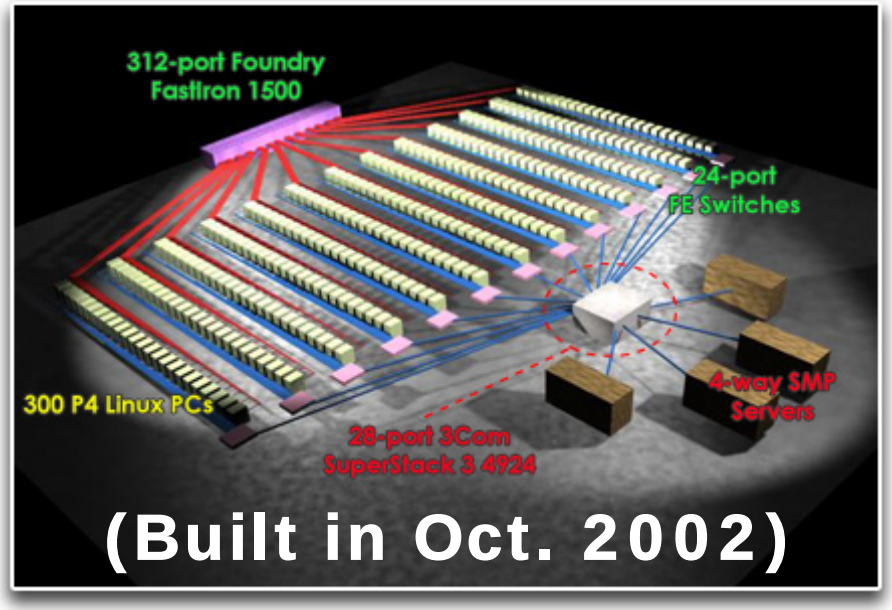
System Software Challenges for Big Data Computing

Cho-Li Wang
The University of Hong Kong

Systems Research Group @HKU

Our Motto: solving REAL problems with the use of REAL computing resources

“Self-Made” Gideon 300 cluster in 2002



- 300 Pentium4 PCs @355 Gflops;
- Ranked #170 in TOP500 (11/2002), #3 in China.
- The highest ranking in the TOP500 list among all machines from Hong Kong academic institutions in history.

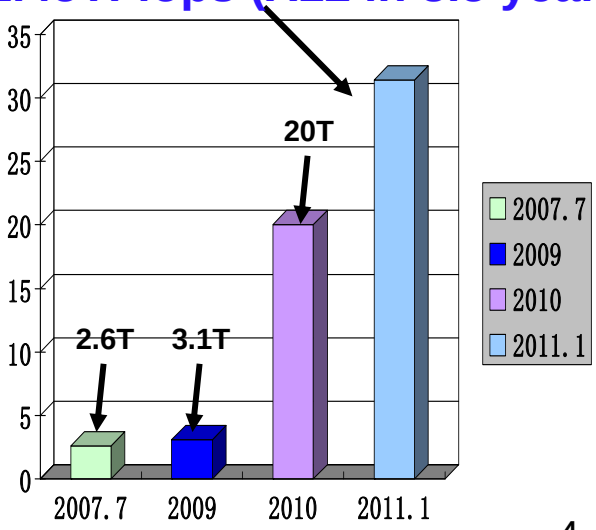
HKU High-Performance Computing Lab.

- Total # of cores: 3004 CPU + 5376 GPU cores
- RAM Size: 8.34 TB
- Disk storage: 130 TB
- Peak computing power: 27.05 TFlops

CS Gideon-II & CC MDRP Clusters

GPU-Cluster (Nvidia M2050, "Tianhe-1a"): 7.62 Tflops

31.45TFlops (X12 in 3.5 years)



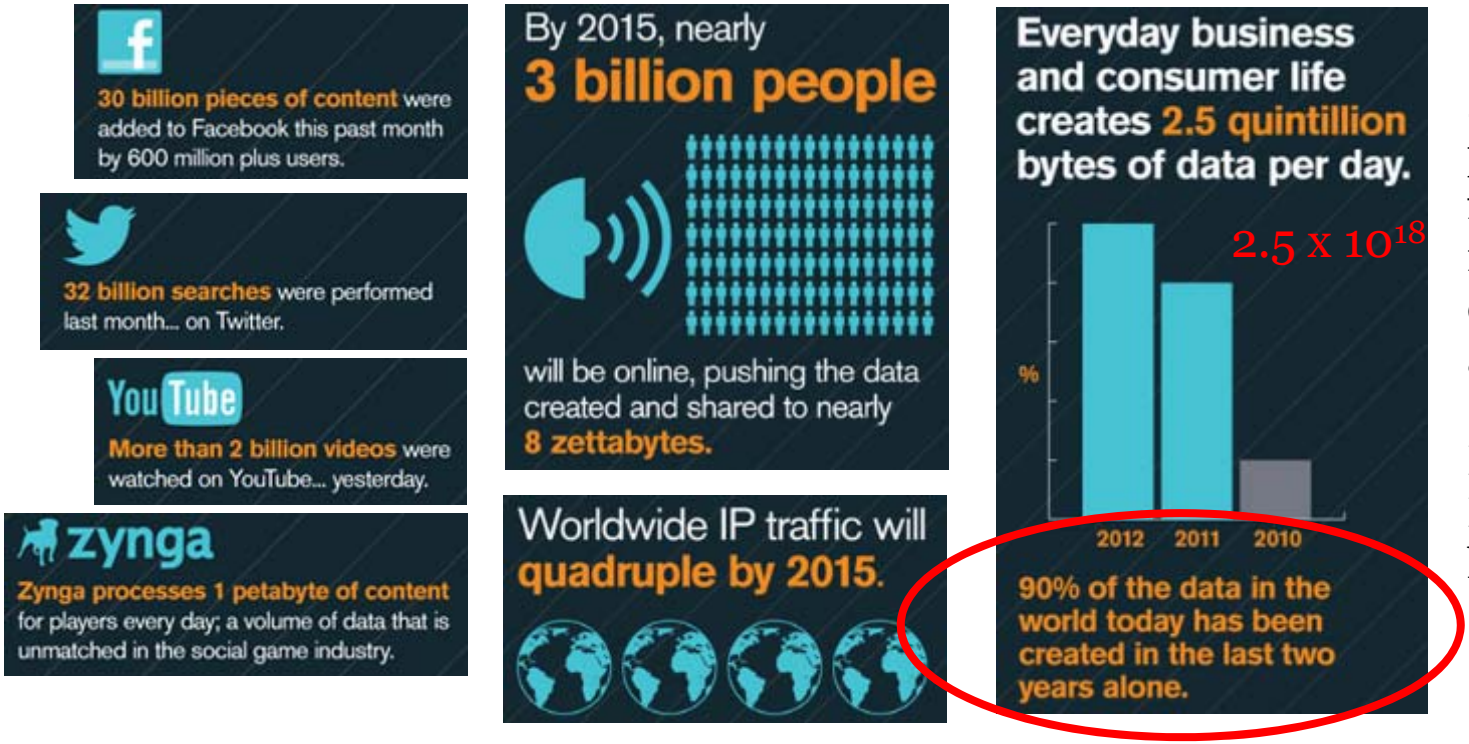
WHAT IS BIG DATA?

