The Future of Multi-core: Tera-Scale Computing

James P. Held
Intel Fellow & Director,
Tera-Scale Computing Research

October 24, 2008

4th Software Engineering Conference (Russia) 2008

Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.
- Intel may make changes to specifications and product descriptions at any time, without notice.
- All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
- Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- 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.
- Intel, Intel Inside, and the Intel logo are trademarks of Intel Corporation in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
-

Agenda

- Tera-scale
  - Motivation
  - Platform Vision
- Research Challenges
  - Applications
  - Hardware
  - Programming
- Tera-scale and Future Product
- UPCRC
- Summary

What is Tera-scale?

TIPs of compute power operating on Tera-bytes of data

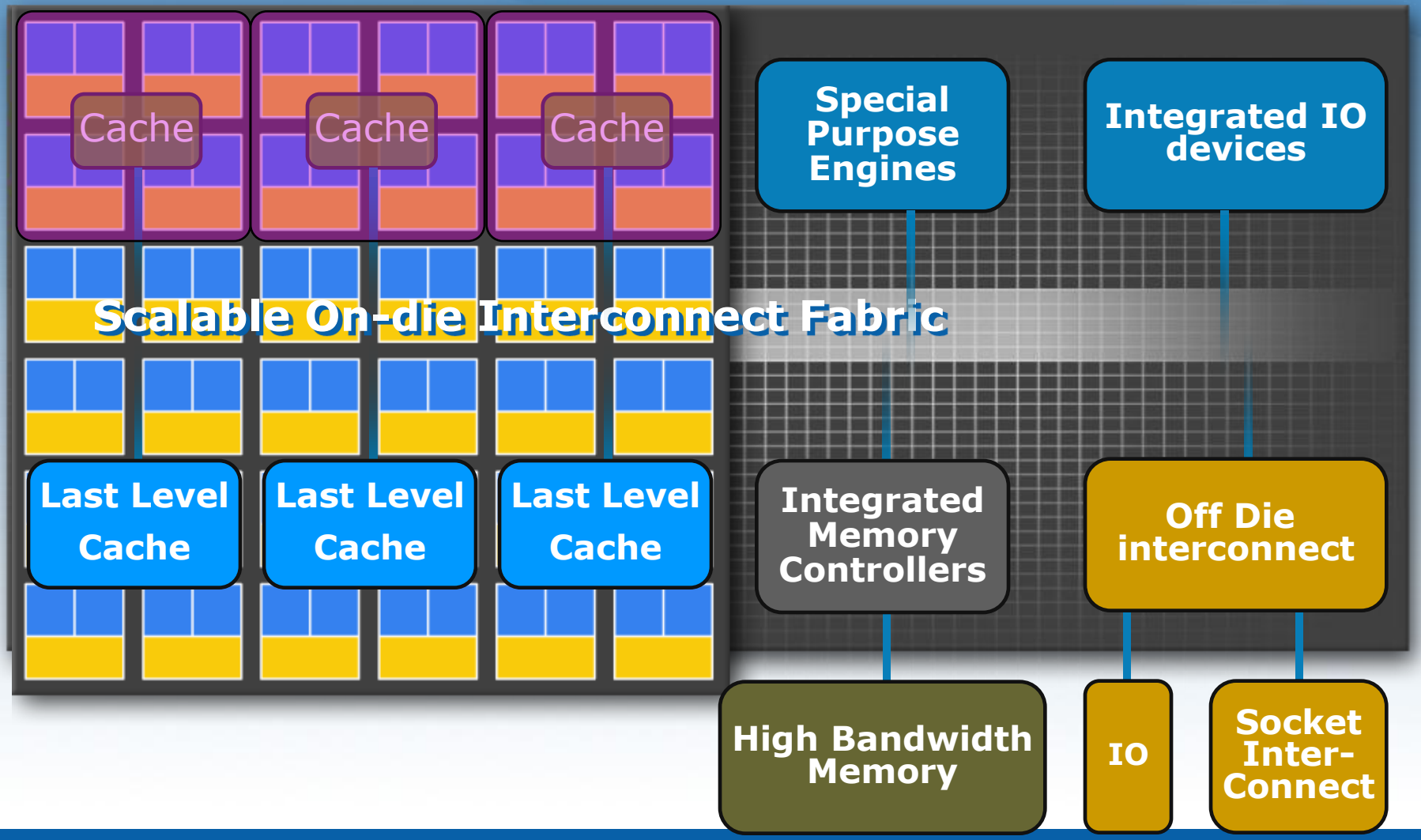
Entertainment

Learning & Travel

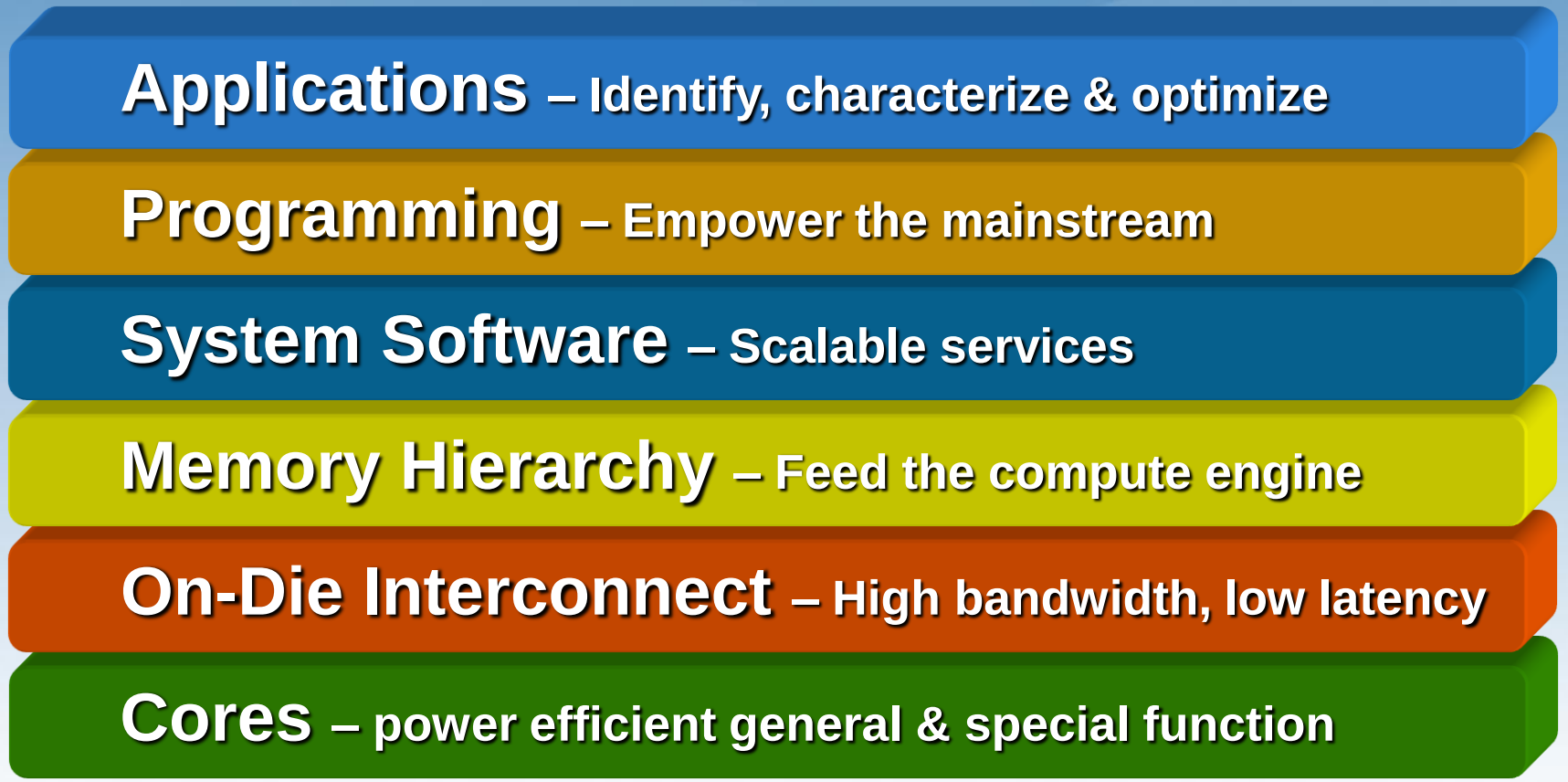
Personal Media Creation and Management

Health

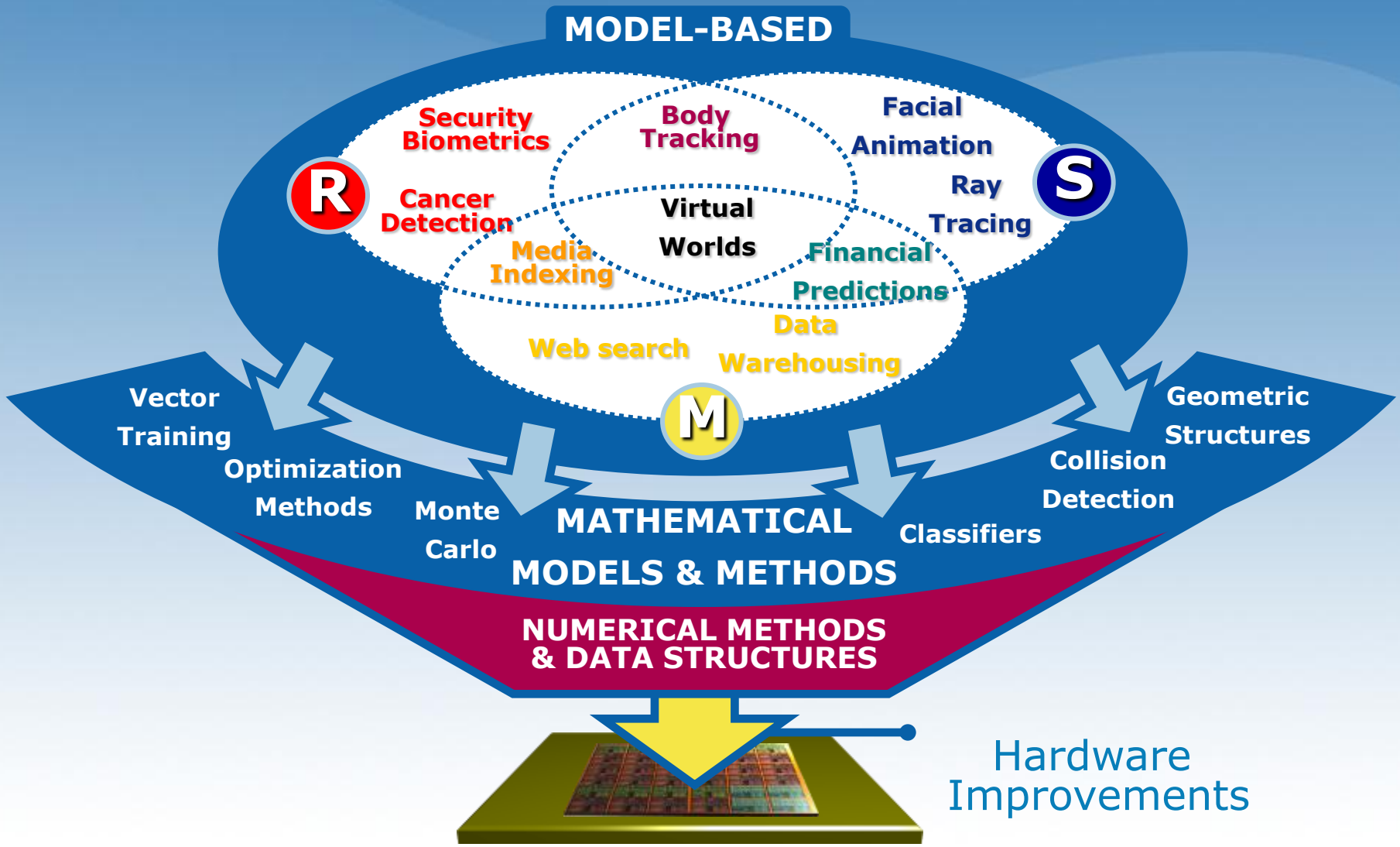
A Tera-scale Platform Vision



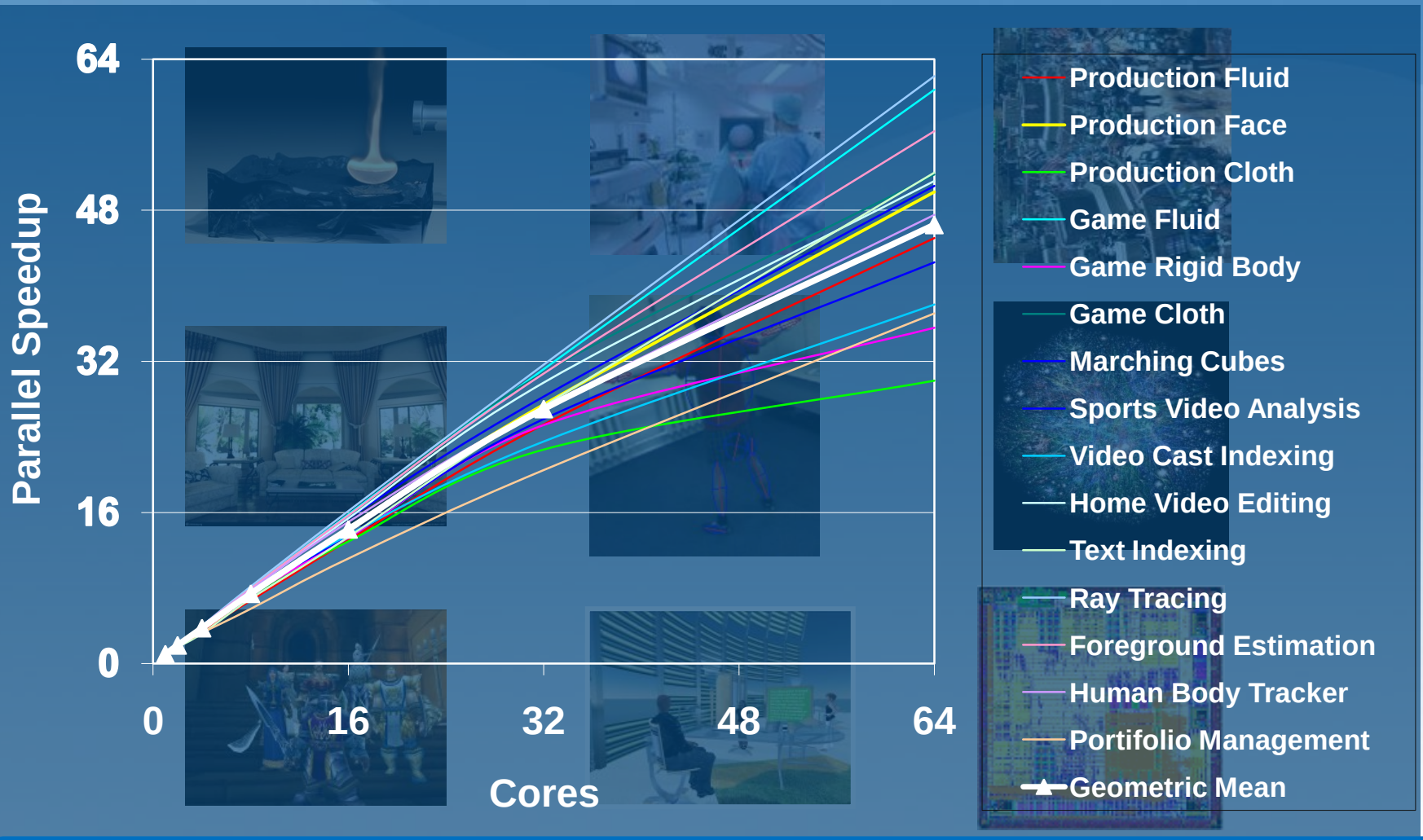
Tera-scale Research Challenges



A Top-down Approach to Analysis

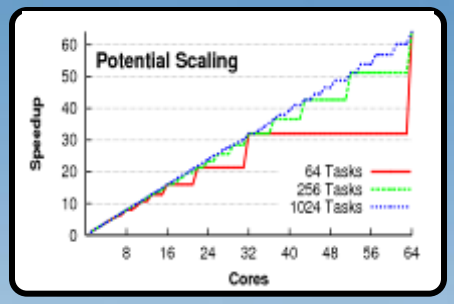


Application Kernel Scaling



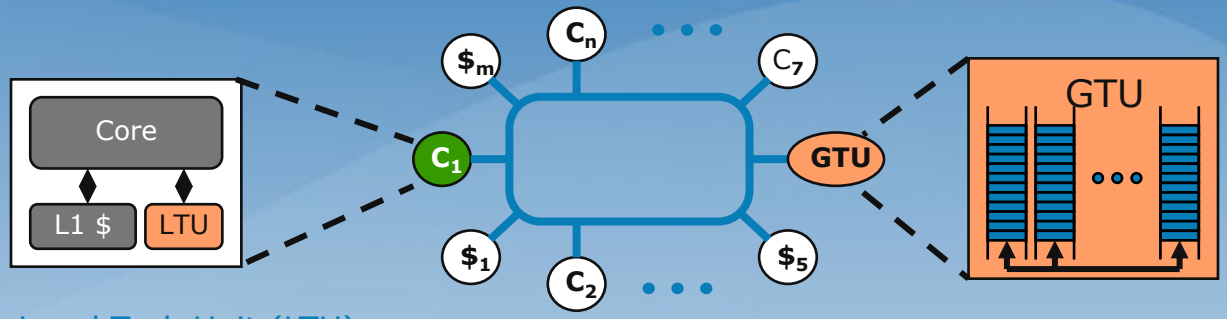
Graphics Rendering – Physical Simulation -- Vision – Data Mining -- Analytics

Application Acceleration: HW Task Queues



Task Queues

- scale effectively to many cores
- deal with asymmetry
- supports task & loop parallelism



Local Task Unit (LTU)

