Outline

• 4 Historic Opportunities
• What’s Going on at Intel
• Which Parallelism Opportunity
• Wouldn’t it be nice if (Challenges)
• Education and Computing Fundamentals
• Questions
Opportunity #1: Highly Portable, Parallel Software

- Broad range of systems (servers, desktops, laptops, MIDs, smart phones,...) converging to...
  - A single framework with parallelism and a selection of CPU’s and specialized elements
  - Energy efficiency and Performance are core drivers
  - Must become “forward scalable”

Parallelism becomes widespread – all software is parallel

Create standard models of parallelism in architecture, expression, and implementation.
Software with **forward scalability** can be moved unchanged from $N \rightarrow 2N \rightarrow 4N$ cores with continued performance increases.
Opportunity #2: Major Architectural Support for Programmability

• Single core growth and aggressive frequency scaling are weakening competitors with other types of architecture innovation

Architecture innovations for functionality – programmability, observability, GC, ... are now possible

Don’t ask for small incremental changes, be bold and ask for LARGE changes... that make a LARGE difference
New Golden Age of Architectural Support for Programming?

- Mid 60’s To Mid 80’s
- Mid 80’s To Mid 200x
- Terascale Era

- Language Support Integration
- Ghz Scaling Issue Scaling
- Programming Support Parallelism System Integration
Opportunity #3: High Performance, High Level Programming Approaches

- Single chip integration enables closer coupling (cores, caches) and innovation in intercore coordination
  - Eases performance concerns
  - Supports irregular, unstructured parallelism

Forward scalable performance with good efficiency may be possible without detailed control.

Transactional, functional, side-effect free, declarative, object-oriented, and many other high level models will thrive.
### Huge Opportunity: Manycore != SMP on Die

<table>
<thead>
<tr>
<th>Parameter</th>
<th>SMP</th>
<th>Tera-scale</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-die Bandwidth</td>
<td>12 GB/s</td>
<td>~1.2 TB/s</td>
<td>~100X</td>
</tr>
<tr>
<td>On-die Latency</td>
<td>400 cycles</td>
<td>20 cycles</td>
<td>~20X</td>
</tr>
</tbody>
</table>

- Less locality sensitive; Efficient sharing
- Runtime techniques more effective for dynamic, irregular data and programs
- Can we do less tuning? And program at a higher level?
Opportunity #4: Parallelism can Add New Kinds of Capability and Value

- Additional core and computational capability will be available on-chip, can be exploited at modest additional cost
  - Single-chip design enables enhancement at low cost

Deploy Parallelism to improve application, software, and user experience

Traditional: user interface, visual computing, large data analysis, machine learning, activity inference

New: Security – network and platform monitoring
Software – health monitoring, debugging and lifecycle monitoring
Performance - Dynamic tuning and Customization
Parallelism for the Masses
“Opportunities and Challenges”

Intel Research Log-Based Architectures

Problem Statement
- Hypothesis: Architectural support can significantly improve the performance of online software correctness-checking tools (lifeguards).
- Approach: As application runs, hardware captures execution log; log is transported via processor cache and consumed by lifeguard running on separate core.

Research Activities
- Design of architecture support for logging.
- Design/evaluation of log compression.
- Port of lifeguards to LBA.
- Linux system software implementation.
- Design of acceleration hardware.
- Detailed performance evaluation.
- Design for monitoring parallel applications (ongoing).

Performance Comparisons
- Binary instrumentation: 20-40X slowdown
- LBA baseline arch: 3.2-4.3X slowdown
- LBA w/acceleration: 2-50% slowdown

Joint w/ CMU
What’s going on at Intel
Intel Ct  --  C for Throughput Computing

- Ct adds parallel collection objects & methods to C++
  - Works with standard C++ compilers (ICC, GCC, VC++)
- Ct abstracts away architectural details
  - Vector ISA width / Core count / Memory model / Cache sizes
- Ct forward-scales software written today
  - Ct is designed to be dynamically retargetable to SSE, AVX, Larrabee, and beyond
- Ct is deterministic
  - No data races

See whatif.intel.com
The application problem

The work of the domain expert
- Semantic correctness
- Constraints required by the application

Concurrent Collections Spec

The work of the tuning expert
- Architecture
- Actual parallelism
- Locality
- Overhead
- Load balancing
- Distribution among processors
- Scheduling within a processor

Mapping to target platform

Supports serious separation of concerns:

The domain expert does not need to know about parallelism.