Big Data: The "3Vs" Model

- **High Volume** (amount of data)
- **High Velocity** (speed of data in and out)
- **High Variety** (range of data types and sources)

2010: 800,000 petabytes (would fill a stack of DVDs reaching from the earth to the moon and back)

By 2020, that pile of DVDs would stretch half way to Mars.



Google Trend:

Big Data vs. Data Analytics vs. Cloud Computing

- **McKinsey Global Institute (MGI) :**
  - Using big data, retailers could increase its operating margin by more than **60%**.
  - The U.S. could reduce its healthcare expenditure by **8%**
  - Government administrators in Europe could save more than **€100 billion** ($143 billion).

2012 CIO Agenda Findings

Success is contingent on anticipating the coming changes

CIO technologies	Ranking of technologies CIOs selected as one of their top 3 priorities in 2012				
Ranking	***2012***	**2011**	**2010**	**2009**	**2008**
Analytics and business intelligence (Big Data)	1	5	5	1	1
Mobile technologies	2	3	6	12	12
Cloud computing (SaaS, IaaS, PaaS)	3	1	2	16	*
Collaboration technologies (workflow)	4	8	11	5	8
Virtualization	5	2	1	3	3
Legacy modernization	6	7	15	4	4
IT management	7	4	10	*	*
Customer relationship management	8	18	*	*	*
ERP applications	9	13	14	2	2
Security	10	12	9	8	5
Social media/Web 2.0	11	10	3	15	15

*Not an option that year

2,335 CIOs from 37 industries across 45 countries

Outline

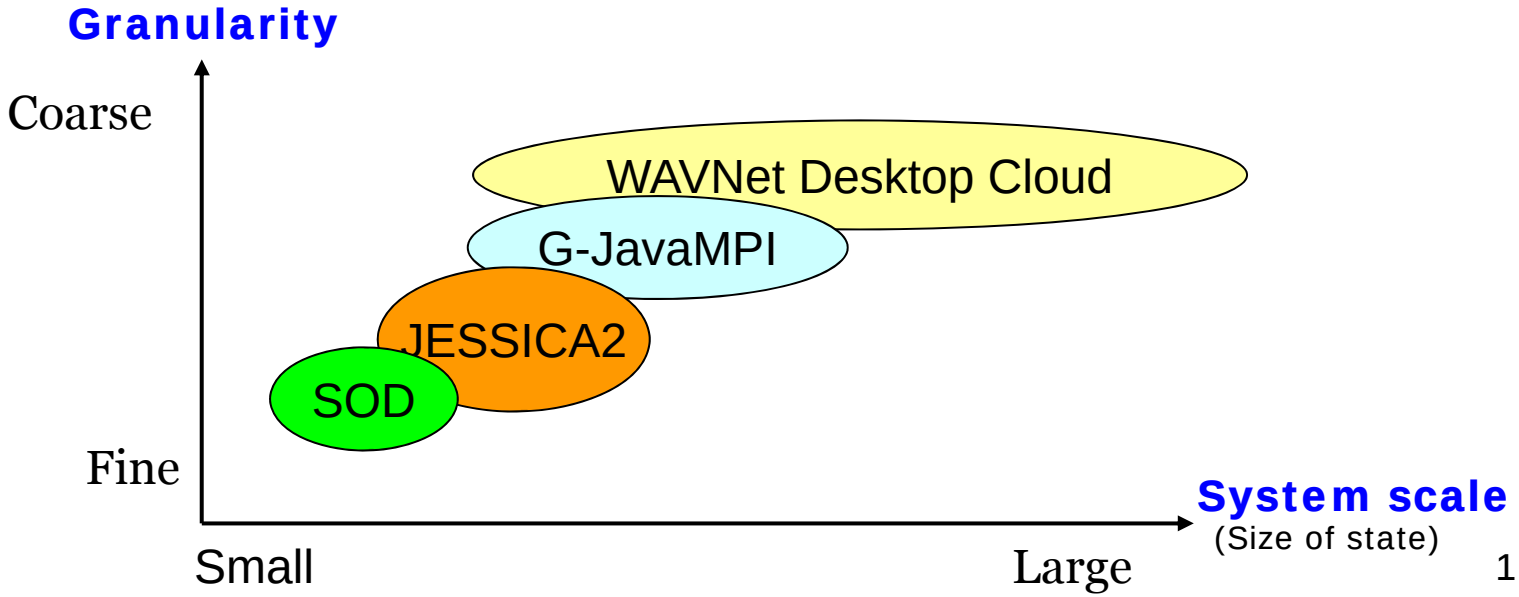
- Part I: Multi-granularity Computation Migration
- Part II: Heterogeneous Manycore Computing (CPUs+ GUPs)
- Part III: Big Data Computing on Future 1000-core Chips
- Part IV: From Data to Intelligence -- Context Reasoning

