Parallelism for the Masses: Opportunities and Challenges

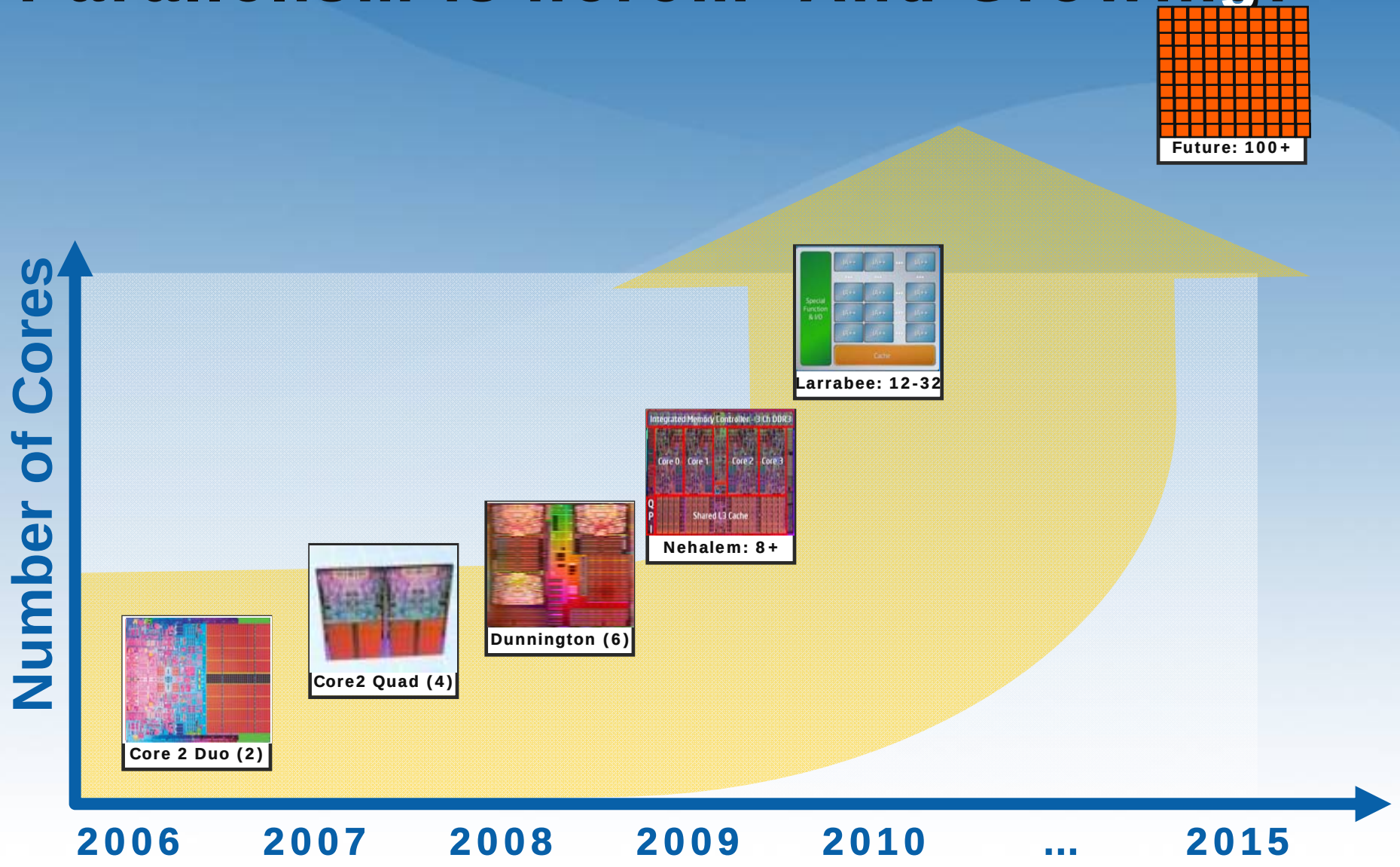
Andrew A. Chien
Vice President of Research
Intel Corporation

Carnegie Mellon University
Parallel Thinking Seminar
October 29, 2008

Outline

- Is Parallelism a crisis?
- Opportunities in Parallelism
- Expectations and Challenges
- Moving Parallel Programming Forward
- What's Going on at Intel
- Questions

