to help compilers

*Steam Hardware Survey: February 2009
http://store.steampowered.com/hwsurvey
Threading performance considerations

Three very different memory/cache designs

Limit Cache thrashing on shared resources;
Consider Cache blocking – use smaller sets of data in inner loops
Consider streaming stores to avoid cache pollution

Minimise communication outside of last level shared cache
• Using Core 2 as an example
  • L1 hit ~ 3 cycle access time
  • L1 miss, L2 hit, ~14 cycle access time
  • L1 miss, L2 miss
    152 ns memory latency for a 3.0 GHz processor with a 1333 MT/s bus
    152 ns is 456 processor clocks (core clock cycles)
False Sharing

Can occur when 2 threads access data that falls into the same cache line

```cpp
class DebugInfo{
public:
    int m_bDebugEnable;
    int m_iDrawCalls;
    int m_iParticleSystems;
    int m_iEntitiesProcessed;
};
```

64-byte aligned global and static;
- __declspec(align(64))

When each thread needs to use its own copy of variables;
- Use modifier __declspec(thread)
- Or use stack variables where reasonable

For dynamic allocation, avoid using malloc and new
- Use routines that allocate from separate 64 byte aligned per thread pools

Basic task queue

Real sharing as both threads access queue data structure
Synchronisation required;
• ~10-1000’s cycles depending if context switch triggered
Hot cache data isn’t used

Cache ignorance will punish your threads
Ensure task queues are Cache aware

New Tasks should be inserted into the parent queue
Queue should work Last in First out
Task stealing should happen only when thread is idle

Similar ideas used in
Intel® Thread Building Blocks

Intel® Hyper-Threading can benefit performance

Also known as Simultaneous Multi-Threading (SMT)
- Run 2 threads at the same time per core
Shares Resources(Cache, Frontend, Execution Units)
Improves Core CPI (Clockticks per Instruction)
Potentially degrades Thread CPI
SMT is better on Intel® Core i7™

Intel® Core i7™ u-architecture advantages
- Larger caches
- Massive memory BW
- 4-wide execution engine
- More execution resources

<table>
<thead>
<tr>
<th></th>
<th>Intel® Core™</th>
<th>Intel® Core™ i7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reservation Station</td>
<td>32</td>
<td>36</td>
</tr>
<tr>
<td>Load Buffers</td>
<td>32</td>
<td>48</td>
</tr>
<tr>
<td>Store Buffers</td>
<td>20</td>
<td>32</td>
</tr>
<tr>
<td>Concurrent uOps</td>
<td>96</td>
<td>128</td>
</tr>
</tbody>
</table>

Performance Gain SMT enabled vs disabled

Floating Point is based on SPECfp_rate_base2006* estimate
Integer is based on SPECint_rate_base2006* estimate

SPEC, SPECint, SPECfp, and SPECrate are trademarks of the Standard Performance Evaluation Corporation.
For more information on SPEC benchmarks, see: http://www.spec.org

Design defensively for SMT

- Duration of individual Tasks is non-deterministic
- Finer grained task parallelism reduces/eliminates scheduling issues
- Don’t spin threads unnecessarily, you consume resources on the shared logical processor
  - If you need to, use a SMT aware spin-wait loop (uses MWAIT)
- Ensure cache algorithms account for shared resource
  - Check for L1 and L2 Cache Hit Rate in VTune
Don't assume CPU topology

Wrong way to get number of cores per package:
- Only use CPUID leaf 4
- CPUID.4.EAX[32:16]
  This used to work by accident
- Getting wrong enumeration and
  affinitizing is a double whammy!

CPUID (don’t cut corners!)

Right way to detect number of Cores per package:
Copy CPU Topology Enumeration whitepaper exactly
http://softwarecommunity.intel.com/articles/eng/3887.htm
- CPUID can detect logical processors
- CPUID can detect shared caches
- CPUID reports cache sizes

Easy to get it wrong - copy exactly
Thread affinity is risky

Make no assumptions!!!!

- Affinitization masks can vary per OS versions, 32-bit or 64-Bit and service packs!
- Does not consider the software environment
  - Middleware within your application
  - Co-existing with other (possibly affinitized) applications
- Short term gain versus many headaches later
  - Increases the danger of deadlocks
  - Game might not run on future processors due to affinitization

OS Thread Scheduling

- Need to understand how Windows* schedules the threads to run.
- Scheduling is preemptive, priority-based and round-robin.
- Scheduler tries to keep a thread on its ideal processor/node to avoid perf degradation of cache/NUMA-memory
- Boosts for GUI foreground, waiting for event
- A quantum is the amount of time a thread runs before Windows* checks for another.

