# Tera-scale Computing Research

## Addressing the challenges of mainstream parallel computing

Jim Held
Intel Fellow, Director
Tera-scale Computing Research

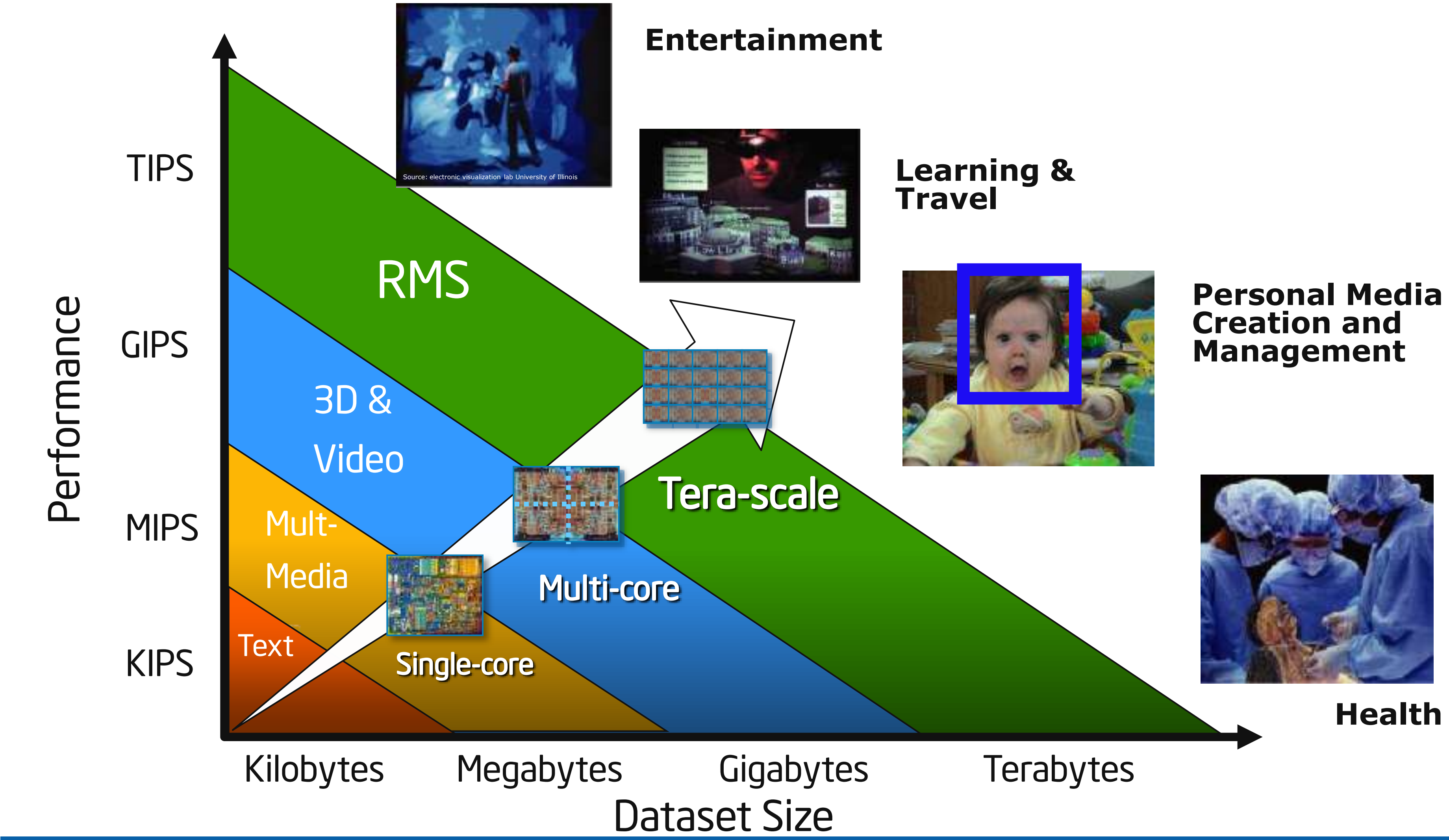
UCB PARLab February 25, 2010

# Agenda

- Tera-scale Computing
- Platform Vision
- Research Agenda
  - Applications
  - Programming
  - System Software
  - Memory Hierarchy
  - Interconnects
  - Cores
- Summary

# What is Tera-scale?

TIPs of compute power operating on Tera-bytes of data



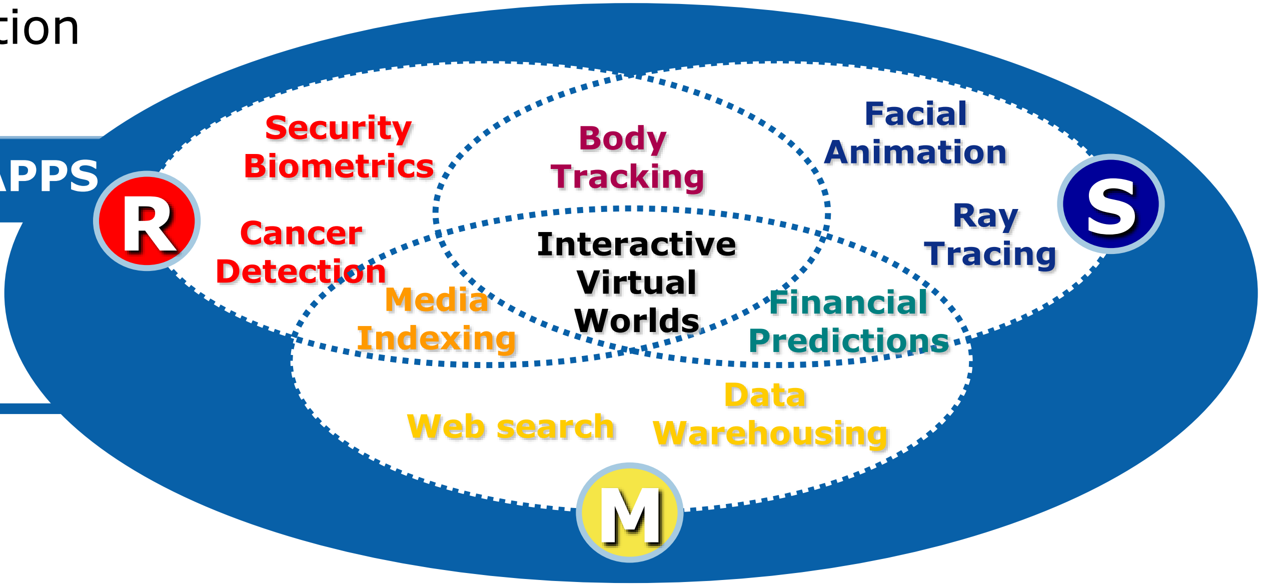
Source: electronic visualization lab University of Illinois

# Tera-scale Computing Applications

- Based on modeling or simulating the real world
- Make technology more immersive and human-like
- Algorithms are highly parallel in nature
- **Real-time** results essential for user-interaction