Part I

Multi-granularity Computation Migration

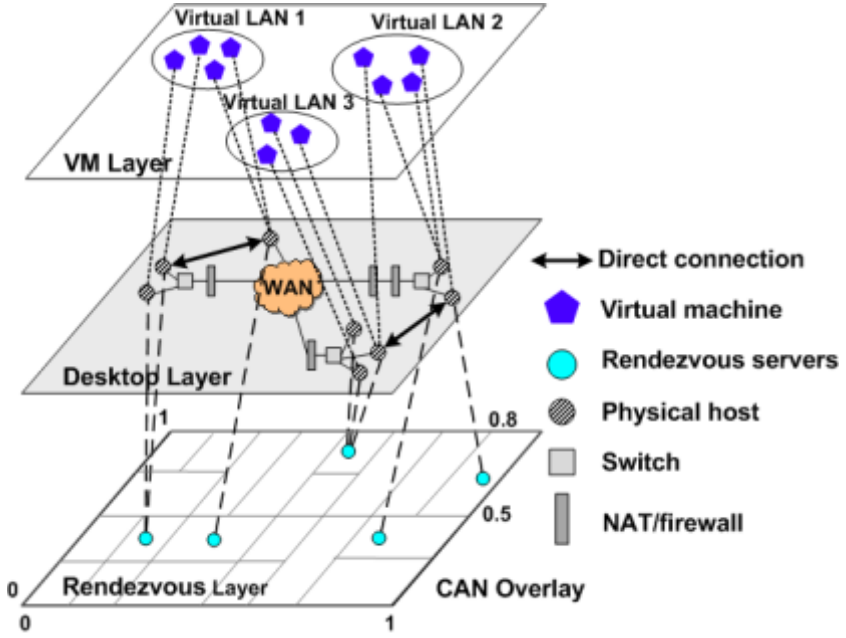
Multi-granularity Computation Migration

Granularity	Migration Technique (System)	Target System Type (Area)
Frame level	Stack-on-demand (SODEE)	Cloud, cloudlet, mobile network (WAN/LAN)
Thread level	Thread migration (JESSICA2)	Cluster (LAN)
Process level	Process migration (G-JavaMPI)	Grid (WAN/LAN)
VM level	Live VM migration (Xen)	Cluster (LAN)
	Wide-area live VM migration (WAVNet)	Cloud, p2p/desktop cloud (WAN)

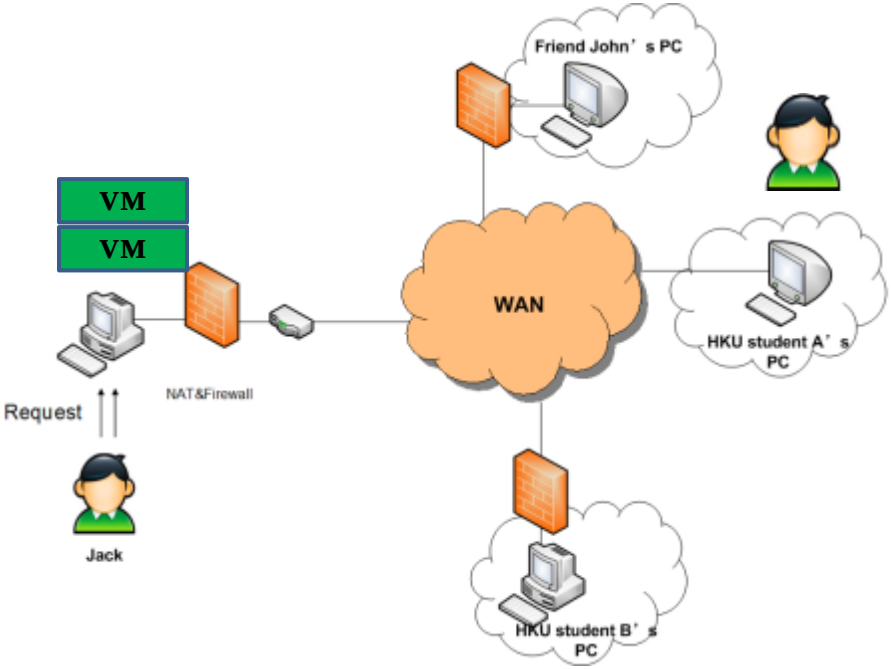


WAVNet: Live VM Migration over WAN

- A P2P Cloud with Live VM Migration over WAN
  - **"Virtualized LAN" over the Internet"**
- High penetration via NAT hole punching
  - Establish direct host-to-host connection
  - Free from proxies, able to traverse most NATs