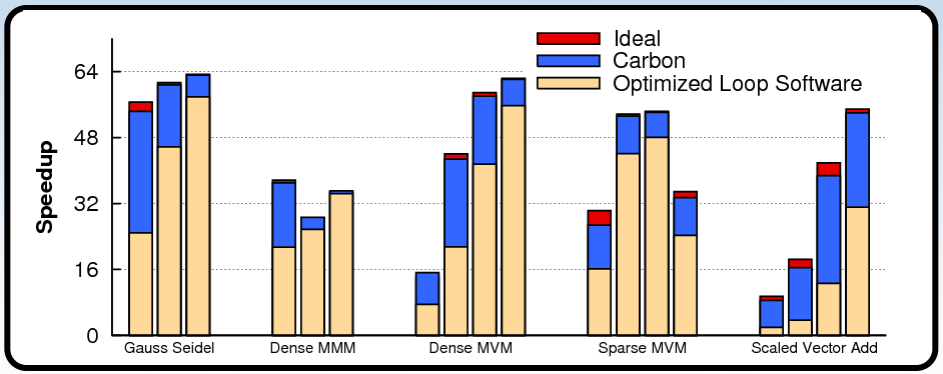
Prefetches and buffers tasks

Global Task Unit (GTU)

Caches the task pool

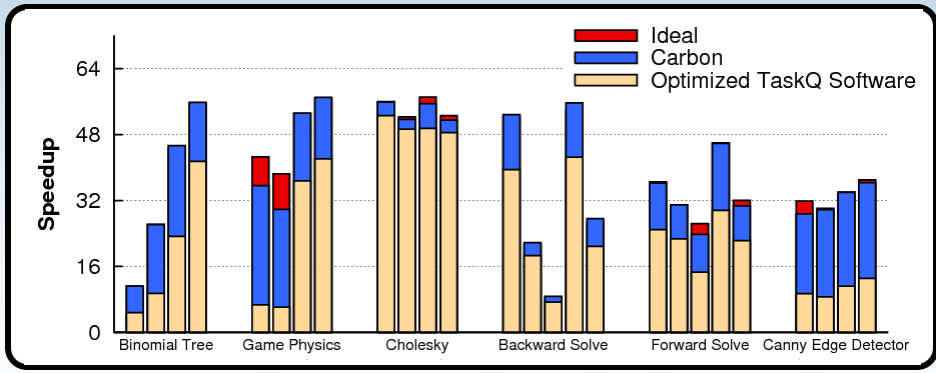
Uses distributed task stealing

Loop Level Parallelism



88% benefit optimized S/W

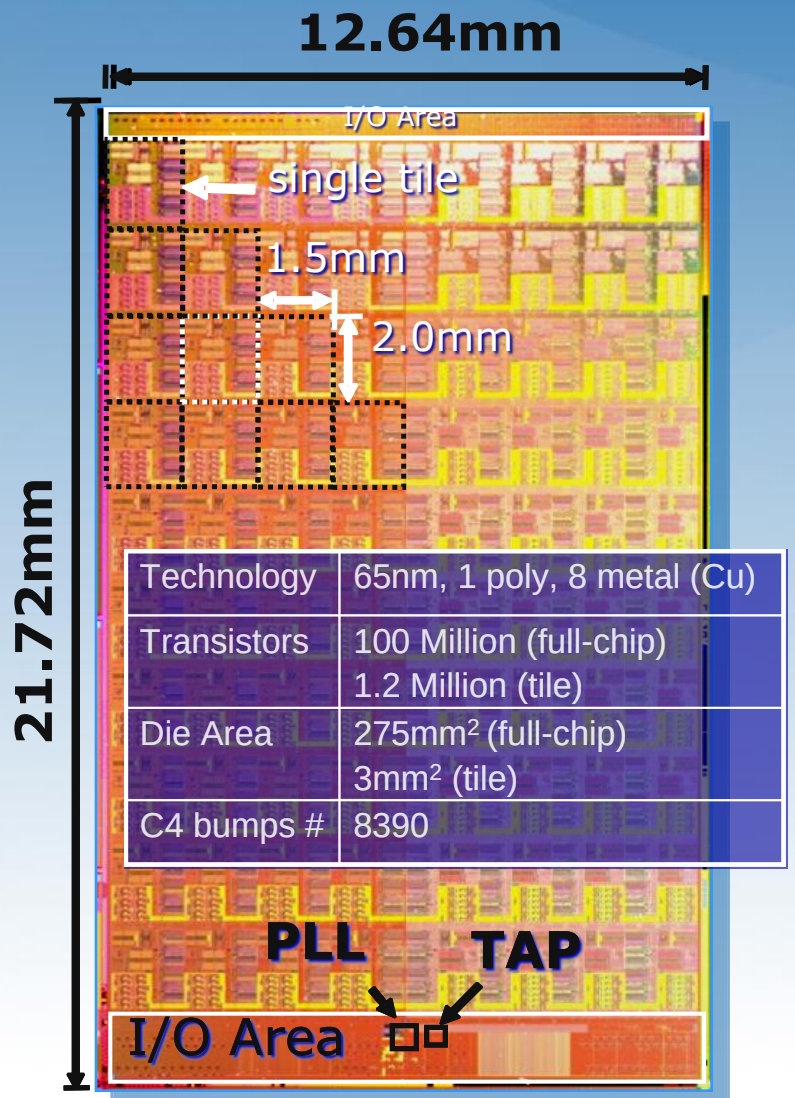
Task Level Parallelism



98% benefit over optimized S/W

Carbon: Architectural Support for Fine-Grained Parallelism on Chip Multiprocessors. Sanjeev Kumar Christopher J. Hughes Anthony Nguyen, *ISCA'07,* June 9–13, 2007, San Diego, California, USA.

Teraflops Research Processor

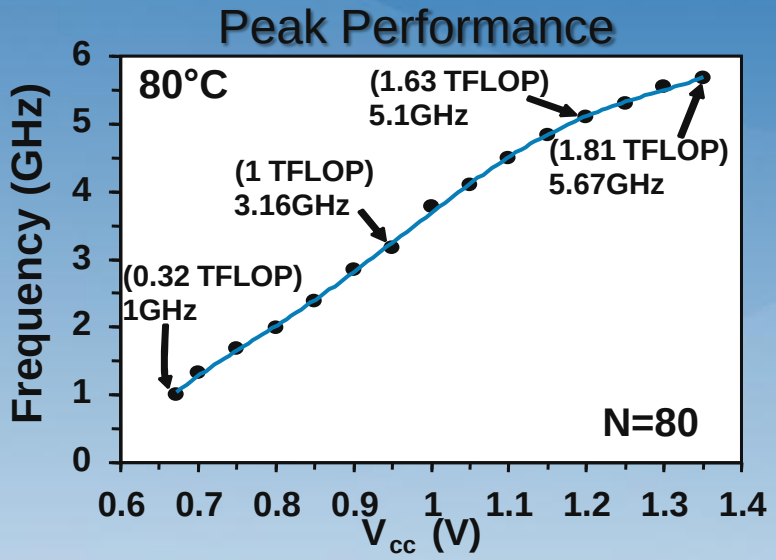


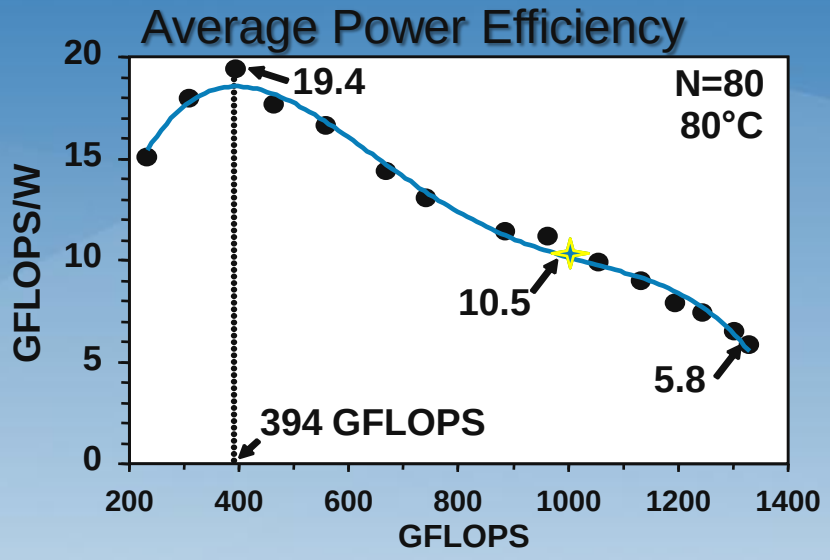
Technology	65nm, 1 poly, 8 metal (Cu)
Transistors	100 Million (full-chip) 1.2 Million (tile)
Die Area	275mm^2 (full-chip) 3mm^2 (tile)
C4 bumps #	8390