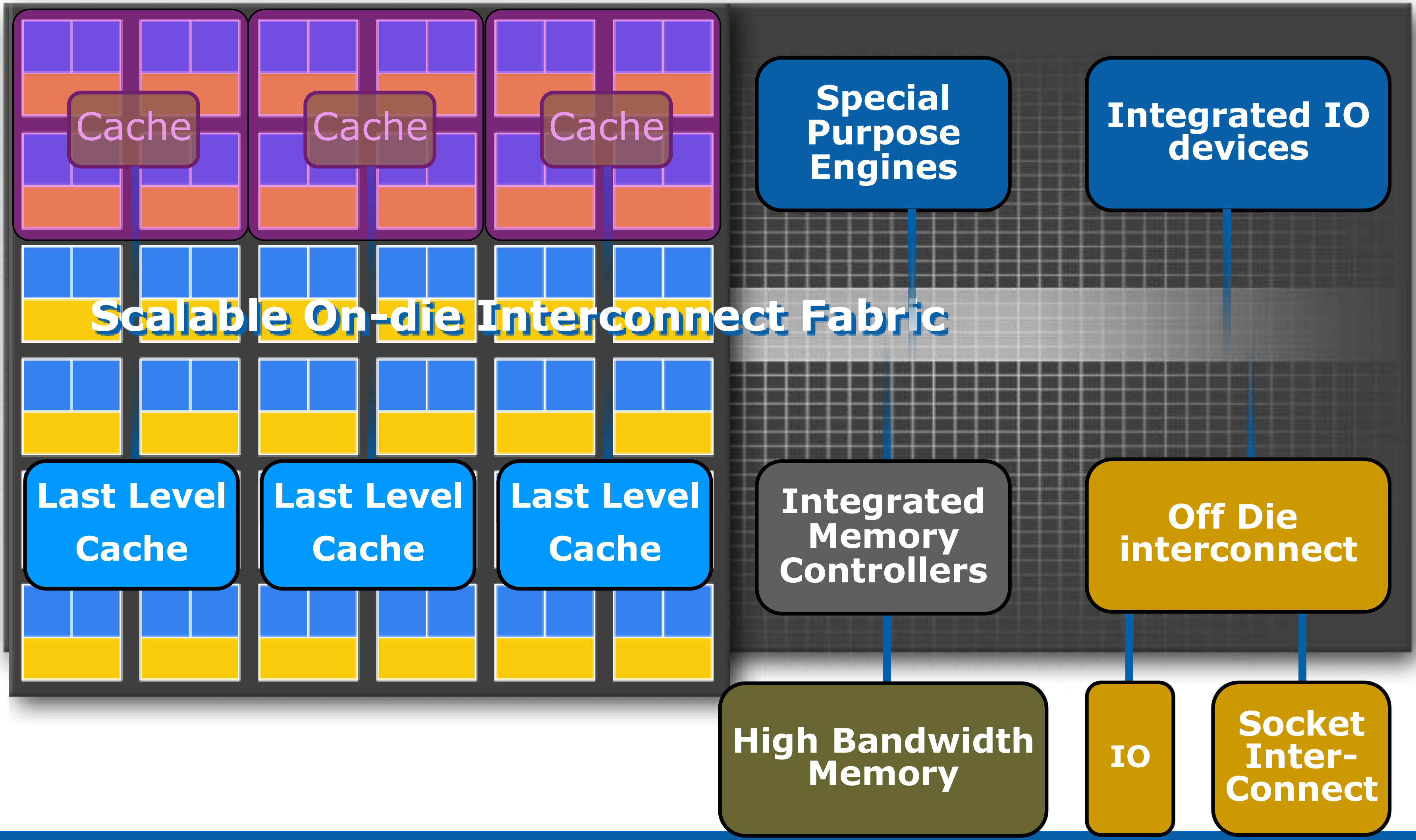
MODEL-BASED APPS

R = Recognition
M = Mining
S = Synthesis

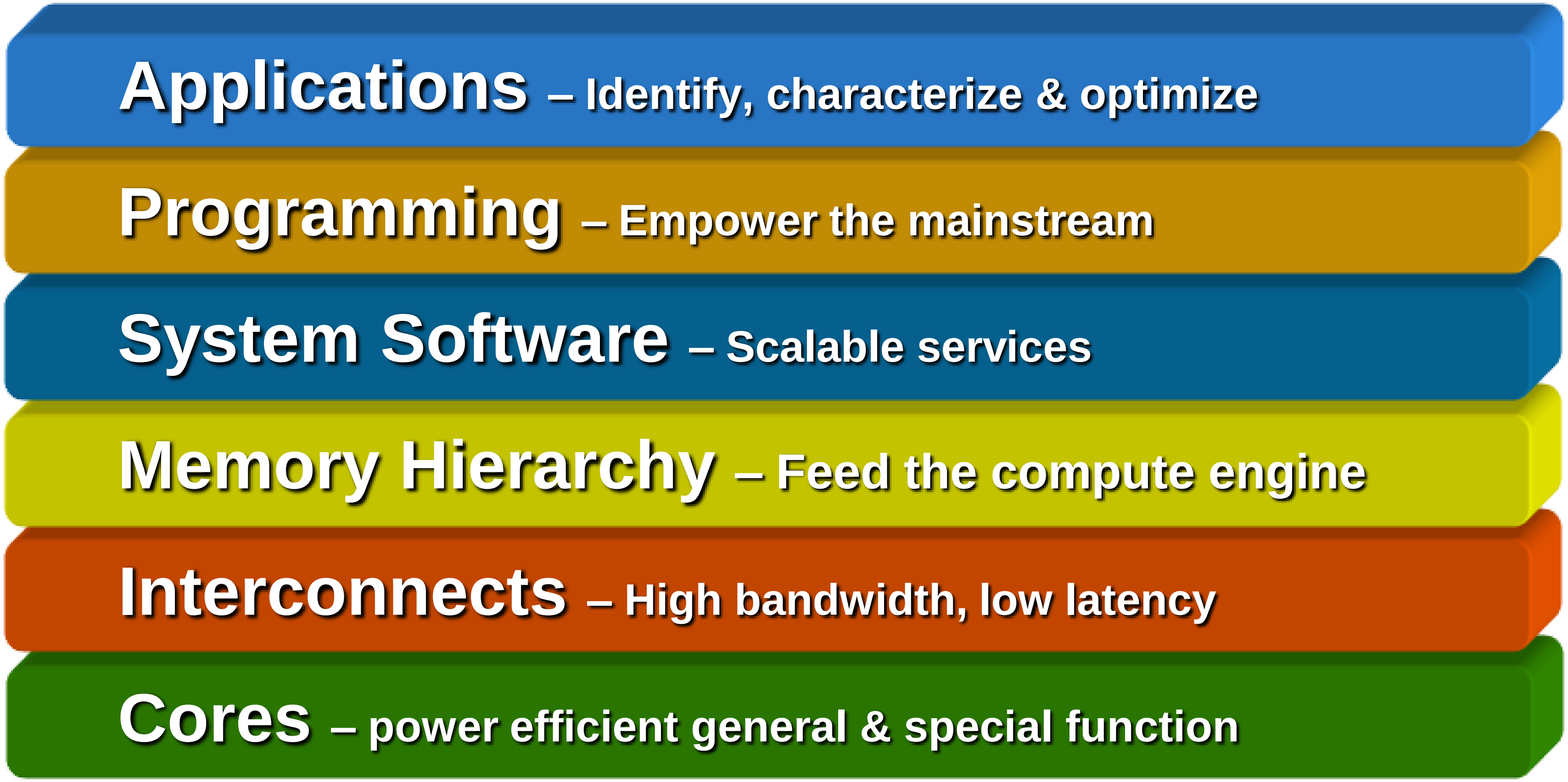


***In 2004, these observations led us to explore tera-scale***

# A Tera-scale Platform Vision



# Tera-scale Research



**Applications** – Identify, characterize & optimize

**Programming** – Empower the mainstream

**System Software** – Scalable services

**Memory Hierarchy** – Feed the compute engine

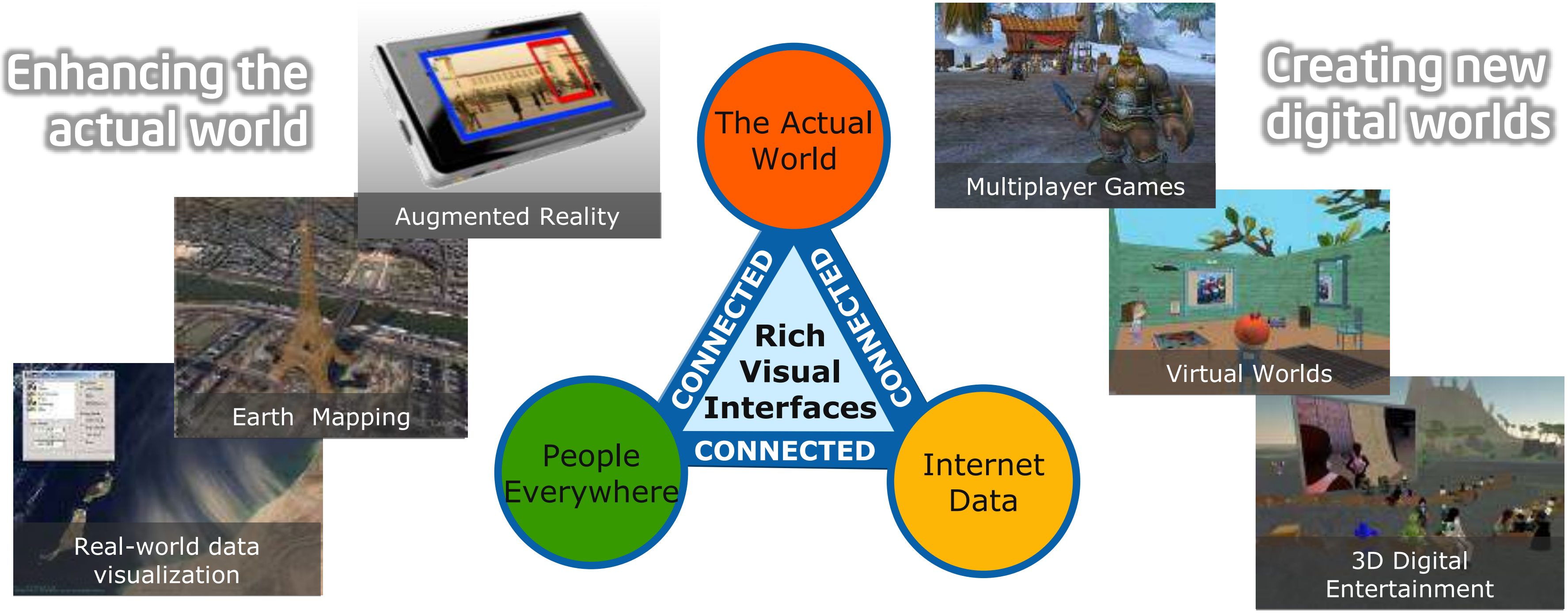
**Interconnects** – High bandwidth, low latency

**Cores** – power efficient general & special function

# Future 3D Internet

## Immersive Connected Experiences

*Bringing the richness of Visual Computing to **connected** usage models such as social networking, collaboration, online gaming, & online retail.*