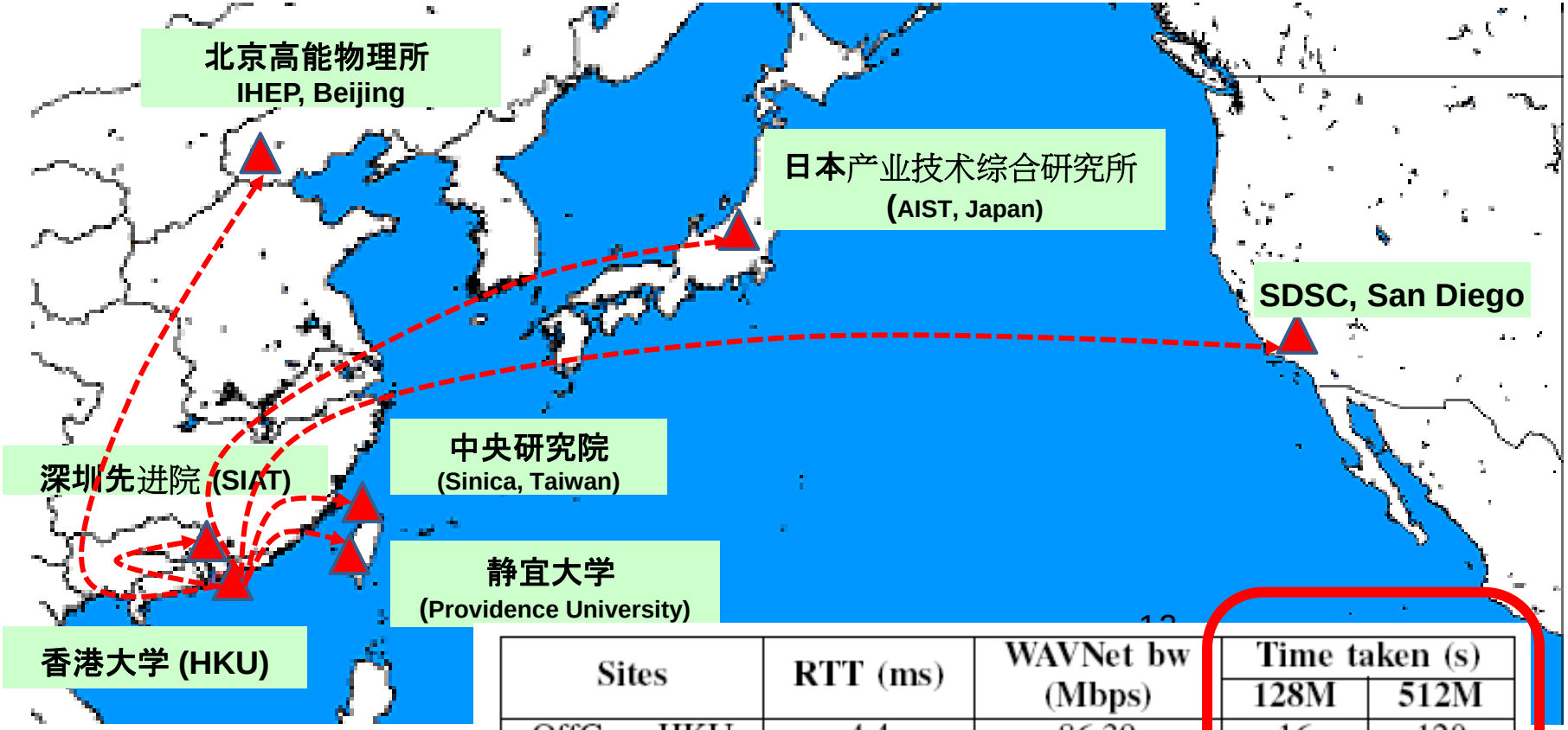
Key Members



Zheming Xu, Sheng Di, Weida Zhang, Luwei Cheng, and Cho-Li Wang, WAVNet: Wide-Area Network Virtualization Technique for Virtual Private Cloud, 2011 International Conference on Parallel Processing (ICPP2011)