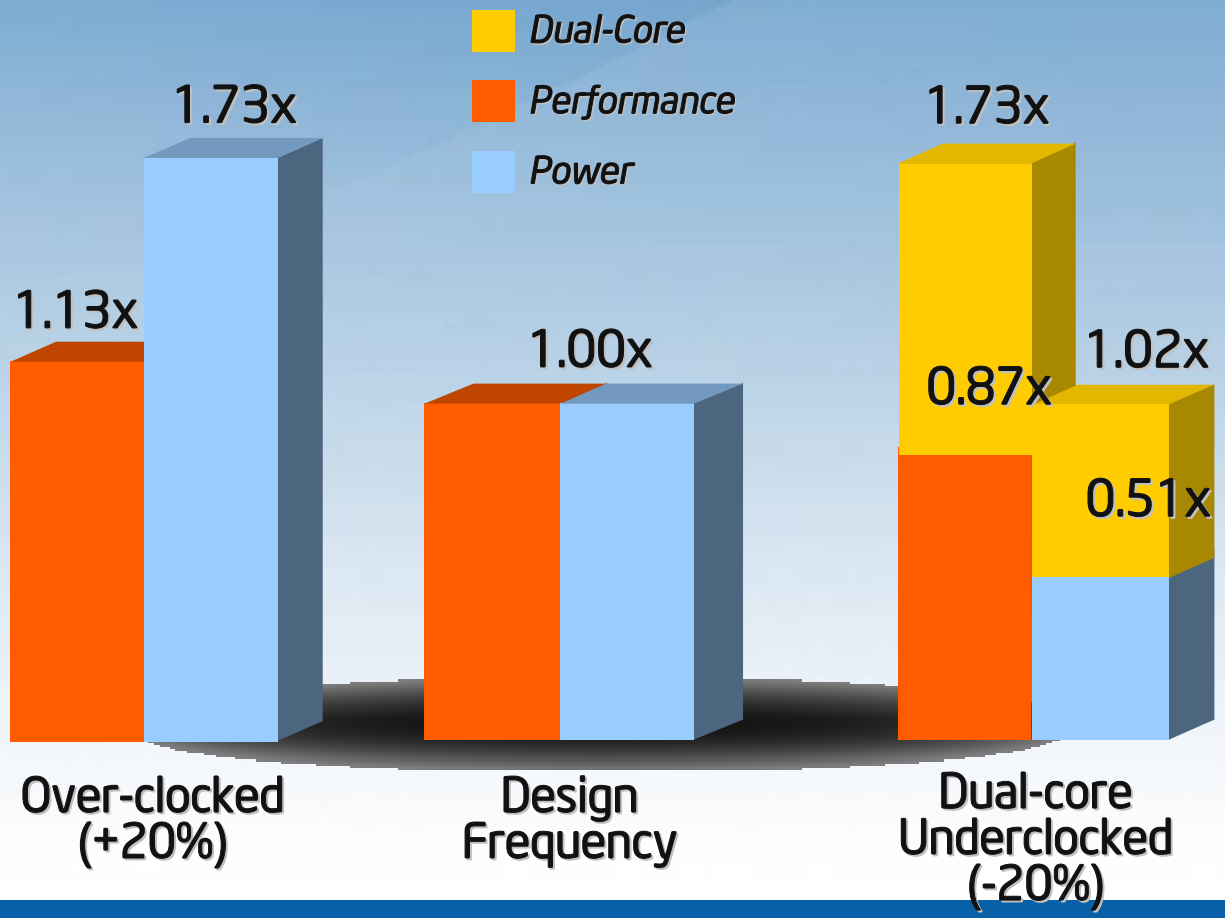
Parallelism is here… And Growing!



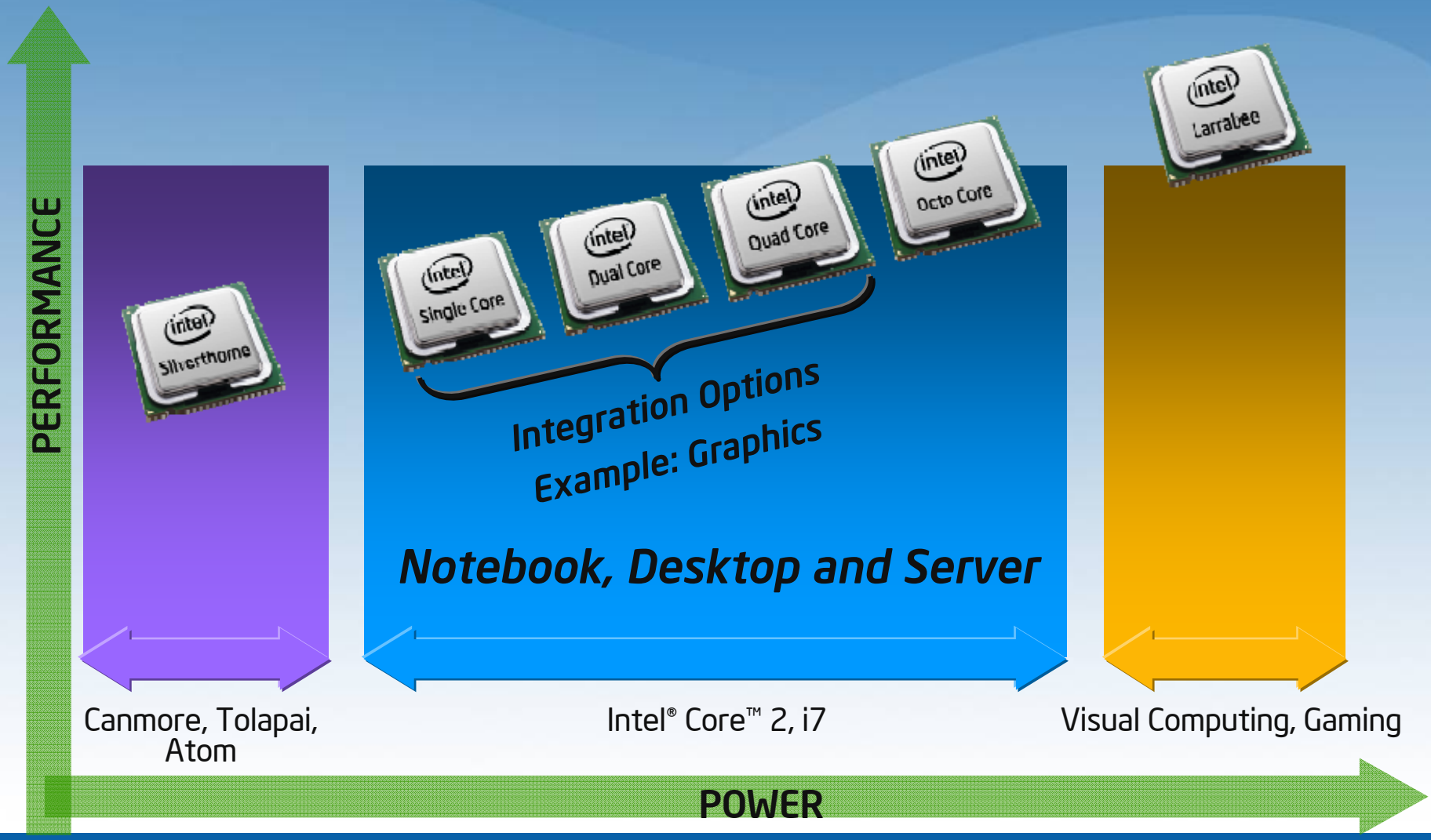
Q. Is parallelism a crisis?