Enhancing the actual world

Augmented Reality

Earth Mapping

Real-world data visualization

The Actual World

CONNECTED

CONNECTED

Rich Visual Interfaces

CONNECTED

People Everywhere

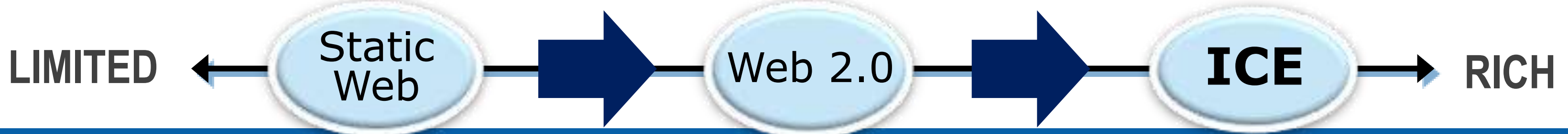
Internet Data

Creating new digital worlds

Multiplayer Games

Virtual Worlds

3D Digital Entertainment



LIMITED

Static Web

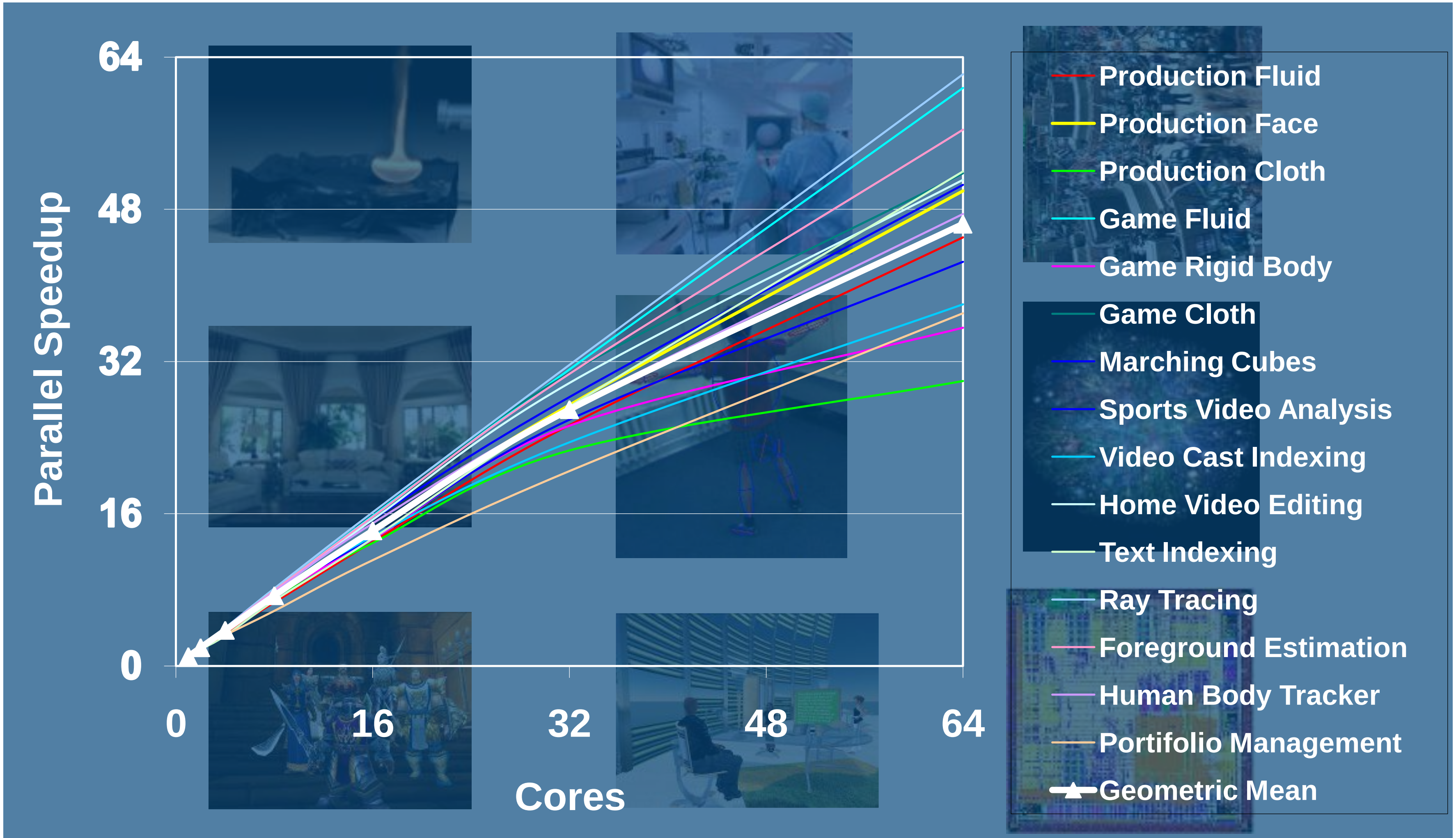
Web 2.0

ICE

RICH

*Better content quality, social interaction*

# Application Kernel Scaling



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

# Platform Performance Demands

## Emerging ICE applications

**SERVERS: 10x More Work**

75%+ Time = Compute Intensive Work

| TYPE | SOFTWARE | MAX CLIENTS PER SERVER |
|---|---|---|
| MMORPGS | WoW | 2500 |
| VWs | Second Life | 160 |

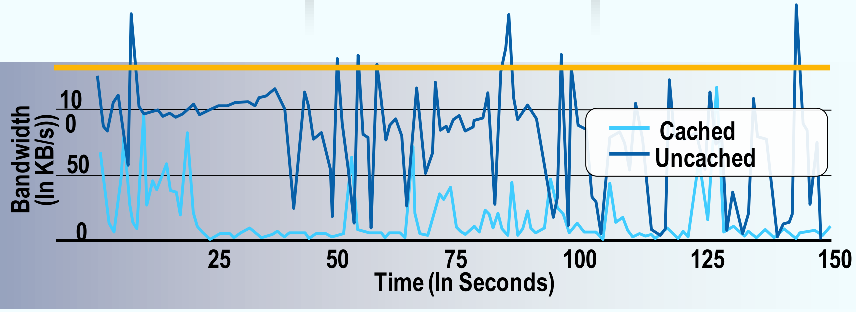
**CLIENTS: 3x CPU, 20x GPU**

65%+ Time = Compute Intensive Work

| APPLICATION | % CPU UTILIZATION | % GPU UTILIZATION |
|---|---|---|
| 2D Websites | 20 | 0-1 |
| Google Earth | 50 | 10-15 |
| Second Life | 70 | 35-75 |

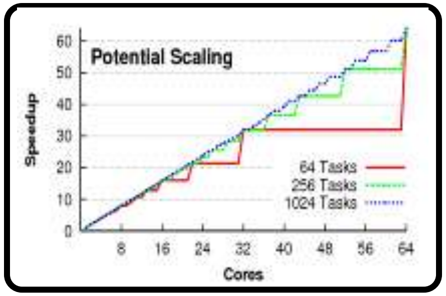
**NETWORK: 100x Bandwidth**

Maximum Bandwidth Limited by Server to Client



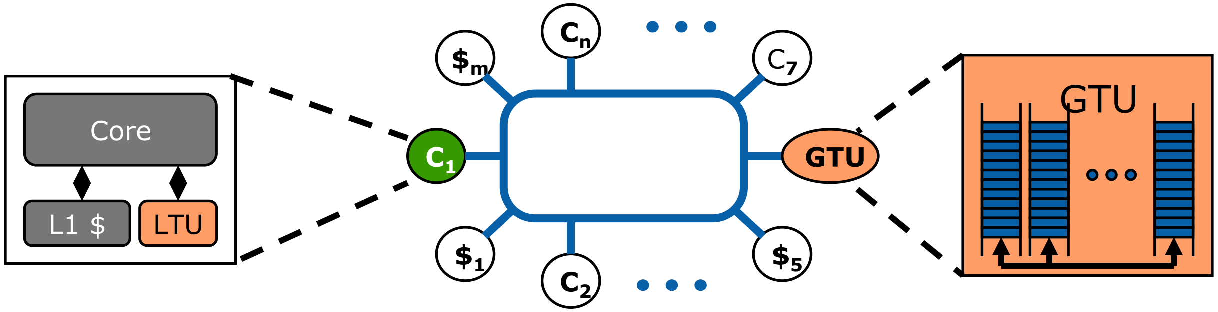
*Sources: WoW data (source www.warcraftrealms.com), Second Life data (source CTO-CTO meeting and www.secondlife.com), and Intel measurements.*