WAVNet: Live VM Migration over WAN

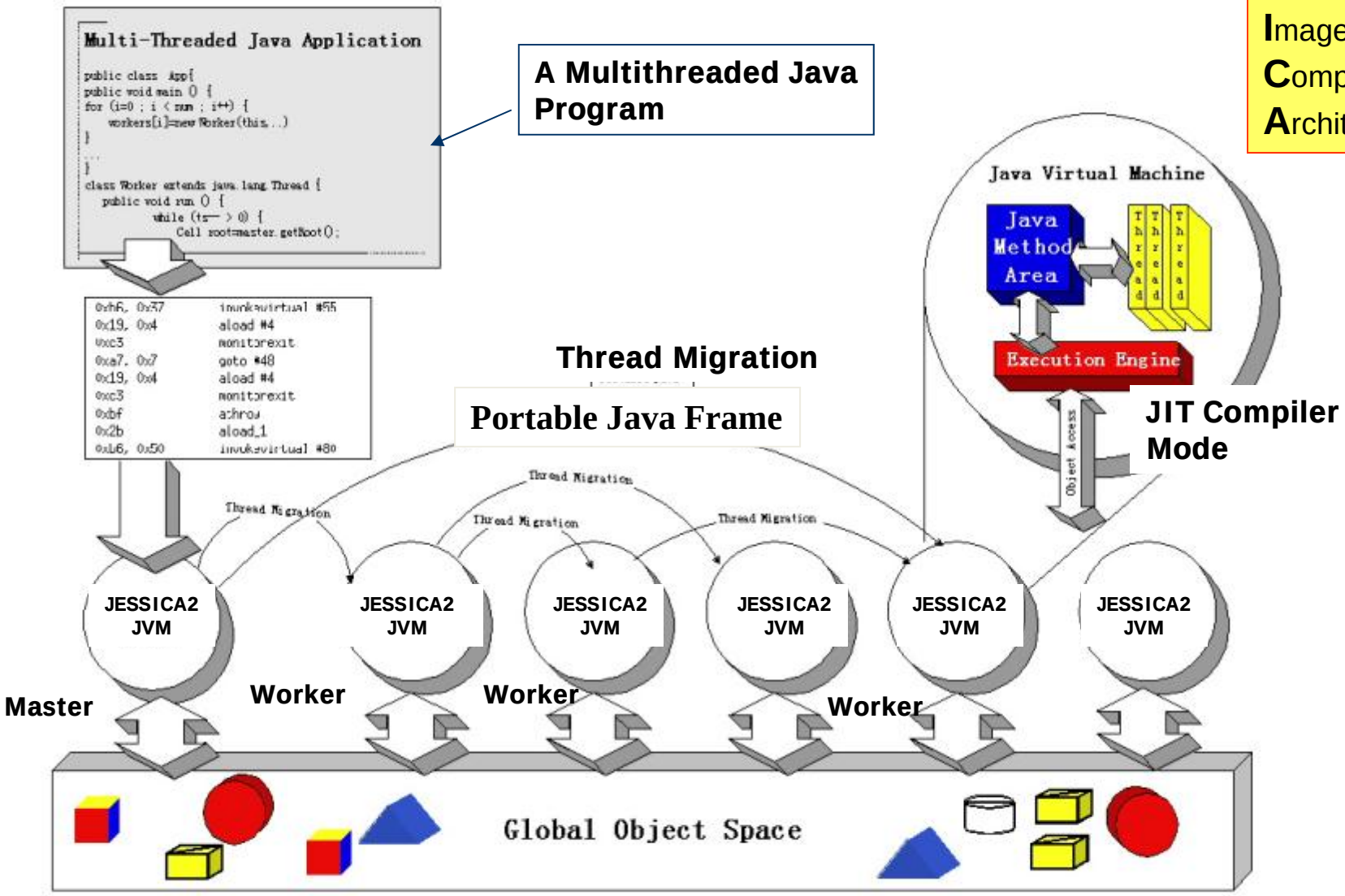
- Experiments at Pacific Rim Areas



Sites	RTT (ms)	WAVNet bw (Mbps)	Time taken (s)	
			128M	512M
OffCam-HKU	4.4	86.39	16	120
Sinica-HKU	24.8	42.93	92.5	202.5
AIST-HKU	75.8	55.1	107.5	208
SIAT-HKU	74.2	18.6	130	377.5
SDSC-HKU	217.2	27.17	310.5	1023

JESSICA2 Distributed JVM

Java
Enabled
Single
System
Image
Computing
Architecture



History and Roadmap of JESSICA Project

- **JESSICA V1.0 (1996-1999)**
  - Execution mode: **Interpreter Mode**
  - JVM kernel modification (Kaffe JVM)
  - Global heap: built on top of TreadMarks (Lazy Release Consistency + homeless)
- **JESSICA V2.0 (2000-2006)**
  - Execution mode: **JIT-Compiler Mode**
  - JVM kernel modification
  - Lazy release consistency + migrating-home protocol
- **JESSICA V3.0 (2008~2010)**
  - **Built above JVM (via JVMTI)**
  - Support Large Object Space
- **JESSICA v.4 (2010~)**
  - **Japonica** : Automatic loop parallization and speculative execution on GPU and multicore CPU
  - **TrC-DC** : a software transactional memory system on cluster with distributed clocks (not discussed)