A. Parallelism is an opportunity.

Parallelism is a key driver for Energy and Performance



Unified IA and Parallelism Vision – a Foundation Across Platforms

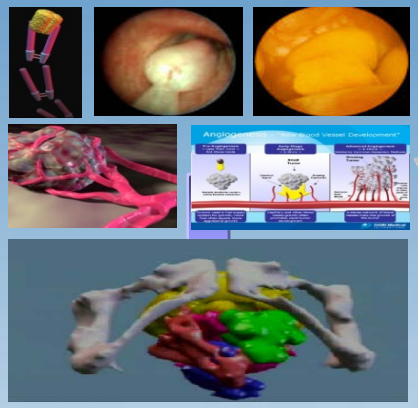


Opportunity in Low-power Computing

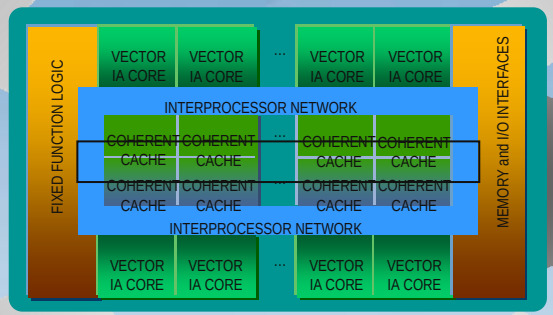
10x Lower Power

Opportunity in Highly Parallel Computing

10x Higher Performance

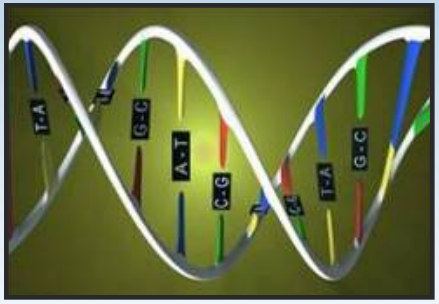


Visual Computing



Financial Modeling

Gaming, Entertainment



Biological Modeling

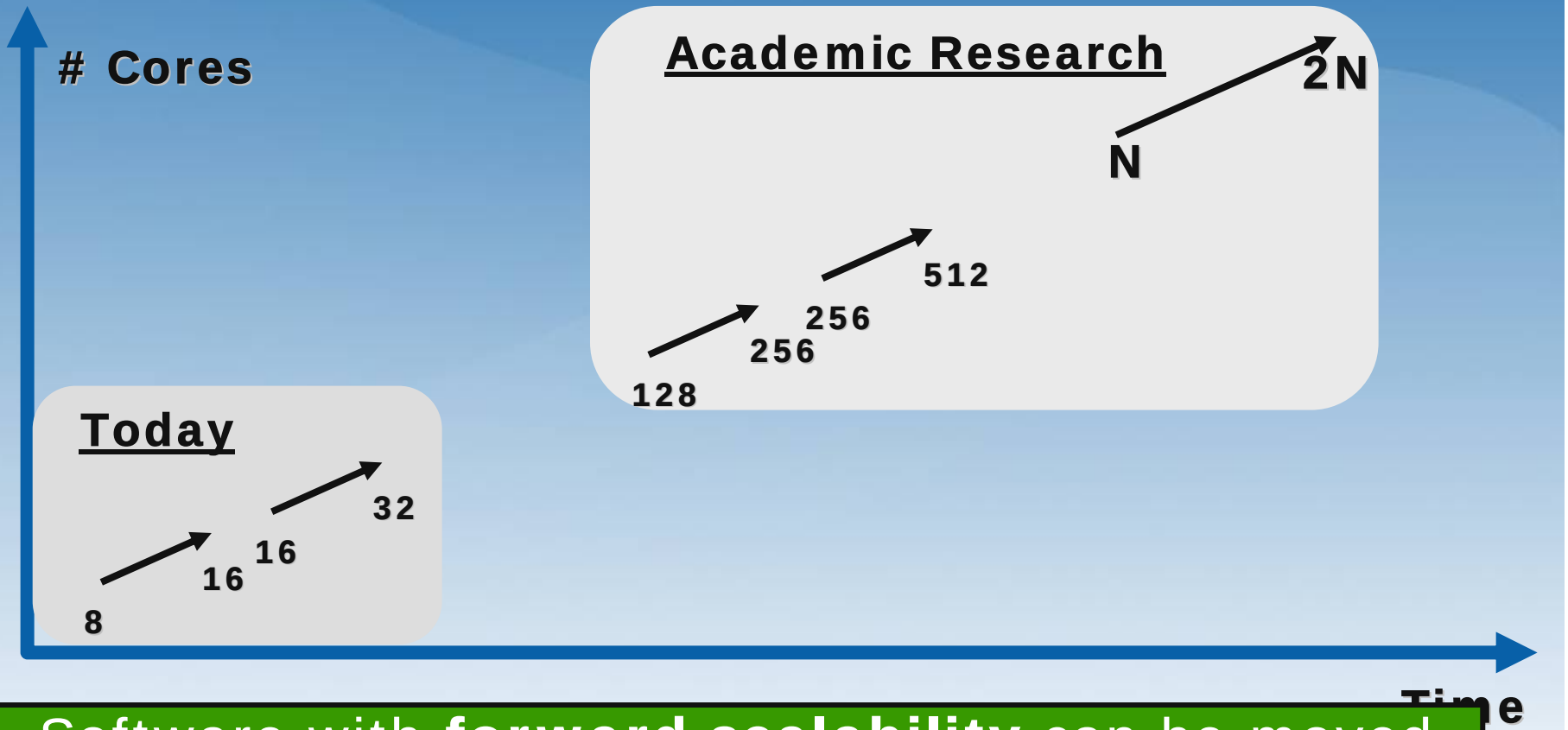
Opportunity #1: Highly Portable, Parallel Software