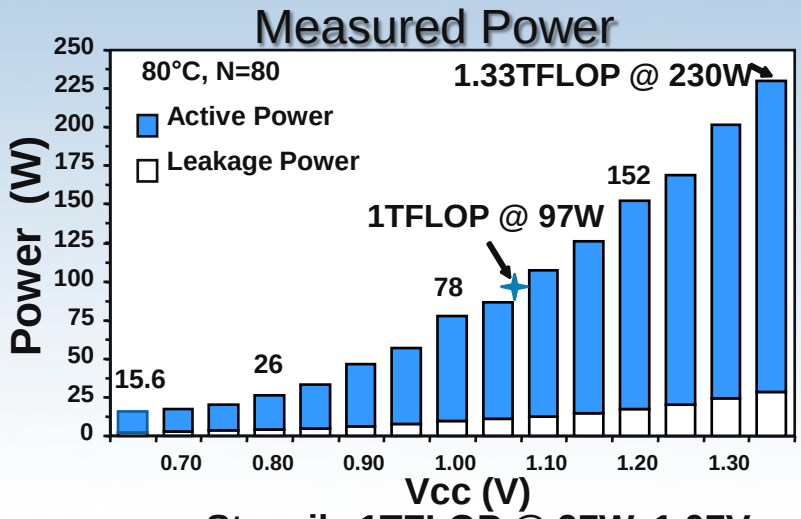
Goals:

- Deliver Tera-scale performance
  - Single precision TFLOP at desktop power
  - Frequency target 5GHz
  - Bi-section B/W order of Terabits/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

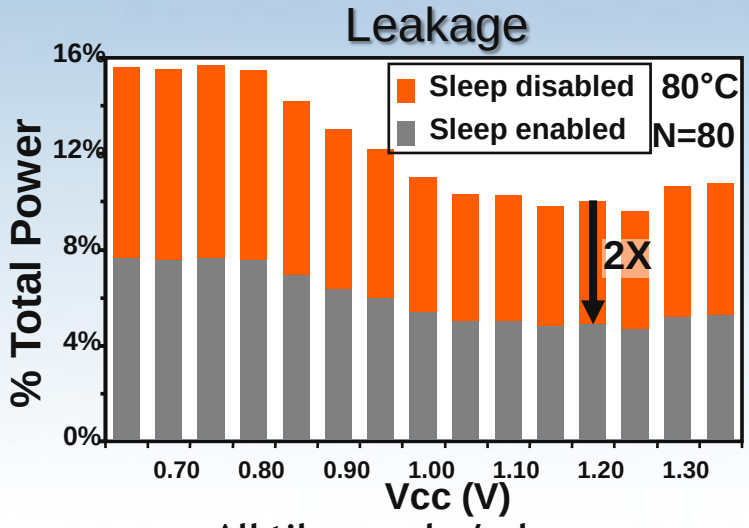
Power Performance Results







Stencil: 1TFLOP @ 97W, 1.07V;



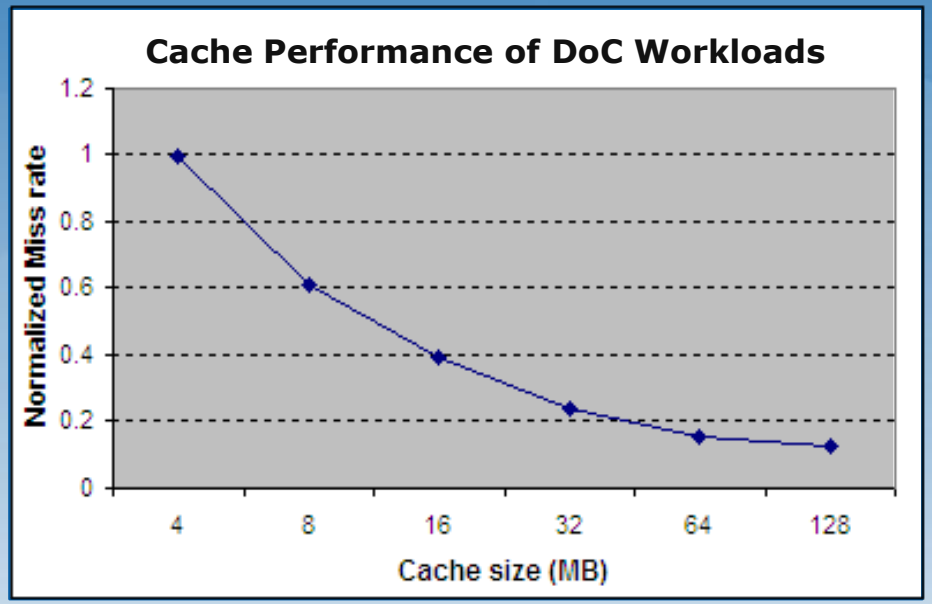
All tiles awake/asleep

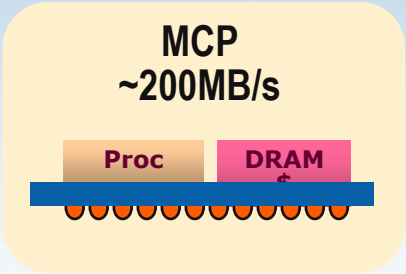
Vangal, S., et al., "An 80-Tile 1.28TFLOPS Network-on-Chip in 65 nm CMOS," in *Proceedings of ISSCC 2007(IEEE International Solid-State Circuits Conference)*, Feb. 12, 2007.

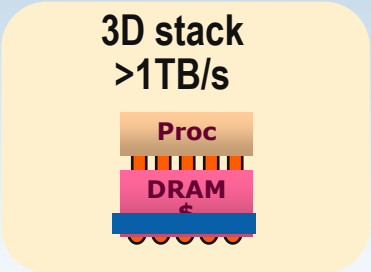
On-Socket DRAM Caches

For Memory Scalability

- Enable Large Capacity L4s
  - Low Latency
  - High Bandwidth
- Technologies
  - Multi-chip Packages (MCP)
  - 3D Stacking
- Benefits
  - Significant Miss Rate Reduction
  - Avoids bandwidth wall
  - At better latency







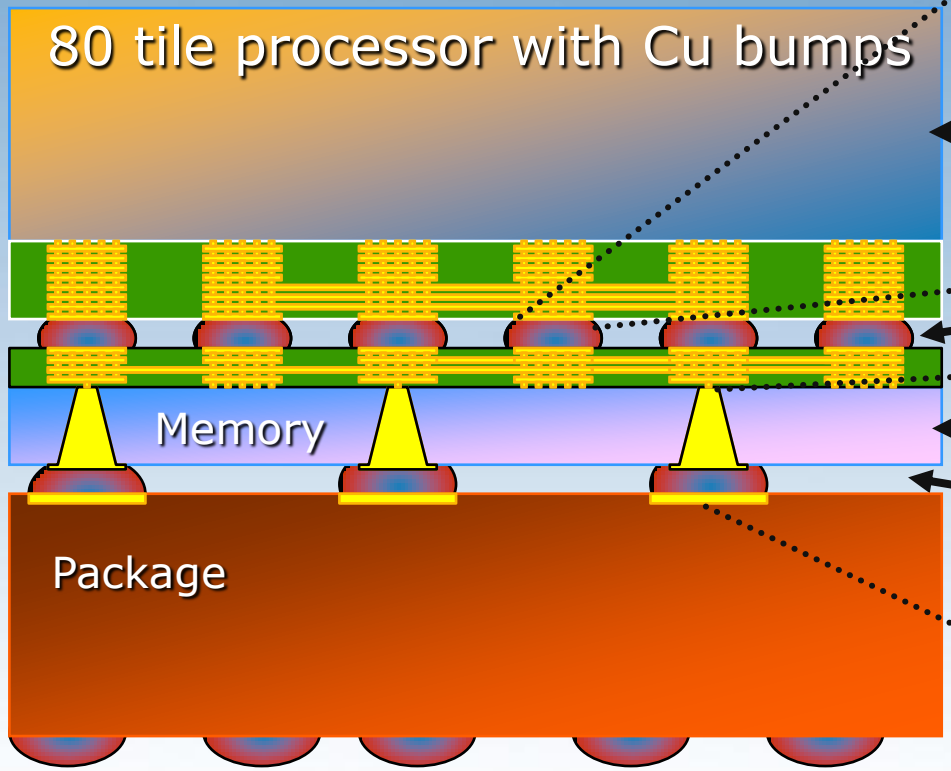
Iyer, R, et al, "Datacenter-on-Chip Architectures: Tera-scale Opportunities and Challenges", and Polka, LA et al, "Package Technology to Address the Memory Bandwidth Challenge for Tera-scale", Computing", *Intel Technology Journal*, Volume 11, Issue 3, 2007