The tuning expert does not need to know about the domain.

http://whatif.intel.com
Parallelism for the Masses
“Opportunities and Challenges”
Intel: Making Parallel Computing Pervasive

**Intel Tera-scale Research**

- Academic Research UPCRCs
  - Academic research seeking disruptive innovations 7-10+ years out

**Software Products**

- Enabling Parallel Computing
  - Joint HW/SW R&D program to enable Intel products 3-7+ in future

**Community and Experimental Tools**

- Wide array of leading multi-core SW development tools & info available today
  - TBB Open Sourced
  - STM-Enabled Compiler on Whatif.intel.com
  - Parallel Benchmarks at Princeton’s PARSEC site

**Multi-core Education**

- Multi-core Education Program
  - 400+ Universities
  - 25,000+ students
  - 2008 Goal: Double this

- Intel® Academic Community
  - Threading for Multi-core SW community
  - Multi-core books

---

**Parallelism for the Masses**

“Opportunities and Challenges”
Multicores and Manycores

- Dunnington – x86, 6 cores
  - 16MB L3, 12 cores/socket
- Nehalem (new uArch),
  - 4 cores, 4-wide, IMC+QPI
  - 2-way SMT

- LRB x86+Graphics
  - Large # of cores
  - Coherent Caches
  - Shared L2
• Deliver Tera-scale performance
  – TFLOP @ 62W, Desktop Power, 16GF/W
  – Frequency target 5GHz, 80 cores
  – Bi-section B/W of 256GB/s
  – Link bandwidth in hundreds of GB/s
• Prototype two key technologies
  – On-die interconnect fabric
  – 3D stacked memory
• Develop a scalable design methodology
  – Tiled design approach
  – Mesochronous clocking
  – Power-aware capability
Terascale Chip Research Questions

• **Cores**
  - How many? What size?
  - Homogenous, Heterogeneous
  - Programmable, Configurable, Fixed-function

• **Chip-level**
  - Interconnect: Topology, Bandwidth
  - Coordination
  - Management

• **Memory Hierarchy**
  - # of levels, sharing, inclusion
  - Bandwidth, novel Technology
  - Integration/Packaging

• **I/O Bandwidth**
  - Silicon-based photonics
  - Terabit links

**Manycore Chips (circa. 2015)?**

**Source:** CTWatchQuarterly, Feb 2007
TeraScale Computing Research

**Microprocessor**

- **Essential**
  - Scalable memory
  - Multi-core architectures
  - Specialized cores
  - Scalable fabrics
  - Energy efficient circuits

- **Complementary**
  - Si Process Technology
  - CMOS VR
  - Resiliency

**Platform**

- **Essential**
  - 3D Stacked Memory
  - Cache Hierarchy
  - Virtualization/Partitioning
  - Scaleable OS’s
  - I/O & Networking

- **Complementary**
  - CMOS Radio
  - Photonics
  - EESA

**Programming**

- **Essential**
  - Speculative Multithreading
  - Transactional memory
  - Workload analysis
  - Compilers & Libraries
  - Tools

- **Complementary**
  - Diamond media indexing
  - Activity Recognition
  - Usage Models

**100+ Projects Worldwide**
Which Parallelism Opportunity?
Which Parallelism Opportunity?

- Data Center, Desktop/Laptop, Mobile/embedded

Internet  
DataCenter  
& HPC  

Single Chip Parallelism
Unified IA and Parallelism Vision – a Foundation Across Platforms

Integration Options
Example: Graphics

Notebook, Desktop and Server

Parallelism for the Masses
“Opportunities and Challenges”
Two Students in 2015

I’ve got an Intel thousand core with 100,000 cores!

How many cores does your computer have?

End Users don’t care about core counts; they care about capability.
Chip Real Estate and Performance/Value

- Tukwila – upcoming Intel Itanium processor
  - 4 cores, 2B transistors, 30MB cache
  - 50/50 split on transistors

- 1% of chip area = 1/3 MB = 30M transistors
- 0.1% of chip area = 1/30 MB = 3M transistors

- How much performance benefit do these elements provide?