- All computing systems (servers, desktops, laptops, MIDs, smart phones, embedded…) 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.

“Forward Scalable” Software



Software with **forward scalability** can be moved unchanged from N -> 2N -> 4N cores with continued performance increases.

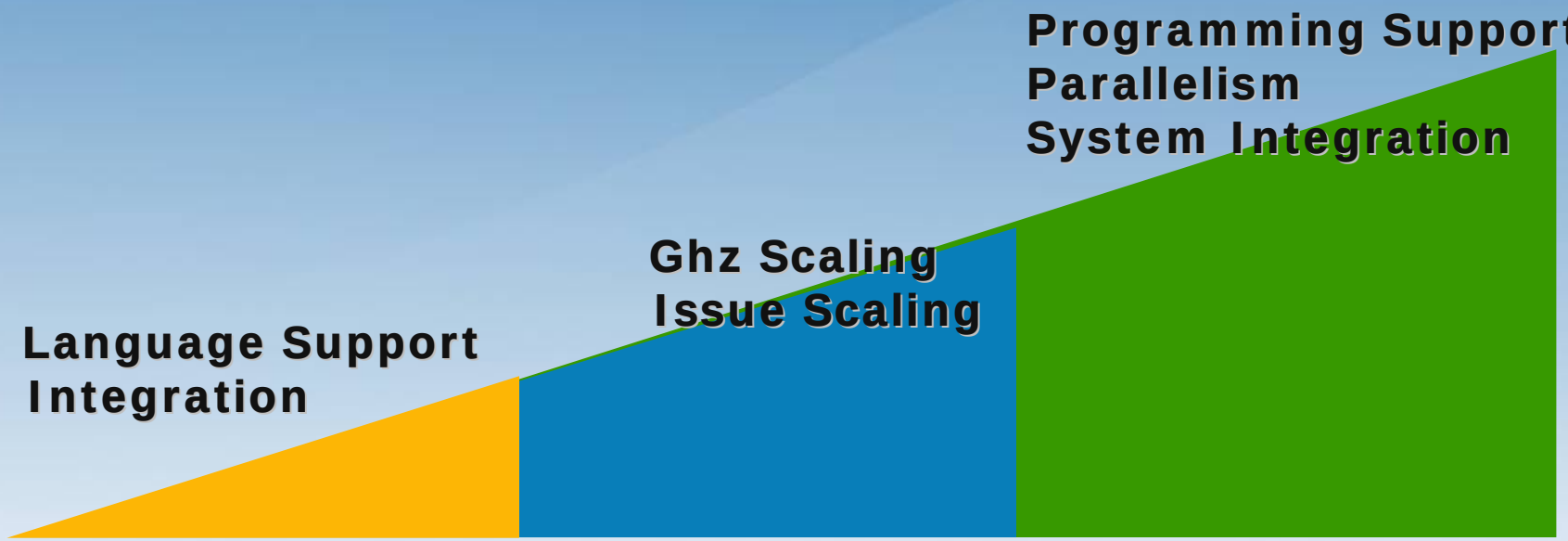
Opportunity #2: Major Architectural Support for Programmability

- Single core growth and aggressive frequency scaling are weakening competitors to 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?