Work in Progress: Stacked Memory Prototype

256 KB SRAM per core

4X C4 bump density

3200 thru-silicon vias

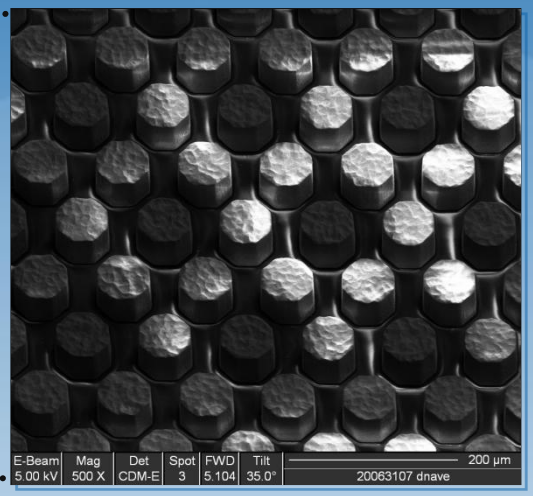


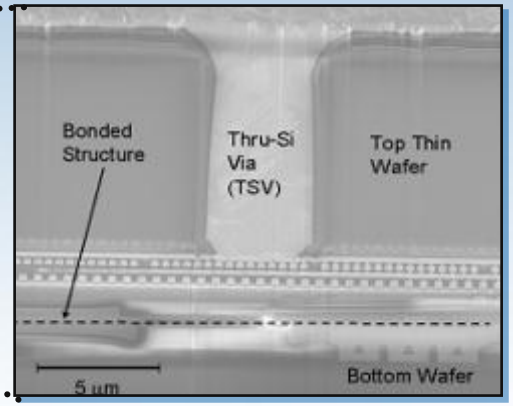
"Polaris"

Denser than C4 pitch

"Freya"

C4 pitch

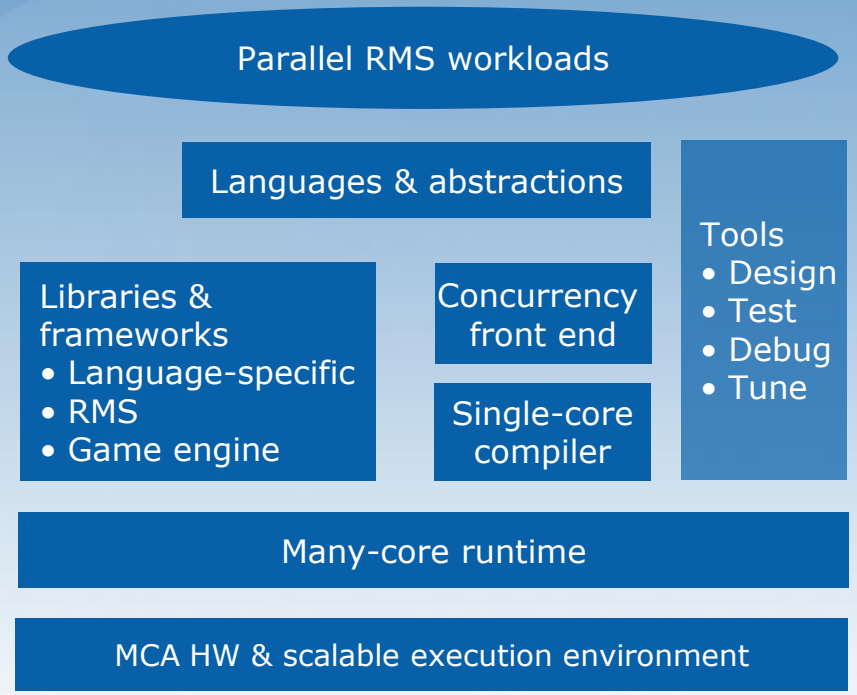




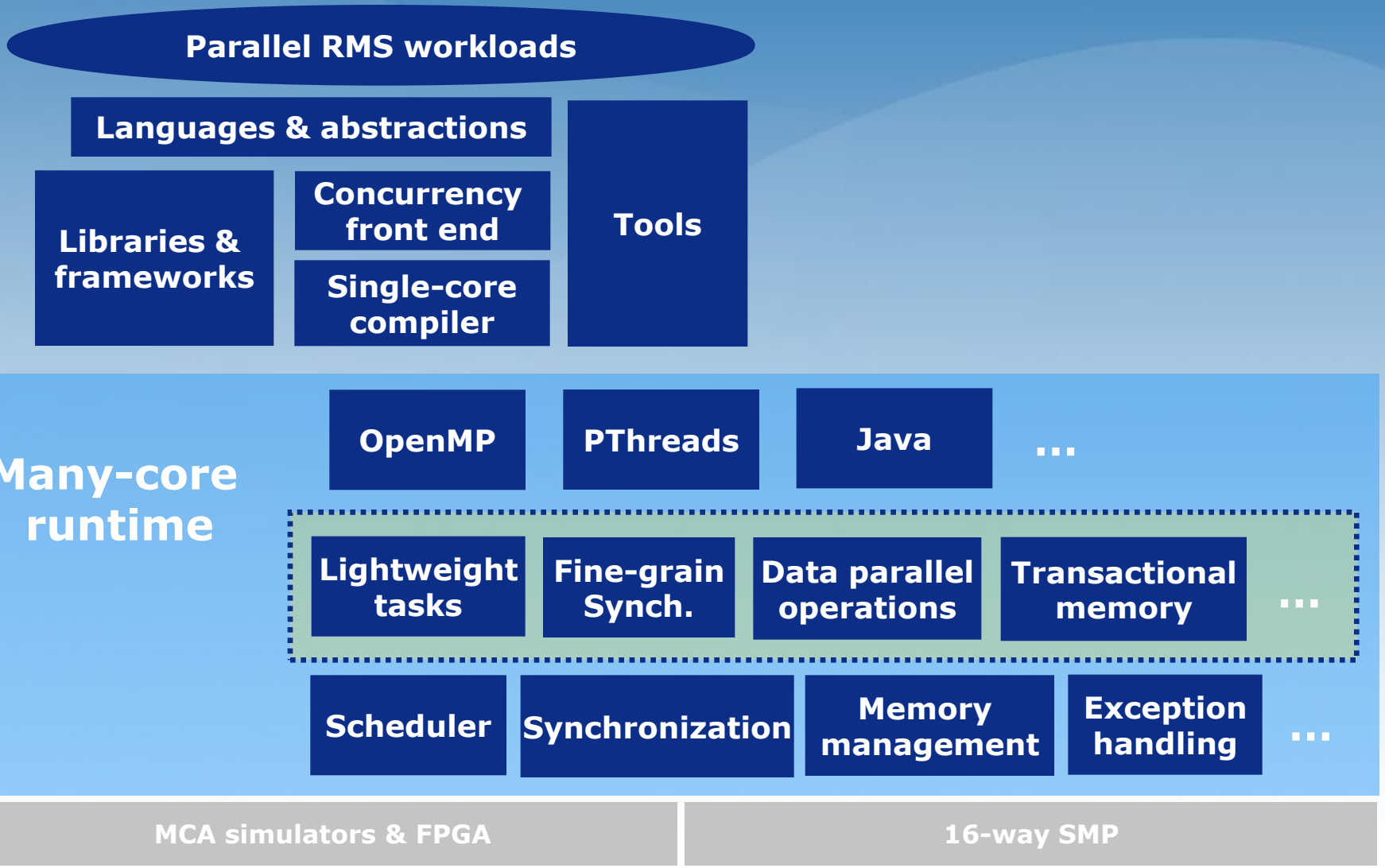
Memory access to match the compute power

Programming Environment Research

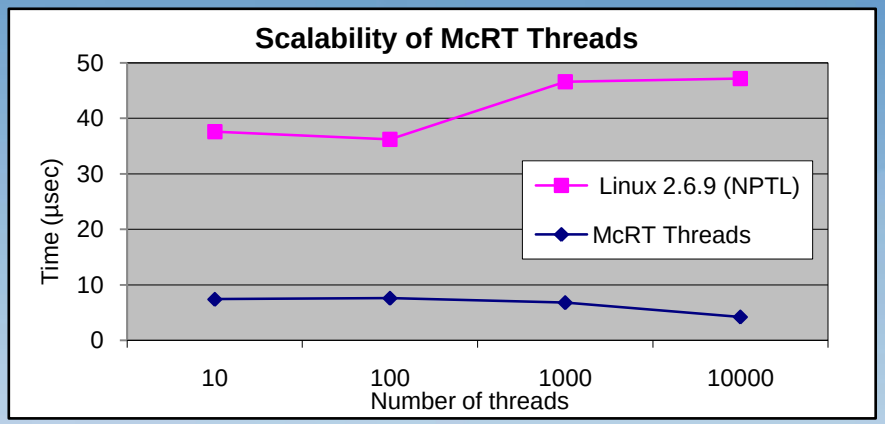
- **Languages & programming abstractions**
  - Transactional Memory
  - Data parallel operations
  - Lightweight tasks
  - Fine-grain synchronization
  - Message passing
- **Compilers**
  - Multi-language support
  - Optimizations for Prog. Abst.
  - Dynamic & static compilation
  - Speculative multithreading
- **Many-core runtime scalability**
  - Efficient scalable support for Programming Abstractions