J1 and J2 received a total of **1107** source code downloads

Past Members

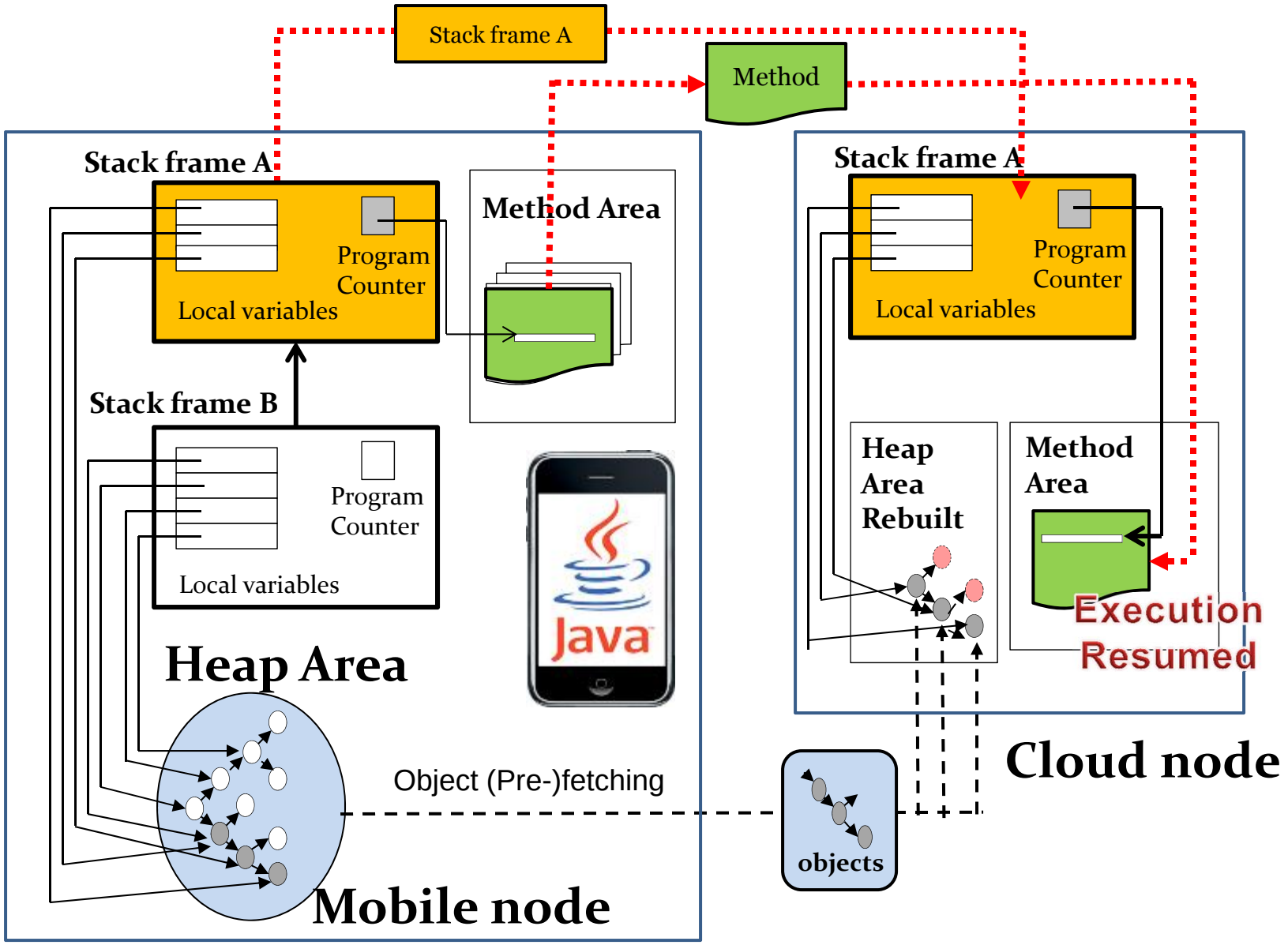
King Tin LAM,

Chenggang Zhang

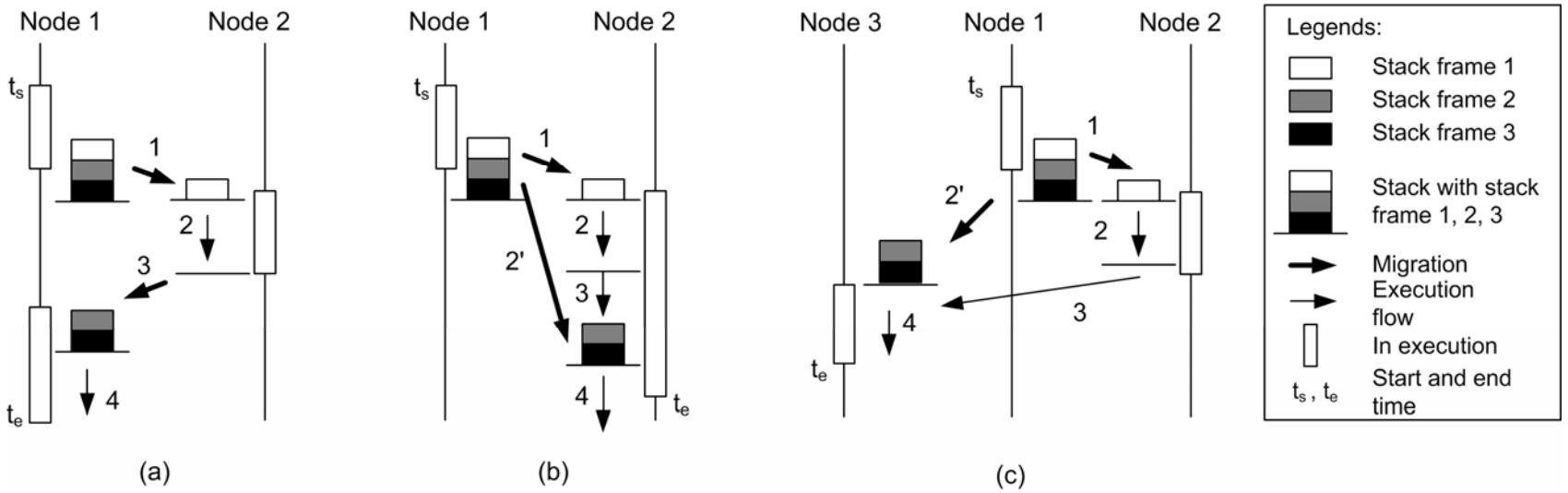
Kinson Chan

Ricky Ma

Stack-on-Demand (SOD)