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.

Functional, declarative, transactional, object-oriented, dynamic, scripting, and many other high level models will thrive.

Manycore != SMP on Die

Parameter	SMP	Tera-scale	Improvement
On-die Bandwidth	12 GB/s	~1.2 TB/s	~100X
On-die Latency	400 cycles	20 cycles	~20X

- 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 can be available on-chip
  - Single-chip design enables enhancement at low cost
  - Integration enables close coupling
  - Security: taintcheck, invariant monitoring, etc.
  - Robustness: race detection, invariant checking, etc.
  - Interface: sensor data processing, self-tuning, activity inference

Deploy Parallelism to enable new applications, software quality, and enhance user experience

Expectations and Challenges

Two Students in 2015

Mine's an Intel 1,000 core with 64 Out-of-Order 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 – next generation Intel Itanium processor
  - 4 cores, 2B transistors, 30MB cache
  - 50% cache, 50% logic
  - 1% of chip area = 30M transistors = 1/3MB
  - 0.1% of chip area = 3M transistors = 1/30MB

- How much performance benefit?
- What incremental performance benefit should you expect for the last core in a 100-core? 1000-core?

Incremental performance benefit or "forward scalability", not efficiency should be the goal.

Key Software Development Challenges

- New functionality
- Productivity
- Portability
- Performance, Performance Robustness
- Debugging/Test
- Security
- Time to market

⇒ Software Development is Hard!

⇒ Parallelism is critical for performance, but must be achieved in conjunction with all of these requirements…

HPC: What we have learned about Parallelism

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

Move beyond conventional wisdom "parallelism is hard", based on these lessons. Parallelism can be easy.

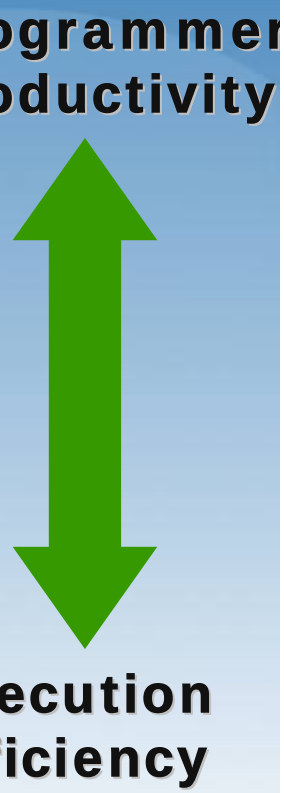
Moving Parallel Programming Forward

Spending Moore's Dividend (Larus)

- 30 year retrospective, analyzing Microsoft's experience
- Spent on --
  - New Application Features
  - Higher level programming
    - Structured programming
    - Object-oriented
    - Managed runtimes
  - Programmer productivity (decreased focus)

Today, most programmers do not focus on performance

- Productivity: "Quick functionality, adequate performance"
  - Matlab, Mathematica, R, SAS, etc.
  - VisualBasic, PERL, Javascript, Python, Ruby|Rails
- Mixed: "Productivity and performance"
  - Java, C# (Managed languages + rich libraries)
- Performance: "Efficiency is critical" (HPC focus)
  - C++ and STL
  - C and Fortran
- How can we enable productivity programmers write 100-fold parallel programs?



Parallelism must be accessible to productivity programmers.

Challenge #1: Can we introduce parallelism in the Productivity Layer?

- Enable productivity programmers to create large-scale parallelism with modest effort
- A simple models of parallelism mated to productivity languages
  - Data and collection parallelism, determinism, functional/declarative
  - Parallel Libraries
- Generate scalable parallelism for many applications
- Exploit with dynamic and compiled approaches

- => May be suitable for introduction to programming classes

Challenge #2: Can we compose parallel programs safely? (and easily)

- Parallel program composition frameworks which preserved correctness of the composed parts
  - Language, execution model, a/o tools
- Interactions when necessary are controllable at the programmer's semantic level (i.e. objects or modules)
- Composition and interactions supported efficiently by hardware

- Fulfill vision behind TM, Concurrent Collections, Concurrent Aggregates, Linda, Lock-free, Race-free, etc.

Challenge #3: Can we invent an easy, modular way to express locality?

- Describe data reuse and spatial locality
- Descriptions compose
- Enable software exploitation
- Enable hardware exploitation

- Computation = Algorithms + Data structures

- Efficient Computation = Algorithms + Data Structures + Locality
  - A + DS + L
  - (A + DS) + L
  - A + (DS + L)
  - (A + L) + DS

Challenge #4: Can we raise the level of efficient parallel programming?

- HPC model = Lowest Level of abstraction
  - Direct Control of resource mapping
  - Explicit control over communication and synchronization
  - Direct management of locality
- Starting from HPC models, how can we elevate?

Parallelism Implications for Computing Education in 2015 (and NOW!)