# Application Acceleration: HW Task Queues



Task Queues

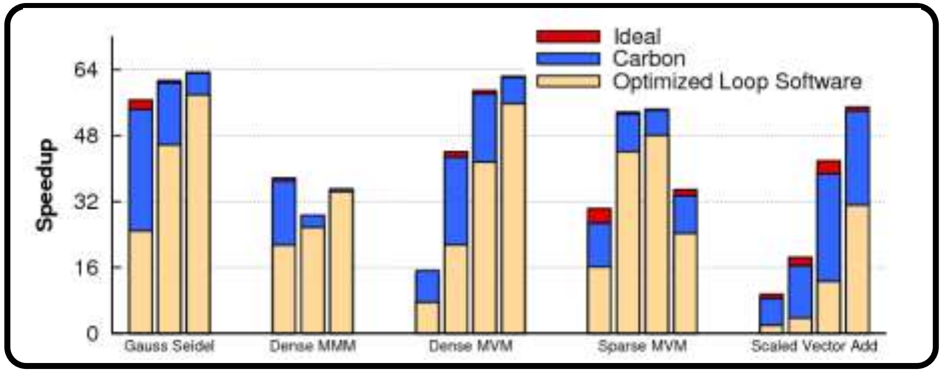
- scale effectively to many cores
- deal with asymmetry
- supports task & loop parallelism



Local Task Unit (LTU)
Prefetches and buffers tasks

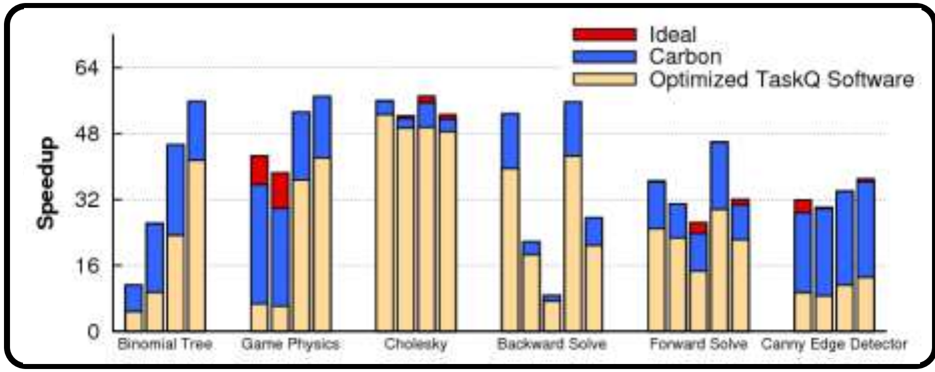
Global Task Unit (GTU)
Caches the task pool
Uses distributed task stealing

## Loop Level Parallelism



88% benefit optimized S/W

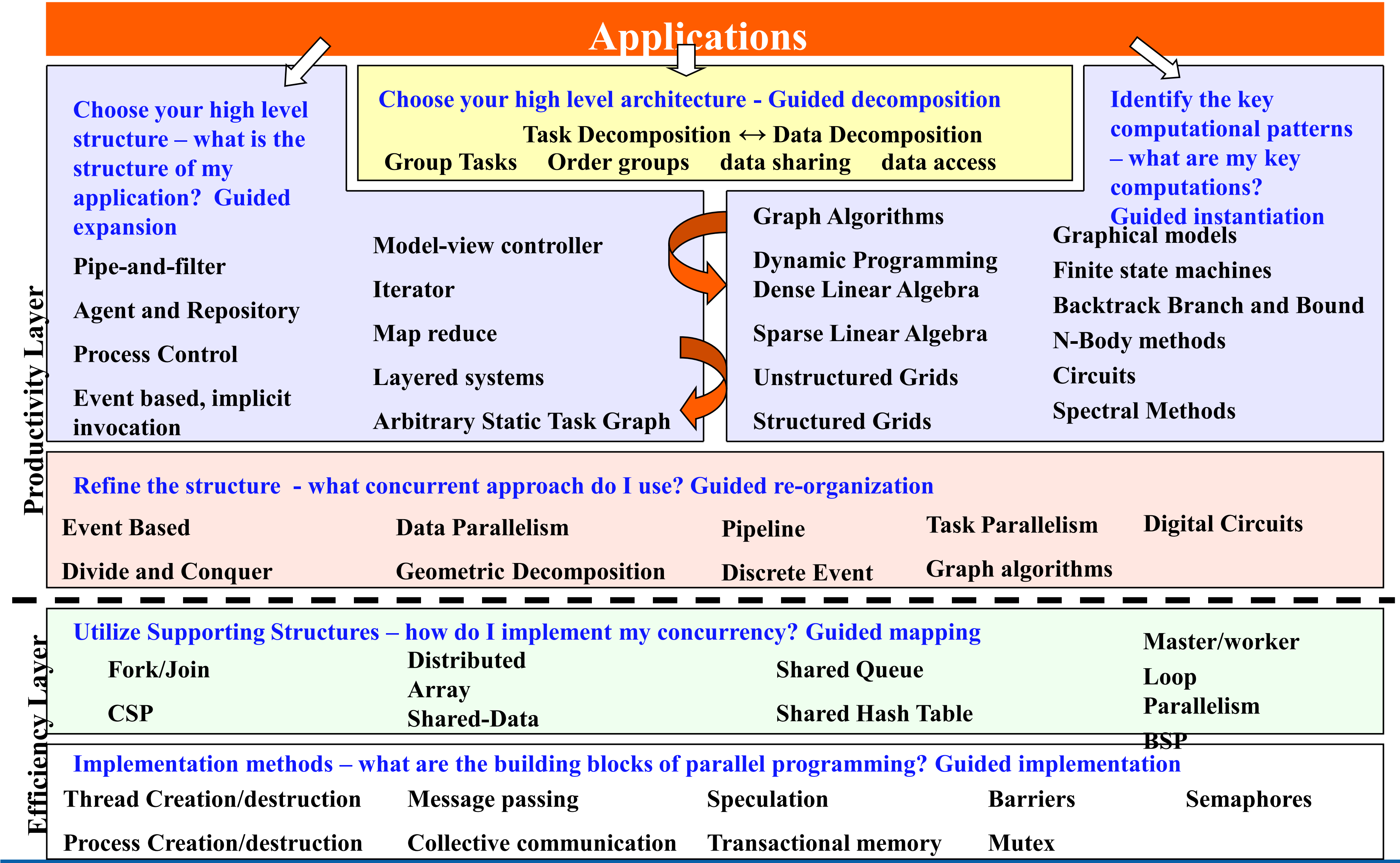
## Task Level Parallelism



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.

# Design Pattern Language



Applications

Productivity Layer

Choose your high level structure – what is the structure of my application? Guided expansion

Pipe-and-filter

Agent and Repository

Process Control

Event based, implicit invocation

Model-view controller

Iterator

Map reduce

Layered systems

Arbitrary Static Task Graph

Choose your high level architecture - Guided decomposition

Task Decomposition ↔ Data Decomposition

Group Tasks Order groups data sharing data access

Identify the key computational patterns – what are my key computations? Guided instantiation

Graph Algorithms

Dynamic Programming

Dense Linear Algebra

Sparse Linear Algebra

Unstructured Grids

Structured Grids

Graphical models

Finite state machines

Backtrack Branch and Bound

N-Body methods

Circuits

Spectral Methods

Refine the structure - what concurrent approach do I use? Guided re-organization

Event Based

Divide and Conquer

Data Parallelism

Geometric Decomposition

Pipeline

Discrete Event

Task Parallelism

Graph algorithms

Digital Circuits

Efficiency Layer

Utilize Supporting Structures – how do I implement my concurrency? Guided mapping

Fork/Join

CSP

Distributed Array

Shared-Data

Shared Queue

Shared Hash Table

Master/worker

Loop Parallelism

BSP

Implementation methods – what are the building blocks of parallel programming? Guided implementation

Thread Creation/destruction

Process Creation/destruction

Message passing

Collective communication

Speculation

Transactional memory

Barriers

Mutex

Semaphores

Keutzer , K., Mattson, T.: "A Design Pattern Language for Engineering (Parallel) Software" to appear in Intel Technology Journal.