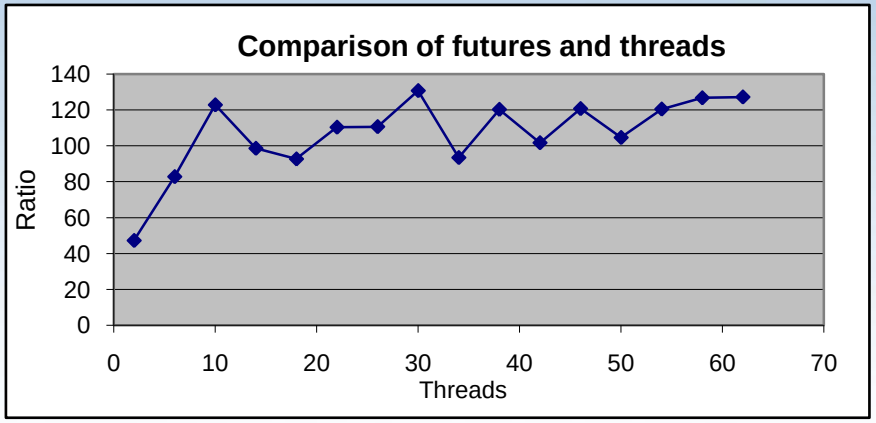


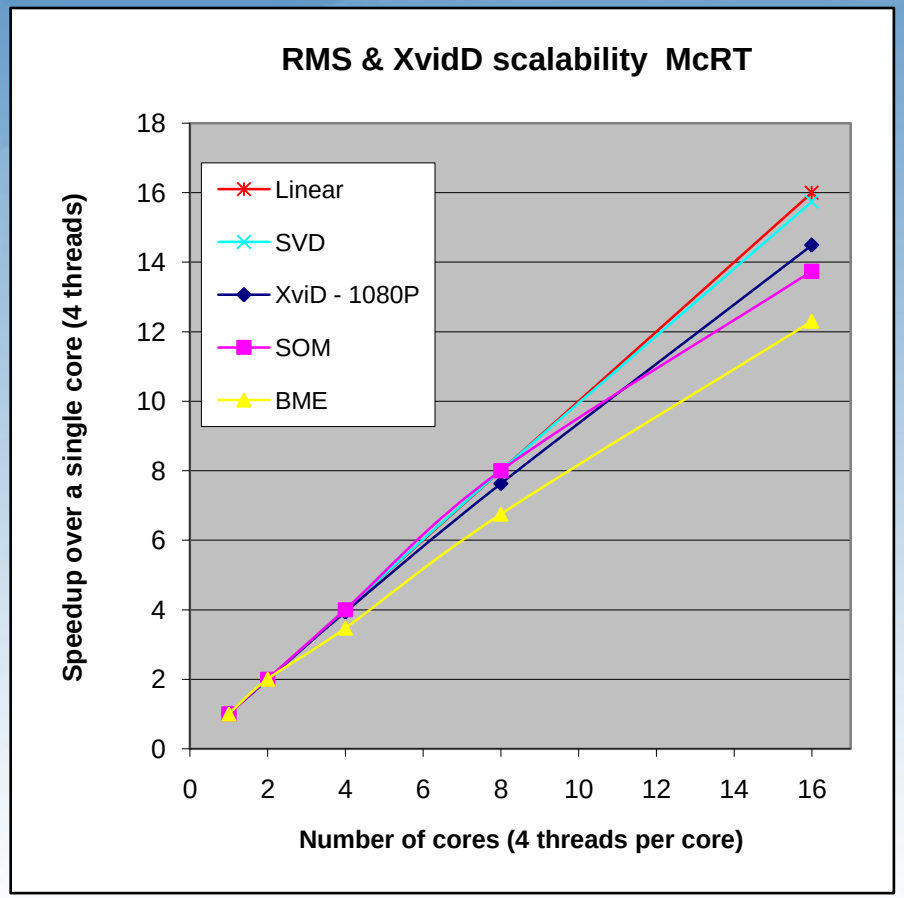
Many Core Runtime



Many Core Runtime Performance & Scalability







B Saha, A Adl-Tabatabai, et al. "Enabling Scalability and Performance in a Large Scale CMP Environment." in *Eurosys'07*, March 21-23, 2007, Lisbon, Portugal.

Transactional Memory

- Definitions
  - Memory transaction: A sequence of memory operations that either execute completely (commit) or have no effect (abort)
  - Atomic: An "all or nothing" sequence of operations
    - On commit, all memory operations appear to take effect as a unit
    - On abort, none of the stores appear to take effect
  - Transactions run in isolation
    - Effects of stores are not visible until transaction commits
    - No concurrent conflicting accesses by other transactions

- Goal – an atomic block language construct
  - As easy to use as coarse-gain locks, but with the scalability of fine-grain locks
  - Safe and scalable composition of SW modules
  - Transactional execution
    - Isolated: No interference from other threads
    - Atomic: All-or-nothing sequence of operations

Intel C/C++ TM compiler

- Based on production Intel C/C++ compiler
- Downloadable from http://whatif.intel.com
  - Use for experimentation & workload development
  - Provide feedback on language extensions
- State-of-the-art STM runtime
  - Multiple STM algorithms + dynamic mode switching
  - Support for calling legacy code & I/O inside transactions
  - Optimized transactional malloc & free

Execute block atomically `__tm_atomic {S}`

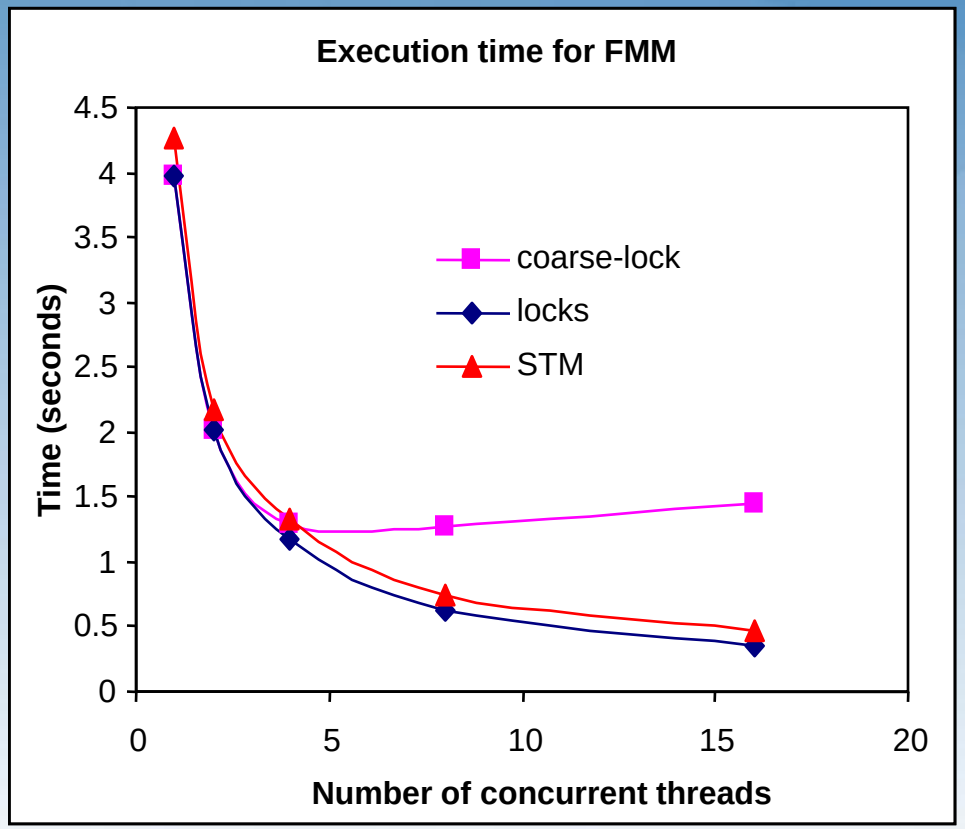
Abort transaction `__tm_abort`

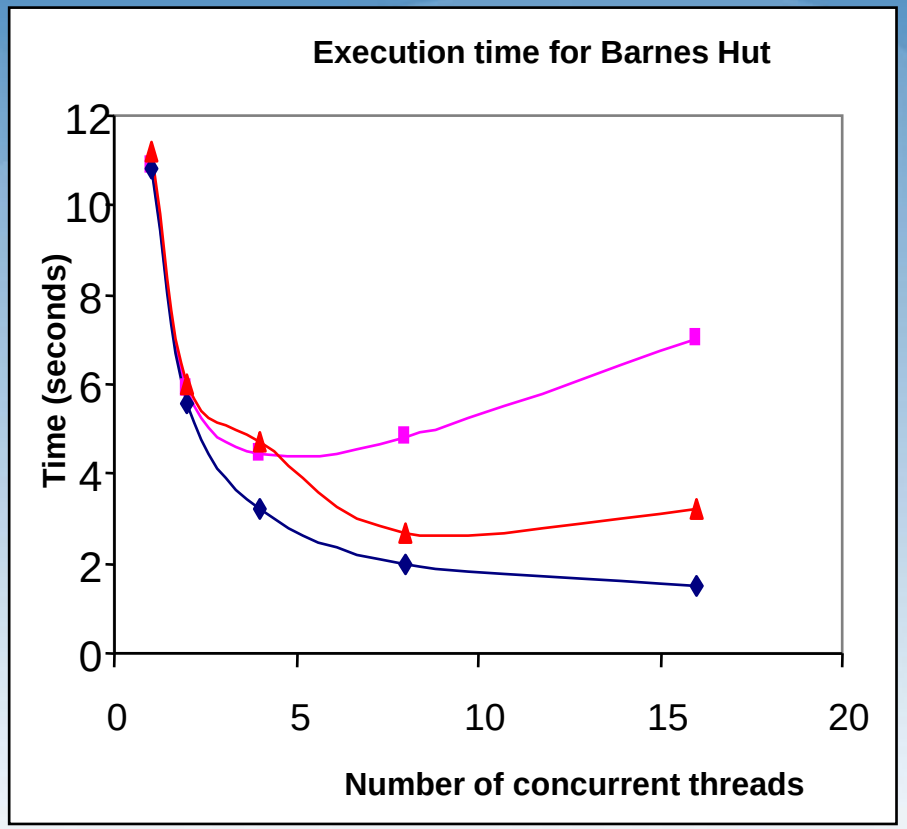
Declare transactional functions, classes & templates

```
__attribute__((tm_callable)) void foo(…);
__attribute__((tm_callable))
class Bar {virtual void foo(…);}
```

http://softwarecommunity.intel.com/articles/eng/1460.htm

Results: SPLASH-2





STM scales well but has single-thread overhead

TM challenges

Lack of TM application development experience
- Are annotations intrusive?
- Is failure atomicity important?
- Which libraries should be supported?
- Do atomic blocks really help the programmer?