- Parallelism is in all computing systems, and should be front and center in education
  - An integral part of early introduction and experience with programming (simple models)
  - An integral part of early theory and algorithms (foundation)
  - 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?

What's going on at Intel

Universal Parallel Computing Research Centers

Catalyze breakthrough research enabling ***pervasive*** use of parallel computing

Parallel Programming
Languages, Compilers, Runtime, Tools

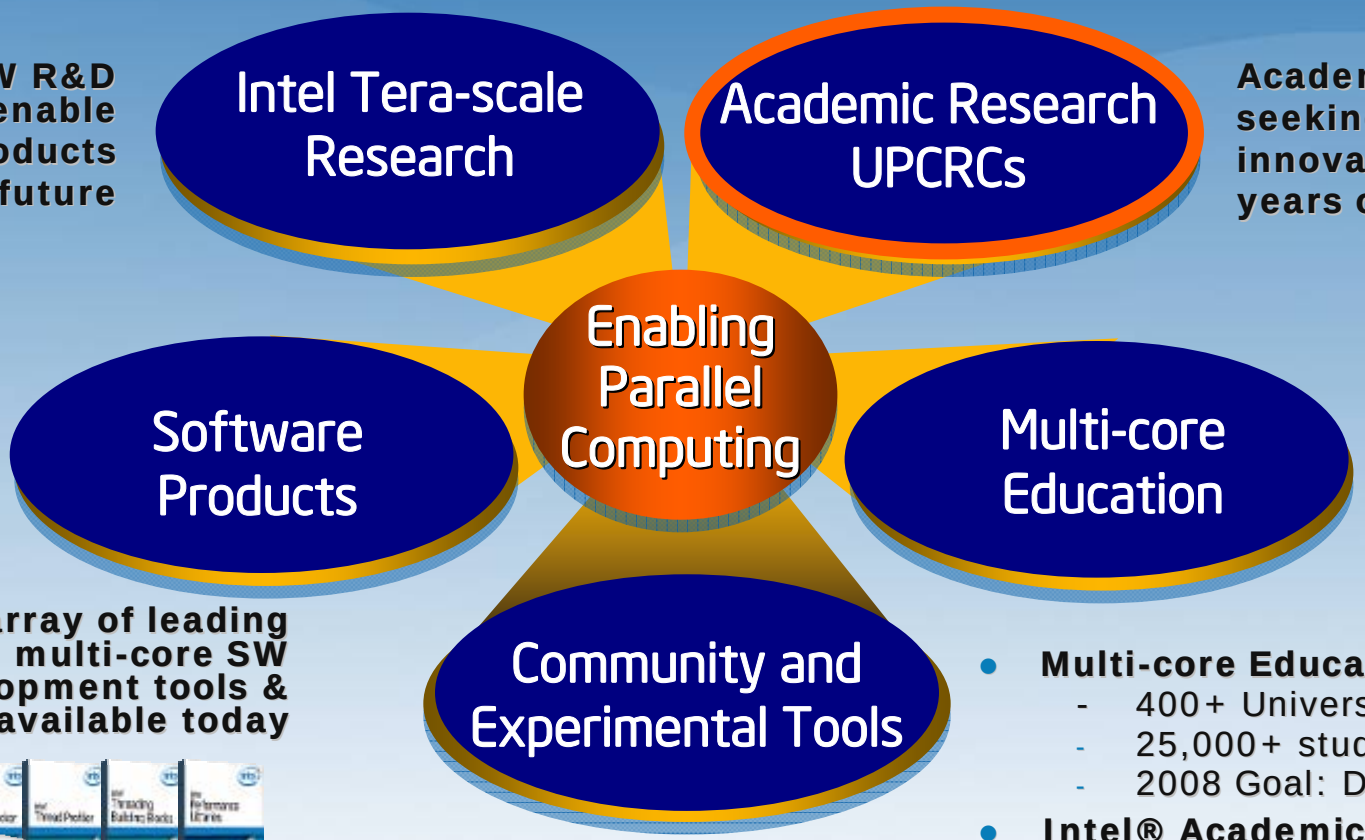
Parallel Applications
For desktop, gaming, and mobile systems