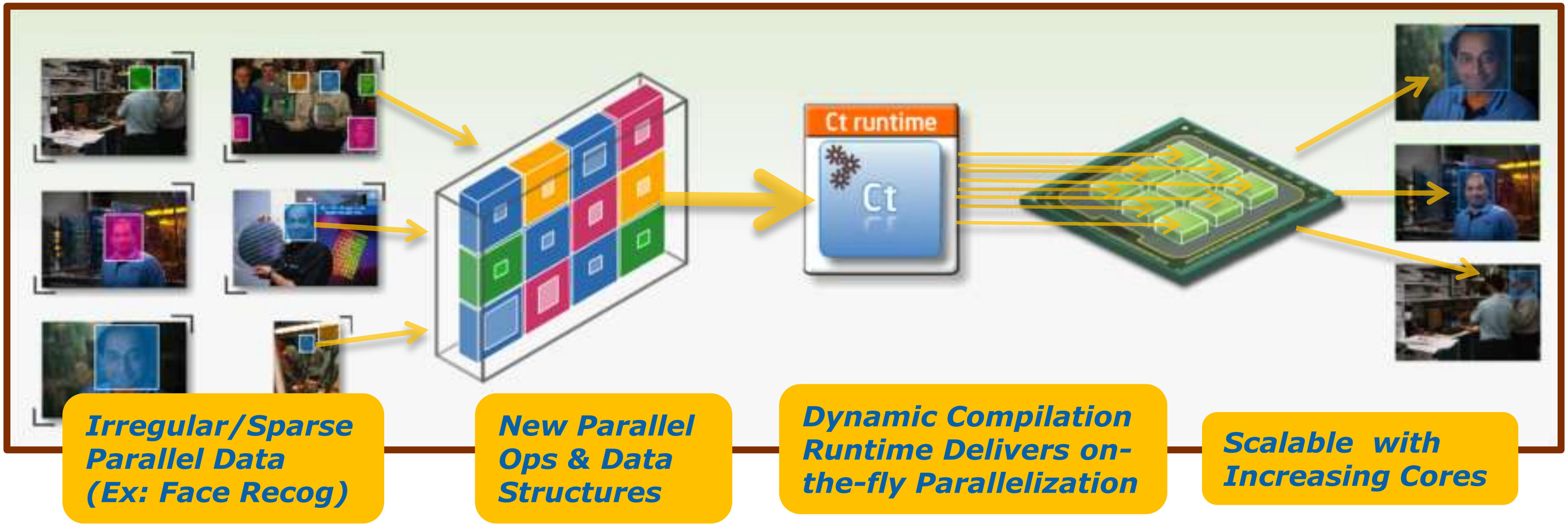
# Transactional Memory

- TM Definition - a sequence of memory operations that either execute completely (commit) or have no effect (abort)
- 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
- Intel C/C++ STM compiler
  - Use for experimentation & workload development
  - Downloadable from http://whatif.intel.com
- Draft specification adding TM to C++
  - Jointly authored with IBM & Sun
  - Discussion on tm-languages@googlegroups.com

Shpeisman, T. et. Al *Towards Transactional Memory Semantics for C++*,
SPAA'09, August 11–13, 2009, Calgary, Alberta, Canada.

# Ct Technology

## An example of Intel turning research into reality



*Irregular/Sparse Parallel Data (Ex: Face Recog)*

*New Parallel Ops & Data Structures*

*Dynamic Compilation Runtime Delivers on-the-fly Parallelization*

*Scalable with Increasing Cores*

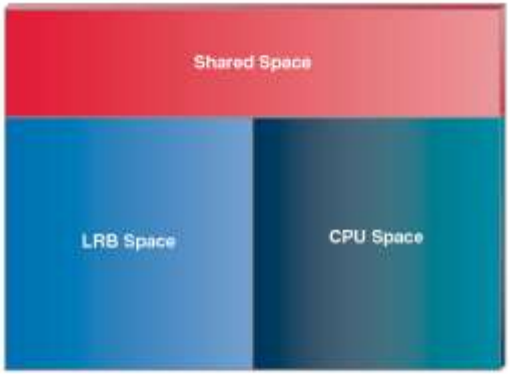
**A new programming model, abstract machine and API**

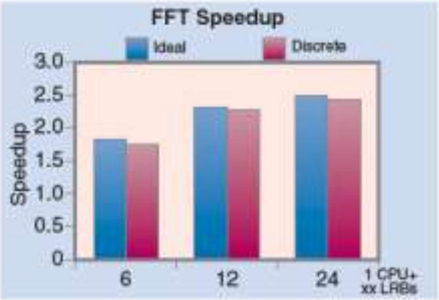
- Expresses data parallelism with sequential semantics
- Deterministic (race-free) parallel programming
- Degree of thread/vector parallelism targeted dynamically according to user's multi-core and SIMD hardware
- Extends C++: Uses templates for new types, operator overloading and lib calls for new operators

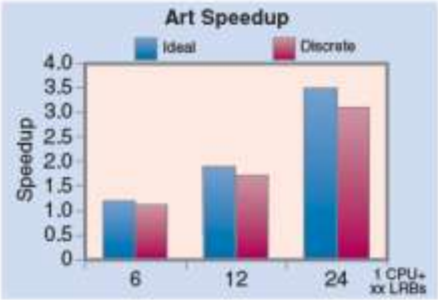
http://software.intel.com/en-us/data-parallel/

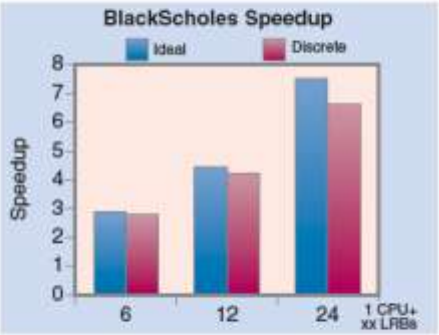
# Heterogeneous platform support

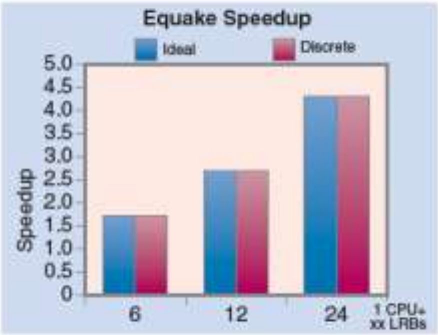
- Shared virtual memory in a mixed ISA, multiple OS environment
- Simplified programming model with data structure and pointer sharing

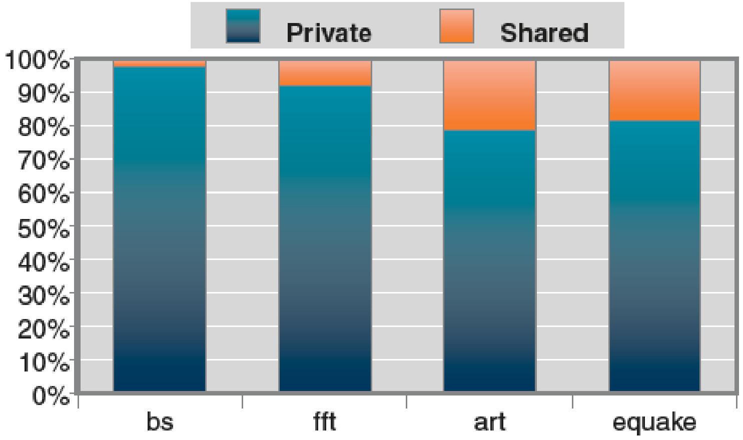








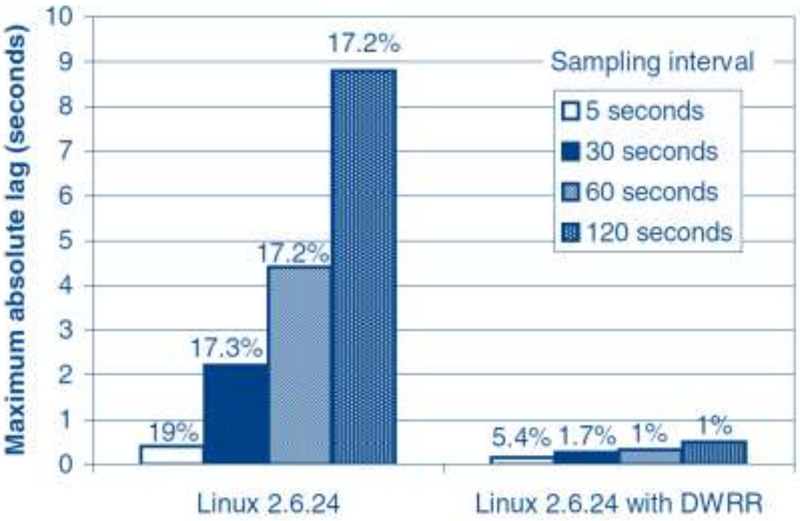




Saha, Bratin, et al. “Programming Model for a Heterogeneous x86 Platform.”
PLDI ’09 June 15-20, 2009, Dublin, Ireland,

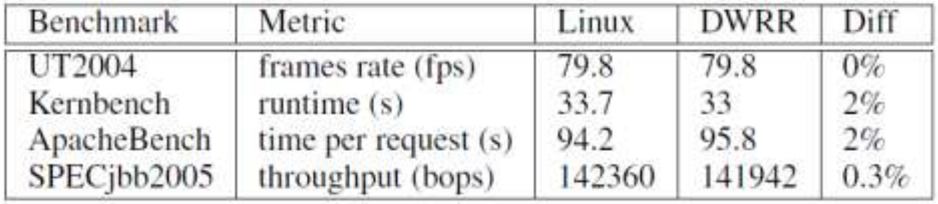
# OS Scheduling

## Fairness on Multi-core



**Maximum lag and relative error for 16 threads on 8 cores, 5 threads have *nice* one.**

**Performance on Benchmarks v Linux CFS alone**



| Benchmark | Metric | Linux | DWRR | Diff |
|---|---|---|---|---|
| UT2004 | frames rate (fps) | 79.8 | 79.8 | 0% |
| Kernbench | runtime (s) | 33.7 | 33 | 2% |
| ApacheBench | time per request (s) | 94.2 | 95.8 | 2% |
| SPECjbb2005 | throughput (bops) | 142360 | 141942 | 0.3% |

- Distributed Weighted Round Robin scheduling
  - *Accurate fairness*
  - *Efficient and scalable operation*
  - *Flexible user control*
  - *High performance*
- Implementation
  - Additional queue per core
  - Monitor runtime per round
  - Balance across cores
- Evaluated against Linux
  - O(1) scheduler 2.6.22.15
  - CFS scheduler 2.6.24