*Other names and brands may be claimed as the property of others
Understand the Windows scheduler

- Each time the clock-interrupt routine is called, a fixed value is deducted from the thread quantum.
- The clock interval for most x86 uni-processors is between 10-15 milliseconds, by default a thread runs for 2 clock intervals.
- A Switch occurs when a thread's quantum value reaches 0 or an event triggers a voluntary switch (critical section, mutex or event).
- It can be beneficial to execute a short spin-lock on resources that are likely to be held for only a short time to reduce unintentional context switching.

- **InitializeCriticalSectionAndSpinCount**
  - Optimized for Hyper-Threading Technology
  - Avoids context switch

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

Thread Information Functions

Code snippet to enumerate all threads within your process

```c
hThreadSnap = CreateToolhelp32Snapshot( TH32CS_SNAPTHREAD, 0 );
t32.dwSize = sizeof(THREADENTRY32);
if( !Thread32First( hThreadSnap, &te32 ) ) {
    CloseHandle( hThreadSnap );
    return( FALSE );
}
Do { 
    if( te32.th32OwnerProcessID == dwOwnerPID ) { 
        HANDLE hThread = OpenThread(THREAD_ALL_ACCESS,FALSE,te32.th32ThreadID);
        CloseHandle( hThread );
    }
} while( Thread32Next(hThreadSnap, &te32 ) );
```

Vista Windows API additions

**QueryThreadCycleTime**
- Retrieves the cycle time for the specified thread, Do not attempt to convert the CPU clock cycles returned to elapsed time

**QueryProcessCycleTime**
- Retrieves the sum of the cycle time of all threads of the specified process.

**QueryIdleProcessorCycleTime**
- Retrieves the cycle time for the idle thread of each processor in the system
Intel® Advanced Vector Extensions
Sandy Bridge: Moves SSE to 256 bits

**New Instructions**

- **Wider Vectors**
  - Increased from 128 bit to 256 bit

- **Works on either**
  - whole 256-bits or lower 128-bits

**Benefits**

- Up to 2x Peak FLOPs Output
- A drop-in replacement for all existing scalar/128-bit SSE instructions
- Efficient Data Access
- Smaller Code Size
- Parallel Operations

**Enhanced Data Rearrangement**
New 256 bit Primitives for Data Permutest

**Three Operand**
Non Destructive Syntax
Efficient and Extensible

- YMM0
- XMM0

- 256 bits (2010)
- 128 bits (1999)
Larrabee: Block Diagram

- Multiple processor cores
- Fully coherent caches
- Dedicated logic blocks where needed

**Larrabee continues the trend seen in CPU's**

**Summary**
• Desktop CPU architecture is evolving at a very fast rate.
• Applications need to be aware of CPU topology.
• Correctly enumerating hardware is hard, but we have good whitepapers available.
• Have a repeatable workload (i.e. demo or benchmark mode).
• Have a simple set of performance metrics, use this on a regular basis to test performance anomalies, just as you would test for bugs.
• Use the right tools.

www.intel.com/software/visualadrenaline
For More Information

http://www.intel.com/software/gdc

Contact info

See Intel at GDC:
- Intel Booth at Expo, North Hall
- Intel Interactive Lounge – West Hall 3rd floor

Take a collateral DVD
- Here in the room!
- Intel Booth or Interactive Lounge

More Intel @ GDC

Thursday, March 26
- Taming Your Game Production Demons: the Offset approach
  - 3:00 PM – 4:00 PM in Room 2011, West Hall
- Optimizing Game Architectures with Intel Threading Building Blocks
  - 4:30 PM – 5:30 PM in Room 2011, West Hall
Last of Intel @ GDC

Friday, March 27
- Procedural and Multi-Core Techniques to take Visuals to the Next Level
  - 9:00 AM – 10:00 AM in Room 2010, West Hall
- SIMD Programming on Larrabee: A Second Look at the Larrabee New Instructions (LRBni) in Action
  - 9:00 AM – 10:00 AM in Room 135, North Hall
- Rasterization on Larrabee: A First Look at the Larrabee New Instructions (LRBni) in Action
  - 10:30 AM – 11:30 AM in Room 3002, West Hall

Questions
Risk Factors
This presentation contains forward-looking statements. All statements made that are not historical facts are subject to a number of risks and uncertainties, and actual results may differ materially. Please refer to our most recent Earnings Release and our most recent Form 10-Q or 10-K filing available on our website for more information on the risk factors that could cause actual results to differ.