Parallel Architecture
Support new generation of programming models and languages

Parallel Sys. S/W
Performance scaling, memory utilization, and power consumption

Making Parallel Computing Pervasive

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



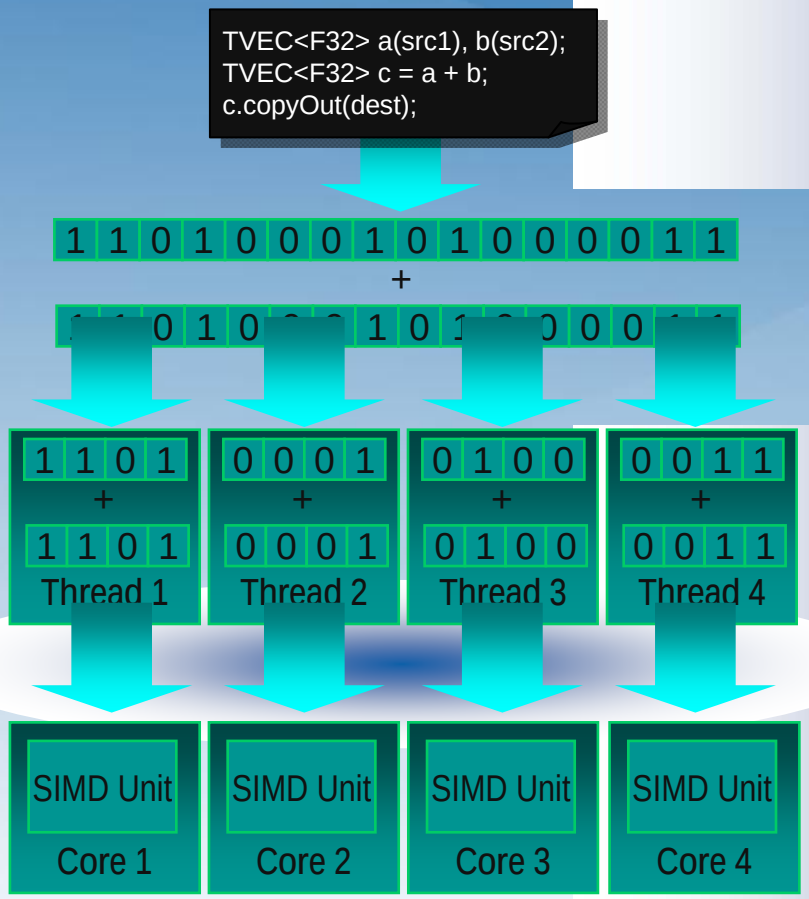
Academic research seeking disruptive innovations 7-10+ years out

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 Program**
  - 400+ Universities
  - 25,000+ students
  - 2008 Goal: Double this
- **Intel® Academic Community**
- **Threading for Multi-core SW community**
- **Multi-core books**

Ct: A Throughput Programming Language



User Writes Serial-Like
Core Independent C++ Code

Primary Data Abstraction is the Nested Vector
Supports Dense, Sparse, and Irregular Data

Ct Parallel Runtime:
Auto-Scale to Increasing Cores

Ct JIT Compiler:
Auto-vectorization, SSE,
AVX, Larrabee

Programmer Thinks Serially; Ct Exploits Parallelism
See whatif.intel.com

Intel® Concurrent Collections for C/C++

The application problem

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

Concurrent Collections Spec

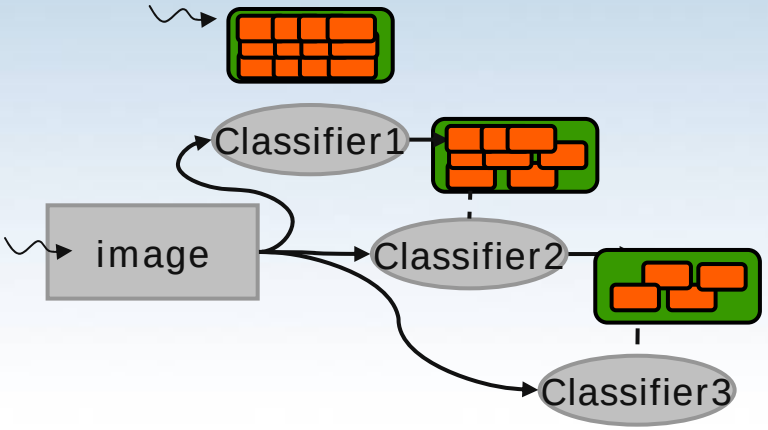
The work of the **tuning expert**
- Architecture
- Actual parallelism
- Locality
- Overhead
- Load balancing
- Distribution among processors
- Scheduling within a processor

Mapping to target platform

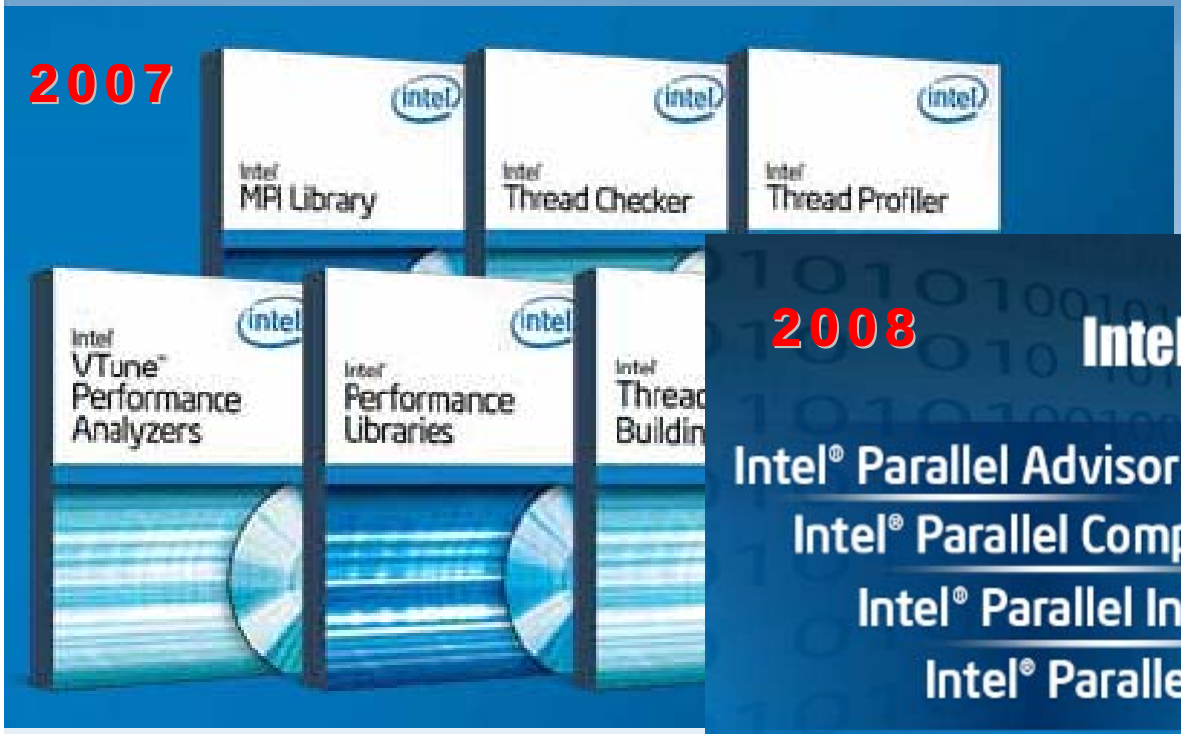
Supports serious **separation of concerns:**