- What incremental performance benefit would you expect for the last core in a 100-core? 1000-core?
Which Programming Community?

- “Quick functionality, adequate performance”
  - Matlab, Mathematica, R, SAS, etc.
  - VisualBasic, PERL, Python, Ruby|Rails
- “Mix of productivity and performance”
  - Java, C# (Managed languages + rich libraries)
- “Performance Programmers”
  - C++ and STL
  - C and Fortran
- HPC has focused nearly exclusively on “Performance Programmers”
A different perspective...

• Guy: “...the parallel part isn’t what makes this hard...”

• Andrew: “... it’s the ugliness of the low level machine details and programming models that makes both parallelism and concurrency difficult...”

• Can we create a world in which the upper layer programmers (Joe’s) can write lots of 100-fold parallel programs?

Don’t forget to focus on high productivity programmers. They write the most code (and the killer apps!) and are least tolerant of low-level abstractions and tedium.
Parallelism for the Masses Focus

- Clients are where the action is – smartphones, laptops, and interaction with the physical world
- Easy to use parallelism for the rapid innovators and moderate programming effort crowd
- Incremental performance improvement matters – more cores translate into higher performance – but program efficiency is less critical

=> 100M’s of systems, M’s of programmers
HPC... what have we learned about parallelism and how to approach this...

- Large-scale – extreme scalability is possible, and typically comes with scaling of problems and data
- Portable expression of parallelism matters
- Multi-version programming (algorithms and implementations) is a good idea
- High level program analysis is a critical technology
- Autotuning is a good idea
- Working with domain experts is a good idea

- Locality is hard, modularity is hard, data structures are hard, Efficiency is hard...

- Of course, this list is not exhaustive....
HPC... lessons not to learn ...

- Programmer effort doesn’t matter
- Hardware efficiency matters
- Low-level programming tools are acceptable
- Low-level control is an acceptable path to performance
- Horizontal locality and control of communication is the critical performance concern
Wouldn’t it be nice if? (Challenges)

• Programmers could write parallel code with minimal effort
  – Perhaps only a simple limited model
  – Serve “fast prototyping” programmers (but all tiers of programmers); a short time to working, parallel code...
  – Reasonable performance (and “forward scalability”) delivered by the implementation

• Programmers could migrate gracefully from this high level view to lower-levels of specification
  – To improve performance
  – To increase control or provide information to the underlying runtime system
  – Parallelism, Locality, “Hints”, etc.
Wouldn’t it be nice if?

- There were an easy, modular way to express locality.
- Computation = Algorithms + Data structures
- Efficient Computation = Algorithms + Data Structures + Locality
- How do we associate these in our programming models? Explicit, Implicit, otherwise?
  - A + DS + L
  - (A + DS) + L
  - A + (DS + L)
  - (A + L) + DS
Wouldn’t it be nice if?

- Parallel program composition preserved correctness of the composed parts
- Interactions when necessary were controllable at the programmer’s semantic level
- These interactions were supported efficiently by the hardware

- The TM vision, but there are still many more approaches to explore
Wouldn’t it be nice if?

• Parallel Data structures could be made correct under Parallel program composition
• Interactions when necessary were controllable at the programmer’s semantic level
• These interactions were supported efficiently by the hardware

• The lock-free vision?; or generalized version for arbitrary user data structures
Education and the Foundation of Computing
Parallelism is in all computing systems, and should be front and center in education
- An integral part of early introduction and experience with programming
- An integral part of early algorithms and theory
- An integral part of software architecture and engineering

Observation: No biologist, physicist, chemist, or engineer thinks about the world with a fundamentally sequential foundation.

The world is parallel! We can design large scale concurrent systems of staggering complexity.
Are All Applications Parallel?

Today

# of Applications Growing Rapidly

Manycore Era

100%

??%
• Four Historic Opportunities
  – #1: Highly-portable Parallel Software
  – #2: Major Architecture Support for Programmability
  – #3: High-level, High Performance Parallel Programming
  – #4: Parallelism Add new kinds of capability and value

• What Parallelism Opportunity?
  – All kinds of clients need parallelism; how to reach the largest community of programmers?

• Some ambitious, but reasonable “Challenges”
  – Programmability, Access to Performance, Portable Locality
Wanted: Breakthroughs in Parallel Programming

Questions?