Elastic Execution Model via SOD



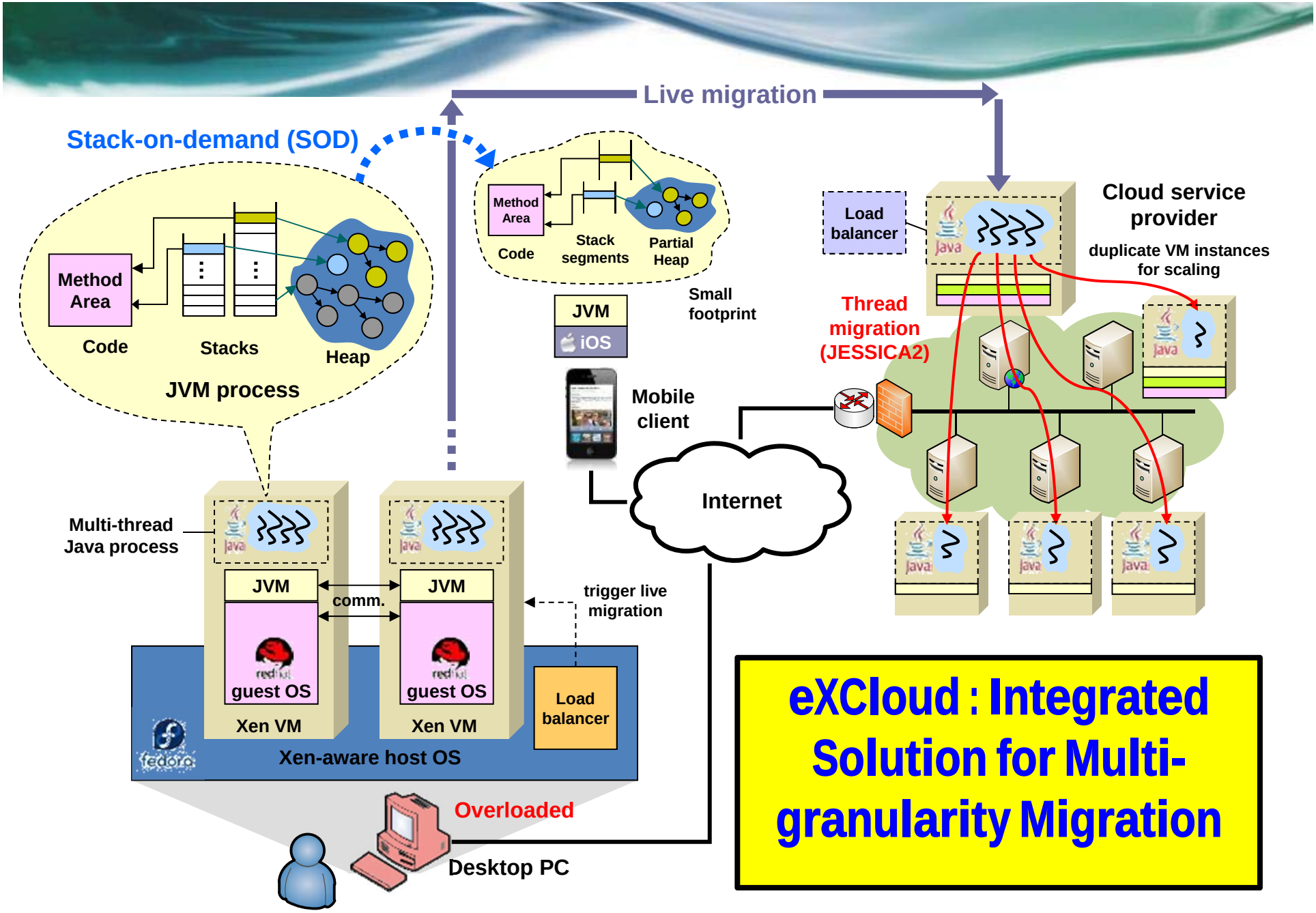
(a) "Remote Method Call"

(b) Mimic thread migration

(c) "Task Roaming": like a mobile agent roaming over the network or workflow

With such flexible or ***composable*** execution paths, SOD enables agile and elastic exploitation of distributed resources (storage), **a Big Data Solution !**

Lightweight, Portable, Adaptable



Ricky K. K. Ma, King Tin Lam, Cho-Li Wang, "eXCloud: Transparent Runtime Support for Scaling Mobile Applications," 2011 IEEE International Conference on Cloud and Service Computing (CSC2011),. (Best Paper Award)

Comparison of Migration Overhead

Migration overhead (MO)

= execution time w/ migration – execution time w/o migration

Sys / App	SOD on Xen (Stack mig.)			JESSICA2 on Xen (Thread mig.)			G-JavaMPI on Xen (Process mig.)			JDK on Xen (VM live mig.)		
	Exec. time (sec)		MO (ms)	Exec. time (sec)		MO (ms)	Exec. time (sec)		MO (ms)	Exec. time (sec)		MO (ms)
	w/ mig	w/o mig		w/ mig	w/o mig		w/ mig	w/o mig		w/ mig	w/o mig	
Fib	12.77	12.69	**83**	47.31	47.21	**96**	16.45	12.68	**3770**	13.37	12.28	1090
NQ	7.72	7.67	**49**	37.49	37.30	**193**	7.93	7.63	**299**	8.36	7.15	1210
TSP	3.59	3.58	**13**	19.54	19.44	**96**	3.67	3.59	**84**	4.76	3.54	1220
FFT	10.79	10.60	**194**	253.63	250.19	**3436**	15.13	10.75	**4379**	12.94	10.15	2790

SOD has the smallest migration overhead : *ranges* ***from 13ms to 194ms*** *under Gigabit Ethernet*