The **domain expert** does not need to know about **parallelism**

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



Intel Software Products



2008

Announcing

Intel® Parallel Studio

Intel® Parallel Advisor

Intel® Parallel Composer

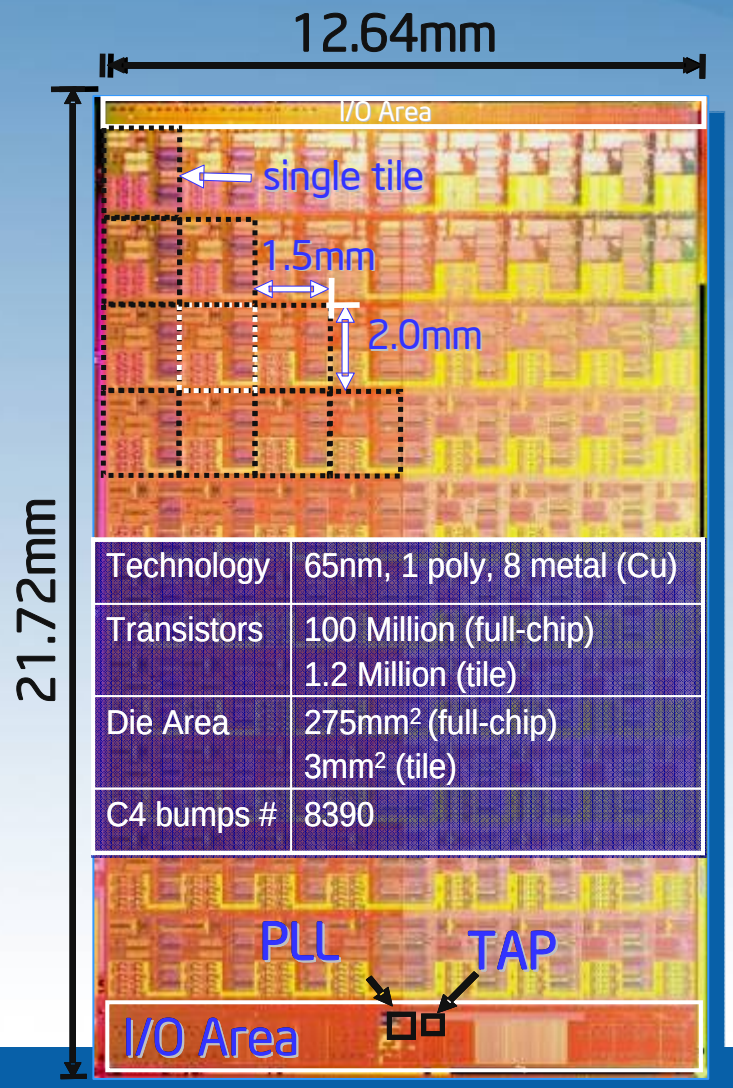
Intel® Parallel Inspector

Intel® Parallel Amplifier

www.intel.com/go/parallel

Performance *and* **Forward Scaling** *to* **Many-Core Processors**

“Polaris” Teraflops Research Processor



- 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

OpenCirrus Testbed

- Open large-scale, global testbed for academic research
  - Catalyze creation of an Open Source Cloud stack
  - Focus on data center management and services
  - Systems-level and application-level research
- Announced July 29, 2008, deployment expected December 2008
  - 6 Testbeds of 1,000 to 4,000 cores each

Create an academic cloud computing research community and open source stack - www.opencirrus.org

Summary

- Is Parallelism a crisis? No, there are major opportunities to lay new foundations and synergy across layers.
- Expectations and Challenges: Efficiency is not king
- Moving Parallel Programming Forward
  - Productivity and Ease, Composition, Locality, and only then Efficiency
- What's Going on at Intel: A broad array of explorations

Questions?