Li, Tong, et al. Efficient and Scalable Multiprocessor Fair Scheduling Using Distributed Weighted Round-Robin. PPoPP’09, February 14–18, 2009, Raleigh, North Carolina, USA

# On-Die Fabric Research

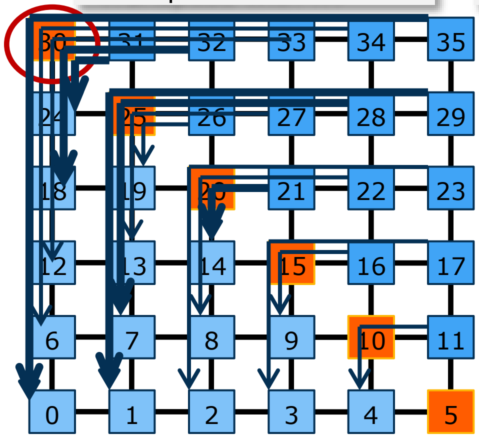
## Adaptive Routing

Adversarial traffic can severely affect network throughput

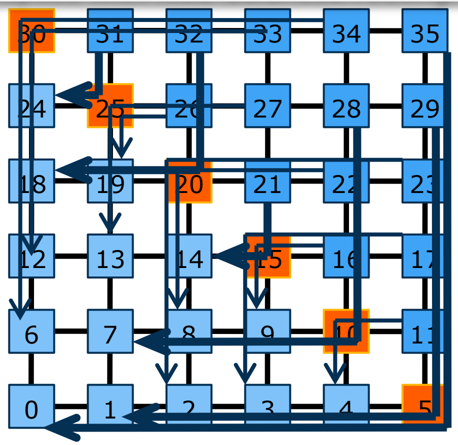
Example: Matrix transpose

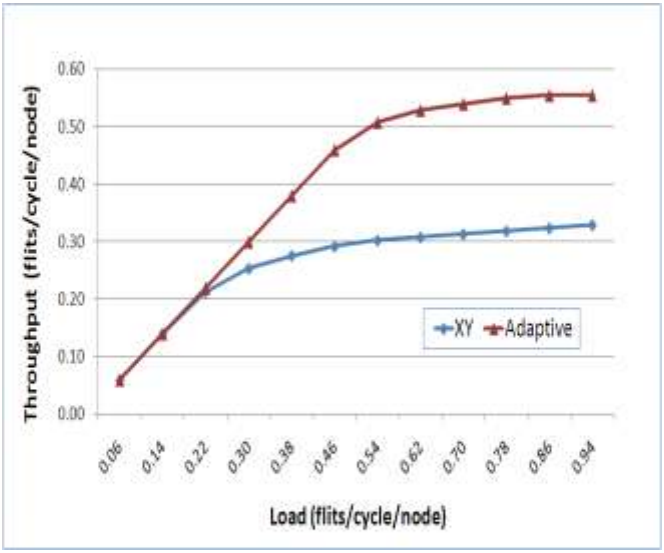
- XY routing does not use all available paths

Transpose with XY



Transpose with load-balanced routing



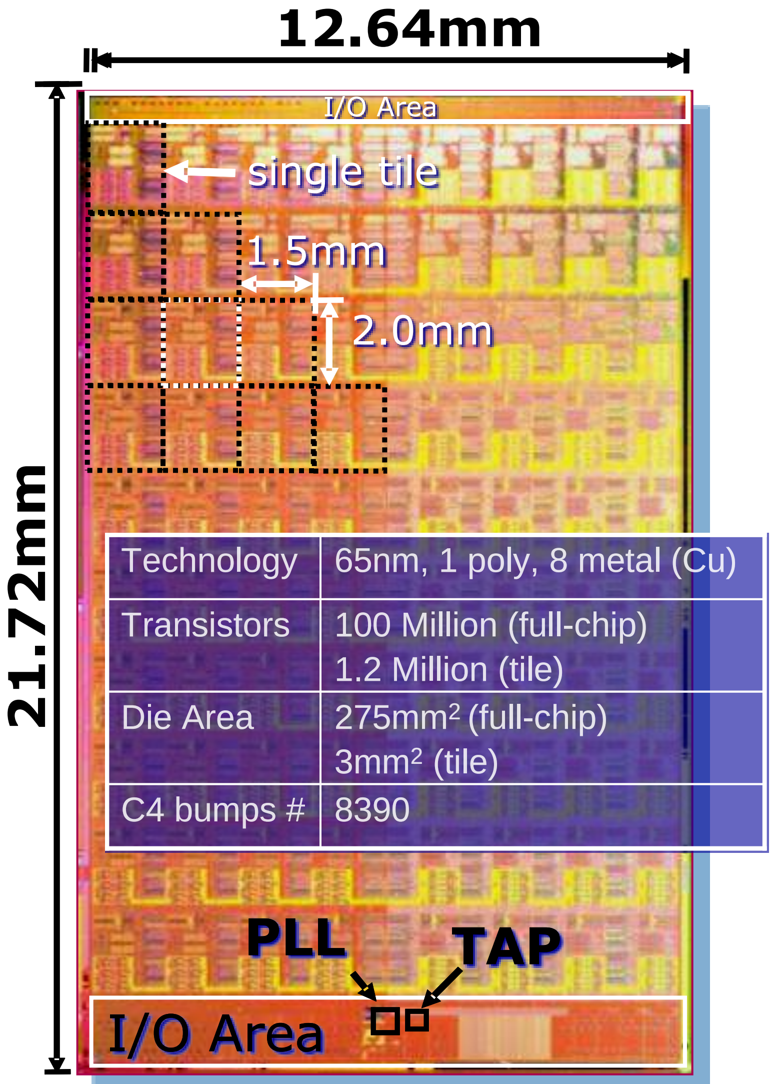


Flexible routing allows all paths to be used

A fully-adaptive scheme provides the best throughput under different traffic patterns but has to consider large set of constraints, e.g.

- No resource bifurcation (required by virtual networks)
- Minimal storage/power overhead
- Zero latency impact

# Teraflops Research Processor

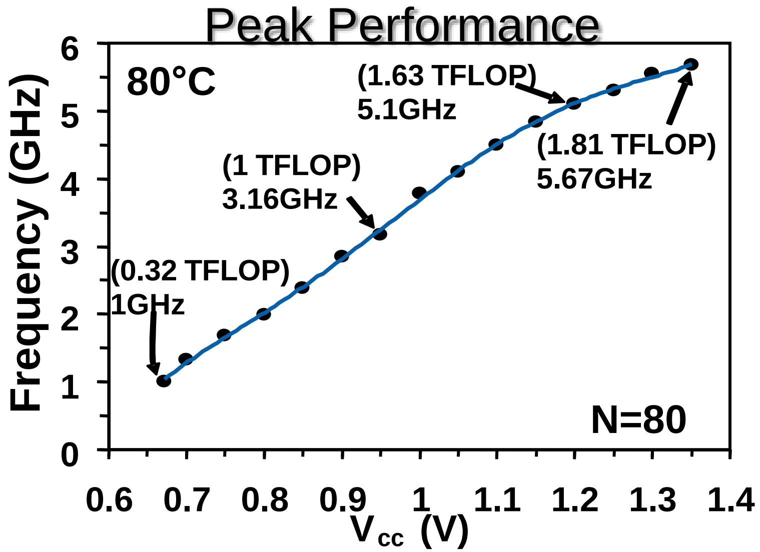


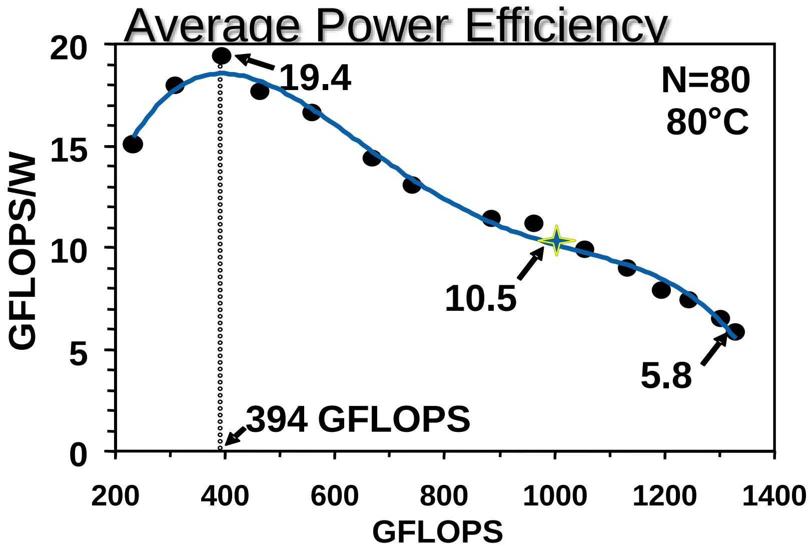
| | |
|---|---|
| Technology | 65nm, 1 poly, 8 metal (Cu) |
| Transistors | 100 Million (full-chip)<br>1.2 Million (tile) |
| Die Area | 275mm$^2$ (full-chip)<br>3mm$^2$ (tile) |
| C4 bumps # | 8390 |