Debug, performance & development tools

Extending atomic block semantics
- Nested parallelism
- Co-ordination
- Open nesting

Language-level memory model for TM
- Strong atomicity – full isolation
- Weak atomicity – isolation between transactions only

Ct: Nested Data Parallel Programming

- Race-free programming with on-the-fly, automatic generation of threads tailored to user's *multi-core* hardware
- Simpler, high performance, scalable, SSE-friendly code
- Library-like interface compatible with existing programming environments and APIs

```
CCtTVEC<double> sparseMatrixVectorProduct(
  CCtVEC<double> A, CCtVEC<int> rowindex,
  CCtVEC<int> cols, CCtVEC<double> v)
{
  CCtVEC expv = ctDistribute(v,cols);
  CCtVEC product = A*expv;
  return ctMultiReduceSum(product,rowindex);
}
```

Extends C/C++ by Adding New Parallel Data Structures & Operators

Ghuloum, A, et al, "Future Proof Data Parallel Algorithms and Software on Intel Multi-core Architecture"
Intel Technology Journal, Volume 11, Issue 4, 2007

Data parallel vector sum

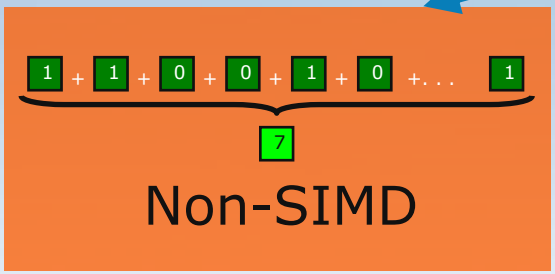
Data parallel program

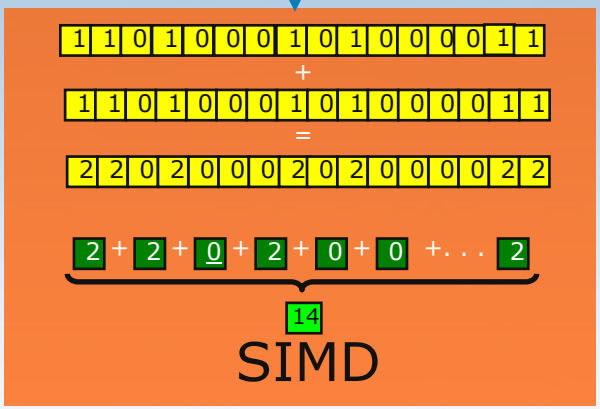
```
VEC B;
  . . .
x = Sum_Of_Elements(B);
```

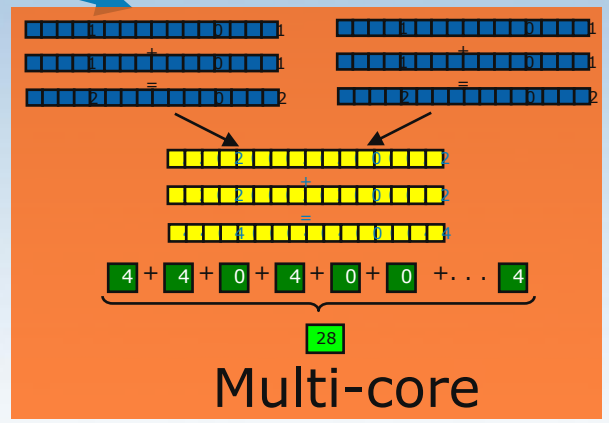
Data parallel compiler & runtime

Compiler understands and optimizes data parallel operations

Automatic mapping based on vector size and architecture

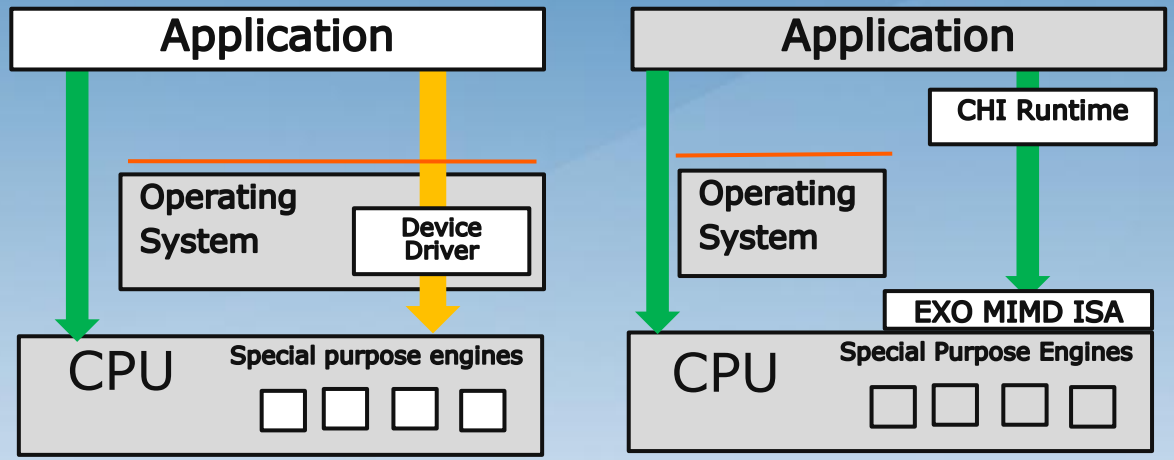






Data Parallelism Manages Parallelism Without Exposing Threads

Accelerator Exoskeleton – Heterogeneous Multi-core

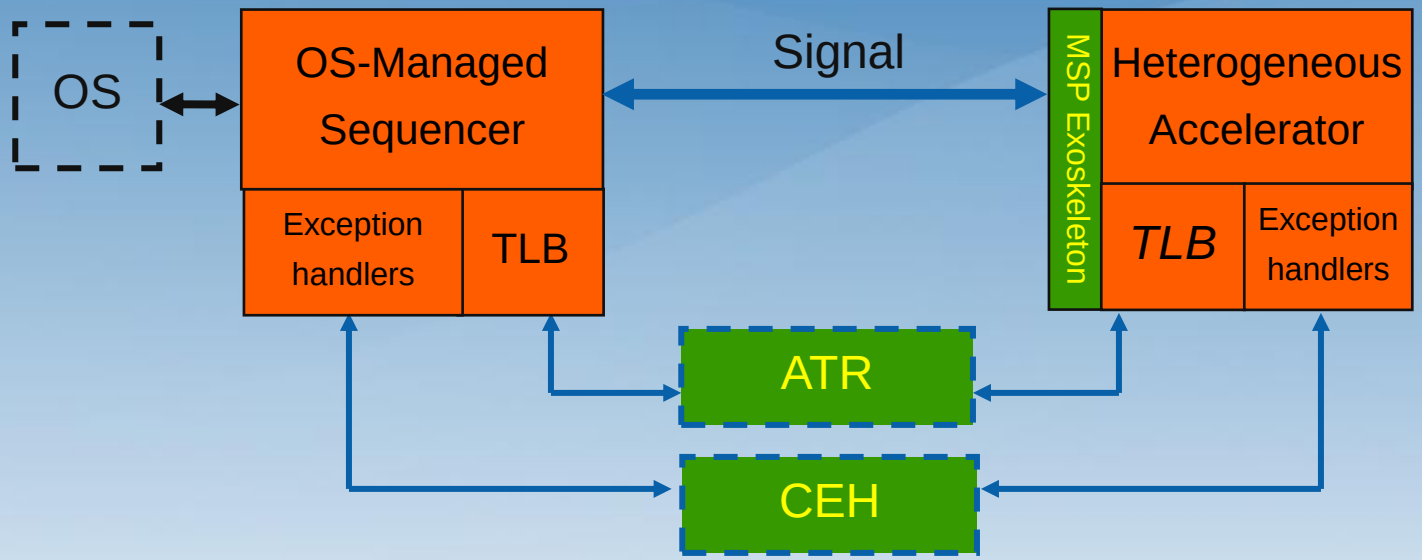


- Provides user-level access to heterogeneous processing
  - ISA alternative to OS device driver model
  - Runtime support for integrated programming environment