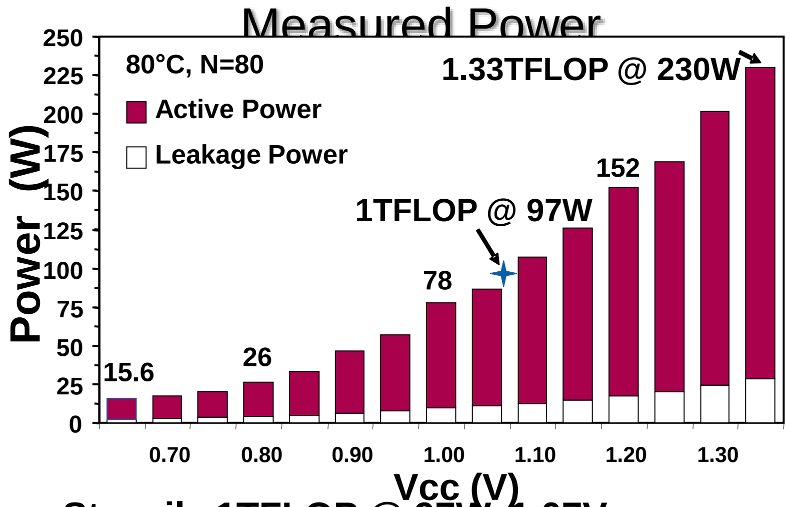
**Goals:**

- 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

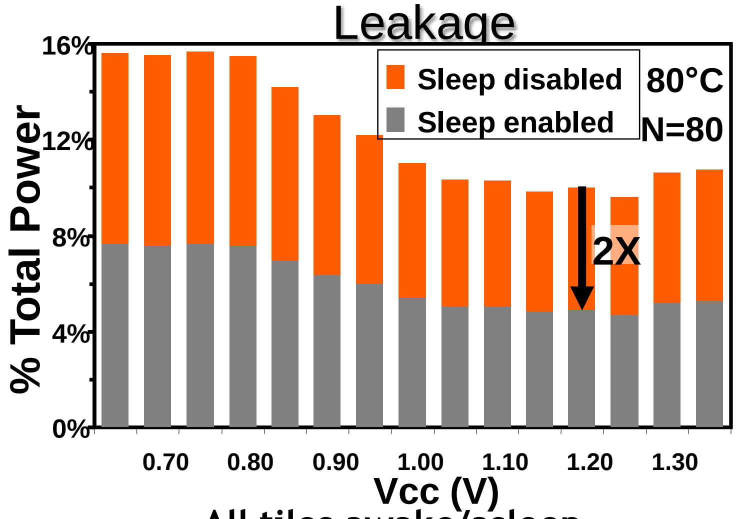
# Power Performance Results







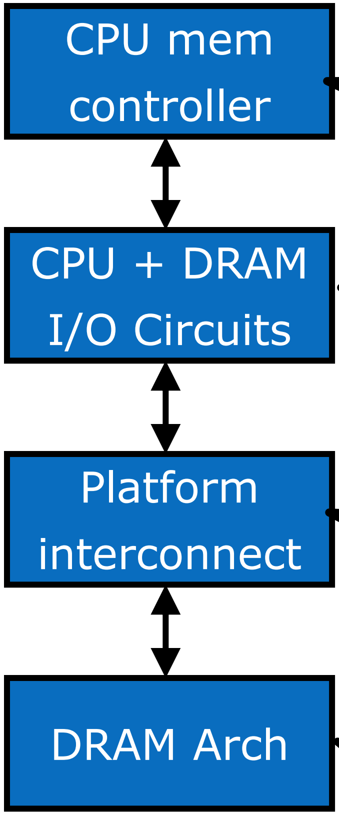
Stencil: 1TFLOP @ 97W, 1.07V;



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.

# Memory



CPU mem controller

CPU + DRAM I/O Circuits

Platform interconnect

DRAM Arch

**Reduce memory controller power & complexity**

- Increased number of banks per channel
- Increased concurrency for accessing memory array
- Scalable & flexible across a wide range of market segments
- Lower latency of on-die cache miss to data returned from DRAM

**Optimized for power efficiency & silicon cost**

- Very low I/O power
- Aggressive power management
- Small silicon area and low complexity
- Scalability in bandwidth and width

**Develop (prove out) new platform physicals**

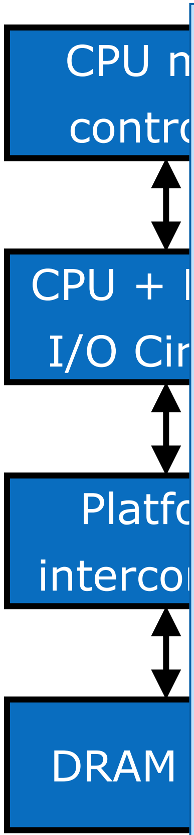
- High density, small form factor
- Low loss & reflections
- Headroom for future scaling
- Low cost & modularity

**Optimize for low power, high concurrency**

- Very low energy per bit read and written
- Investigate the role of three dimensional stacked products
- Work with DRAM vendors

O'Mahoney, F., et al., "A 27Gb/s Forwarded-Clock I/O Receiver using an injection-Locked LC-DCO in 45nm CMOS," in *Proceedings of ISSCC 2008(IEEE International Solid-State Circuits Conference)*, Feb. 12, 2008.

# Memory



CPU m

contro

CPU + I

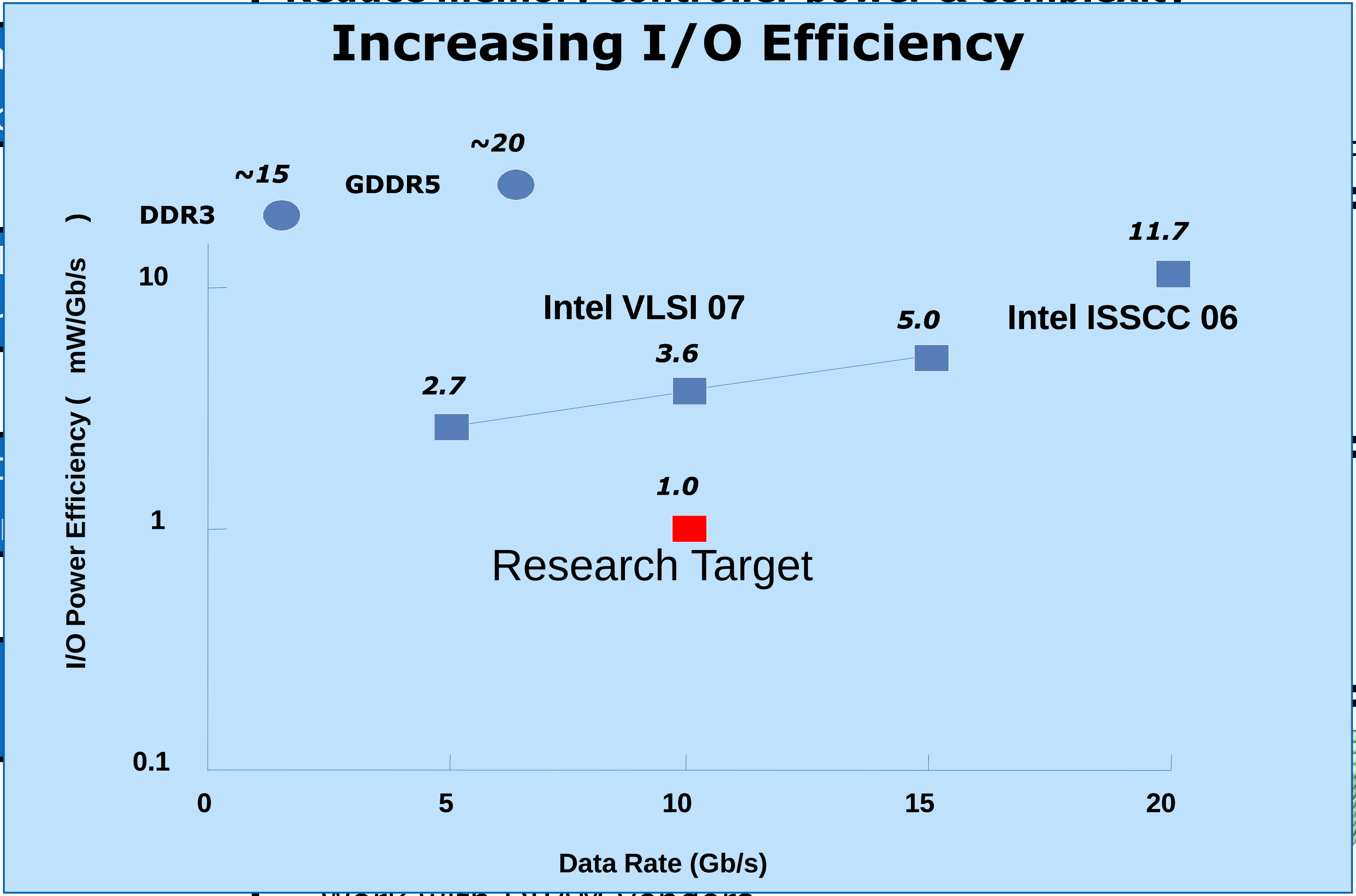
I/O Cir

Platfo

interco

DRAM

**Reduce memory controller power & complexity**



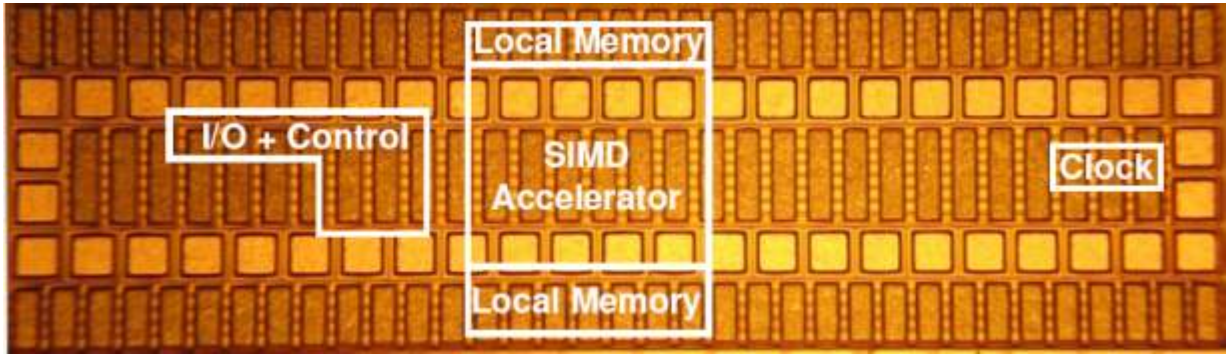
## Increasing I/O Efficiency

RAM

- Work with DRAM vendors

O'Mahoney, F., et al., "A 27Gb/s Forwarded-Clock I/O Receiver using an injection-Locked LC-DCO in 45nm CMOS," in *Proceedings of ISSCC 2008(IEEE International Solid-State Circuits Conference)*, Feb. 12, 2008.

# Increasing power efficiency
## Low-power scalable SIMD Vector processing

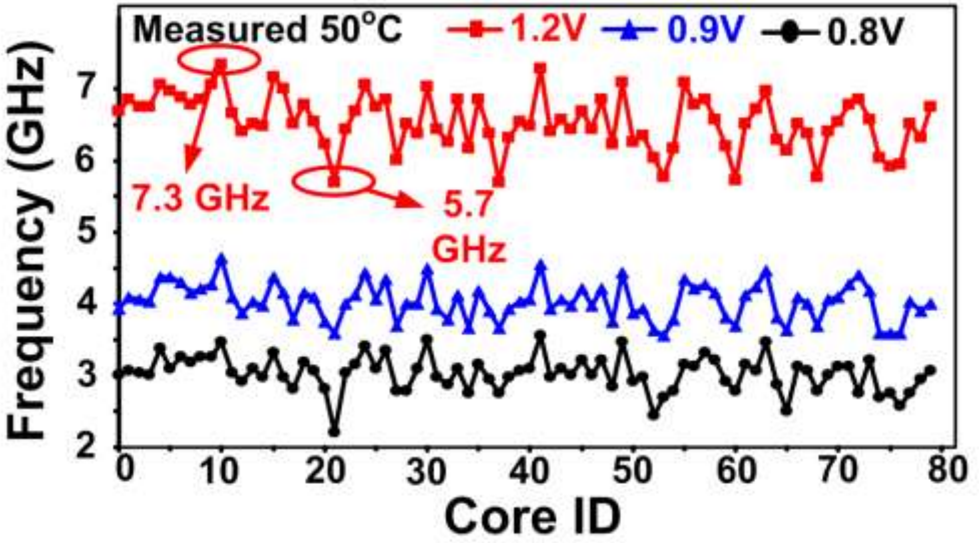


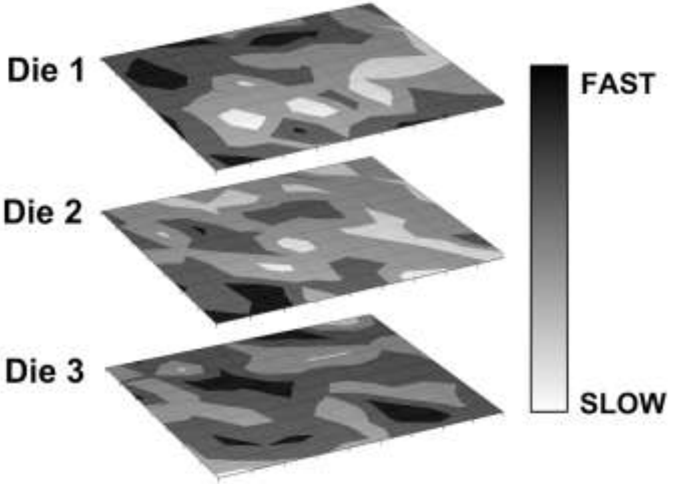
Energy-efficiency up to **10X better** at normal voltages
Up to **80x better** at ultra-low voltages

- 45nm CMOS occupies 0.081mm2
- Signed 32b multiply using reconfigurable16b multipliers and adder circuits
- Operation from 1.3V down to ultra-low 230MV
- 2.3GHz, 161mW operation at 1.1V
- Peak SIMD energy efficiency of 494GOPS/W measured at 300mV, 50°C.

Kaul, H., et al., "A 300mV 494GOPS/W Reconfigurable Dual-Supply 4-way SIMD Vector Processing Accelerator in 45nm CMOS," in *Proceedings of ISSCC 2009 (IEEE International Solid-State Circuits Conference)*, Feb. 2009.

# Within-Die Variation-Aware DVFS and scheduling



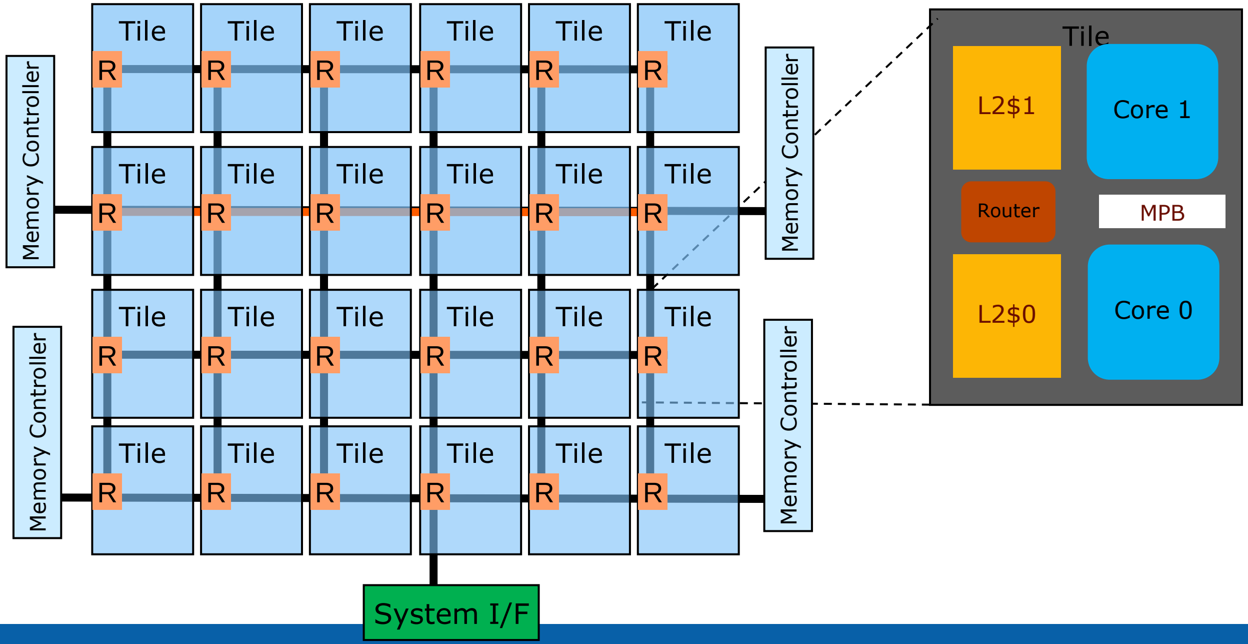


- Max Frequency variation per core 28% at 1.2V 62% at 0.8V
- No correlation die to die – individual characterization required
- Improved performance or energy efficiency with:
  - Multiple frequency islands
  - Dynamic scheduling of processing to core

Dighe, S, et al., "Within-Die Variation-Aware Dynamic Voltage-Frequency Scaling, Core Mapping and Thread Hopping for an 80-Core Processor", in *Proceedings of ISSCC 2010 (IEEE International Solid-State Circuits Conference)*, Feb. 2010

# Experimental Single-chip Cloud Computer

- Experimental many-core CPU on 45nm Hi-K metal-gate silicon
- 48 IA-compatible cores – the most ever built on a single chip
- Message-passing architecture – no HW cache coherence
- Research Vehicle
  - Fine-grained software-controlled power management
  - Scale-out programming models on-die



Howard, J, et al., "A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS", in *Proceedings of ISSCC 2010 (IEEE International Solid-State Circuits Conference)*, Feb. 2010

# 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 academic research

- Intel is developing hardware and software technologies to enable Tera-scale computing

# Q&A