Wang, P et al. "Accelerator Exoskeleton" in *Intel Technology Journal*, Volume 11, Issue 4, 2007

R. Hankins, et al, "Multiple Instruction Stream Processor", in *Proceedings of the 33rd ISCA*, June 2006.

Accelerator Exoskeleton – Heterogeneous Multi-core



- Provides user-level access to heterogeneous processing
  - ISA alternative to OS device driver model
  - Runtime support for integrated programming environment
- Multi-Instruction Stream Processor (MISP) "Exoskeleton"
  - Signaling instruction and proxy execution
  - Address Translation Remapping
  - Collaborative Exception Handling

Wang, P et al. "Accelerator Exoskeleton" in *Intel Technology Journal*, Volume 11, Issue 4, 2007

R. Hankins, et al, "Multiple Instruction Stream Processor", in *Proceedings of the 33rd ISCA*, June 2006.

Exo-Programming Environment: C for Heterogeneous Integration

- Programming model
  - Multiple “shreds” within a single OS thread context
  - Shared virtual address space
  - Heterogeneous sequencers directly exposed to application
- Compiler & user-level runtime
  - Modified front-end and OpenMP pragmas
    - Fork/join
    - Producer/consumer parallelism
  - “Fat” binary
  - Extensible to multiple types of heterogeneous cores

Inline Data-stream Programming Language

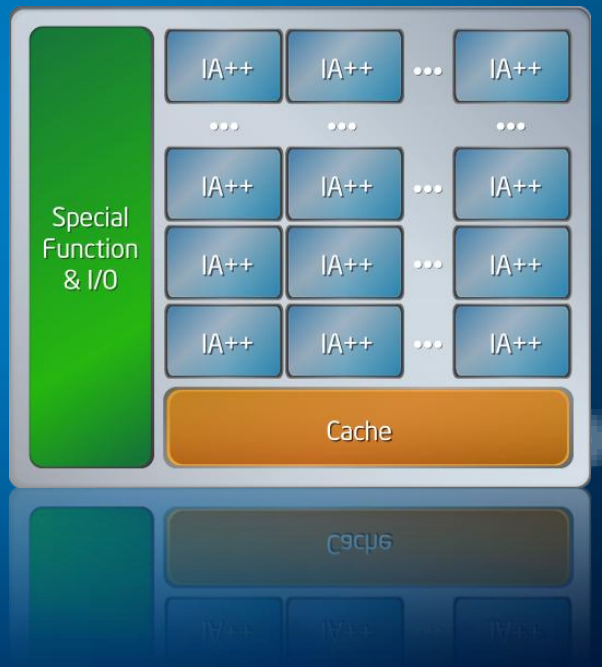
```
float Vin[4];
float Vout[4];

void *in_desc  = (void *)chi_alloc_buffer_desc
        (DPE_INPUT_BUFFER,  Vin, 4, 1);
void *out_desc = (void *)chi_alloc_buffer_desc
        (DPE_OUTPUT_BUFFER, Vout, 4, 1);

#pragma omp parallel target(dpe)
       shared(Vin,Vout)
descriptor(in_desc,out_desc)
{
  __dpl {
    configuration[1] cfgMult( vector val[1],
                    vector coeff[1] )
    {
      result bs( mull(val, coeff), 13 );
    }
    flow[4] multiFlow( vector vec[4],
           vector coeffs[4])
    {
      vector ret[4]; result out;
      selector[iter : 4] sel[1] = {{ iter }};
      selector[iter : 4] selRev[1] = {{ 3 - iter
}};
      ret[sel] = cfgMult(vec[sel], coeffs[selRev]);
    }
    vector cf[4] = { 0.5 + I * 0.0 };
    program dlMain()
    {
      Vout = multiFlow(Vin, cf);
    }
  }
}
```

J. Hoffman, D. A. Ilitzky, A. Chun, A. Chapyzhenka, “Architecture of Scalable Communication Core,” in *First International Symposium on Networks-on-Chip, 2007.*

Teraflops on IA product

Larrabee Architecture for Visual Computing

- Many IA cores
  - Scalable to TeraFLOPS
- New cache architecture
- Throughput architecture
- New vector instruction set
  - Vector memory operations
  - Conditionals
  - Integer and FP arithmetic
- New vector processing unit / wide SIMD



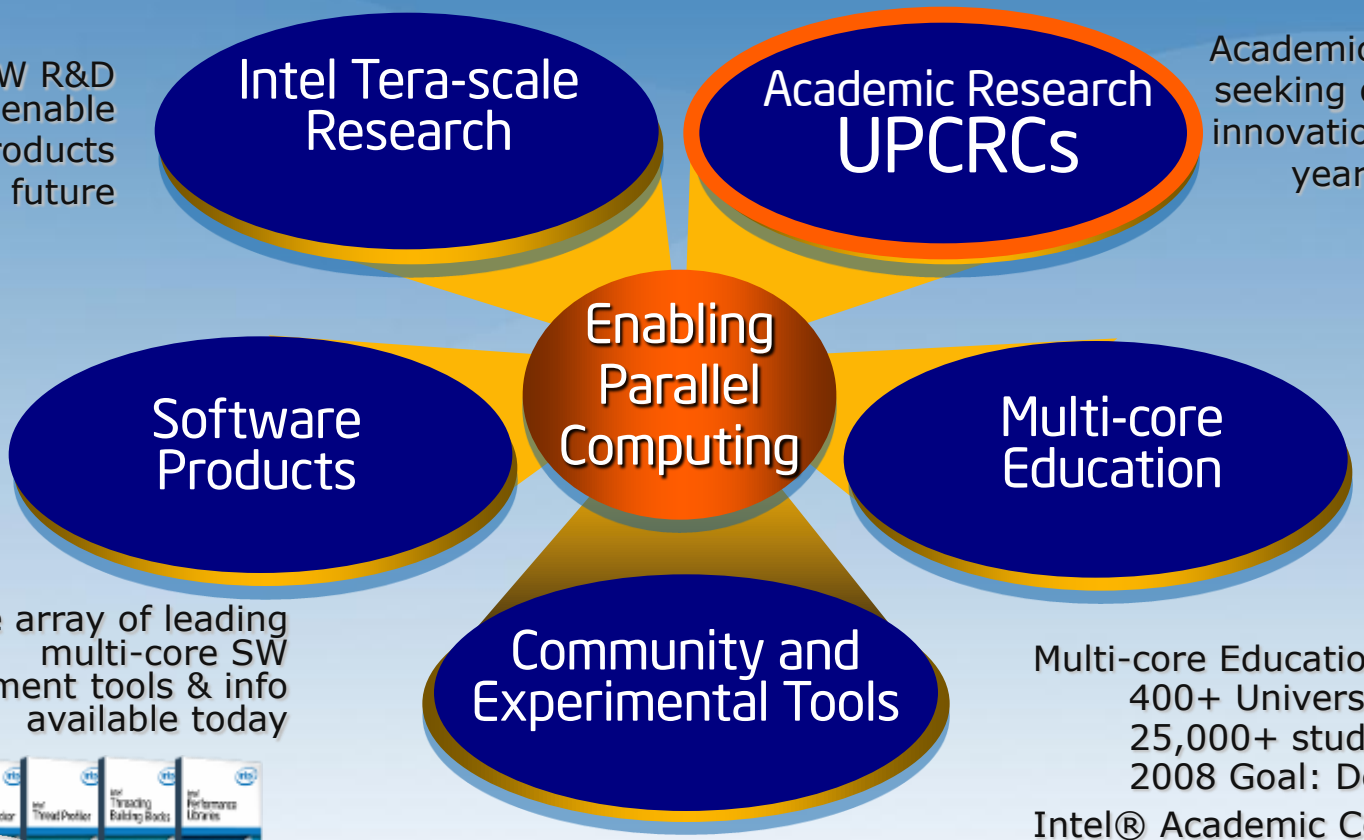
Pat Gelsinger Keynote - Intel IDF Spring 2008

Invent the new reality. Intel Developer FORUM

Seiler, L. et al. "Larrabee: A many-core x86 architecture for visual computing." *SIGGRAPH '08: ACM SIGGRAPH 2008 Papers*, ACM Pres, New York.

Making Parallel Computing Pervasive

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

UPCRC Partners in Research

- Intel & Microsoft provide funding and guidance
- Universities direct groundbreaking research

Professor David Patterson
UCB UPCRC Director

Prof. Wen-Mei Hwu

Prof. Marc Snir

UIUC UPCRC
Co-Directors

Summary

- Intel research is addressing the challenges of parallel computing with Intel platforms
  - Teraflop hardware performance within mainstream power and cost constraints
  - ISA enhancements to address emerging workload requirements
  - Language and runtimes to better support parallel programming models
  - Partnering with Microsoft to support academic research
- Intel is developing hardware and software technologies to enable Tera-scale computing


intel®
Leap ahead™