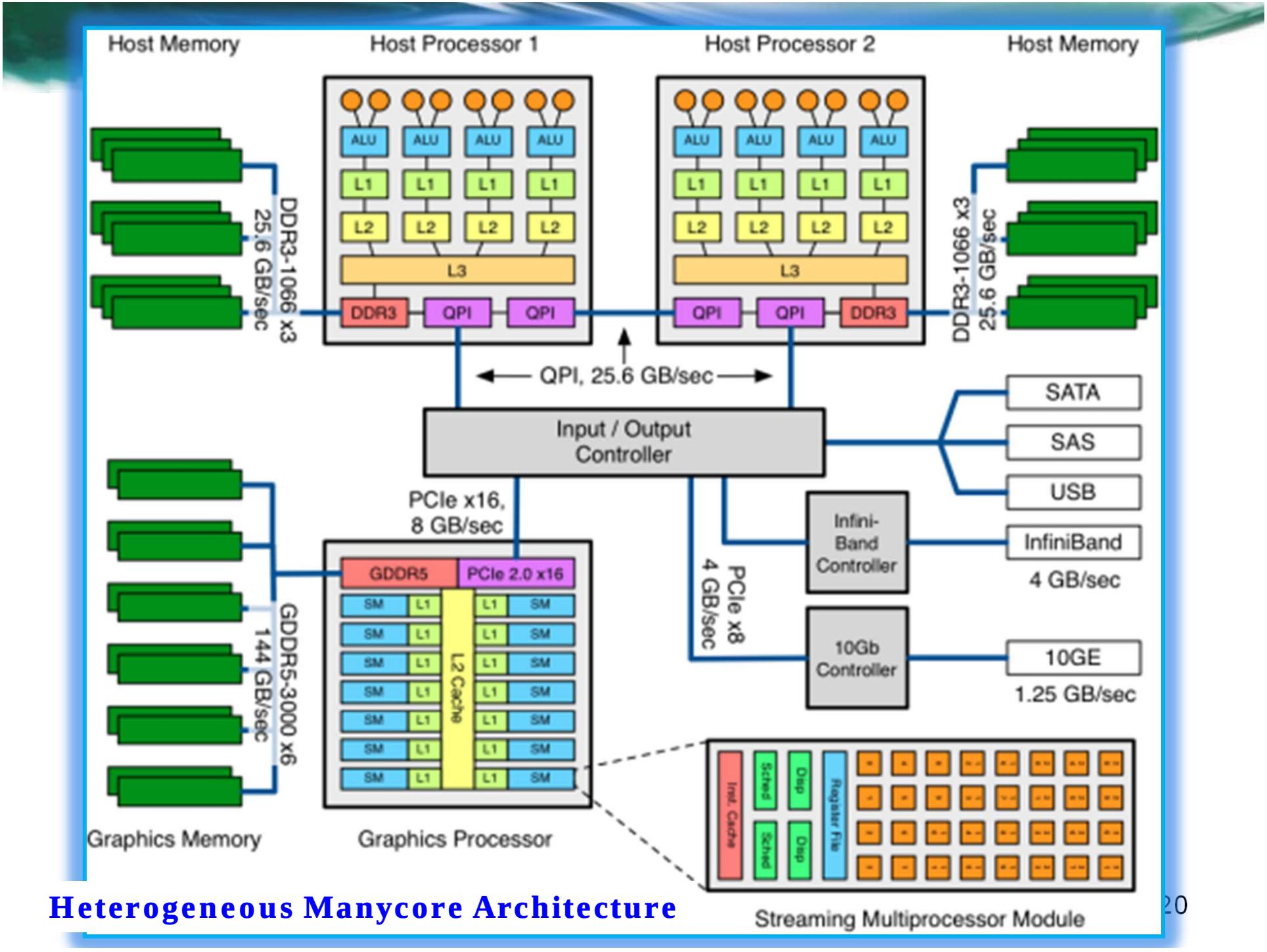
Frame (SOD): Thread : Process : VM = 1 : 3 : 10 : 150

"A Stack-On-Demand Execution Model for Elastic Computing", **IEEE ICPP2010**, San Diego, California, USA, September 13-16, 2010.

Part II

Heterogeneous Manycore Computing (CPUs+ GUPs)

JAPONICA : Java with Auto-Parallelization ON GraphIcs Coprocessing Architecture



Heterogeneous Manycore Architecture

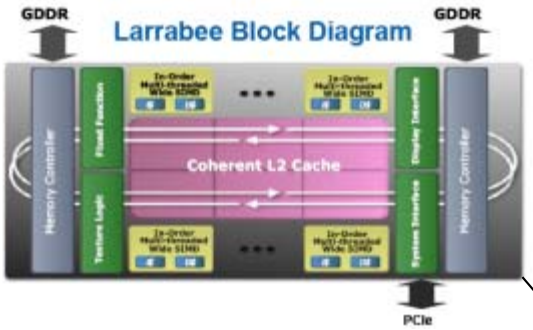
A Variety of Coprocessors

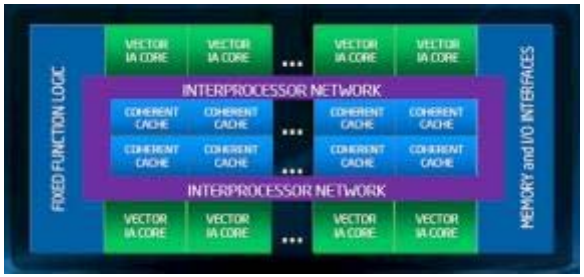
Vendor	Model	Launch Date	Fab. (nm)	#Accelerator Cores (Max.)	GPU Clock (MHz)	TDP (watts)	Memory	Bandwidth (GB/s)	Programming Model	Remarks
Intel	Sandy Bridge	2011Q1	32	12 HD graphics 3000 EUs (8 threads/EU)	850 – 1350	95	L3: 8MB Sys mem (DDR3)	21	OpenCL	Bandwidth is system DDR3 memory bandwidth
	Ivy Bridge	2012Q2	22	16 HD graphics 4000 EUs (8 threads/EU)	650 – 1150	77	L3: 8MB Sys mem (DDR3)	25.6		
	Xeon Phi	2012H2	22	57 x86 cores (with a 512-bit vector unit)	600-1100	300	8GB GDDR5	300	OpenMP#, OpenCL*, OpenACC%	Less sensitive to branch divergent workloads
AMD	Brazos 2.0	2012Q2	40	80 Evergreen shader cores	488-680	18	L2: 1MB Sys mem (DDR3)	21	OpenCL, C++AMP	
	Trinity	2012Q2	32	128-384 Northern Islands cores	723-800	17-100	L2: 4MB Sys mem (DDR3)	25		APU
Nvidia	Fermi	2010Q1	40	512 Cuda cores (16 SMs)	1300	238	L1: 48KB L2: 768KB 6GB	148	CUDA, OpenCL, OpenACC	
	Kepler	2012Q2	28	1536 Cuda cores	1000	300	8GB GDDR5	320		3X Perf/Watt, Dynamic Parallelism, HyperQ

Intel-specific OpenMP
* Not yet officially confirmed

Intel Many Integrated Core Architecture (MIC)

Larrabee (2006-2010)



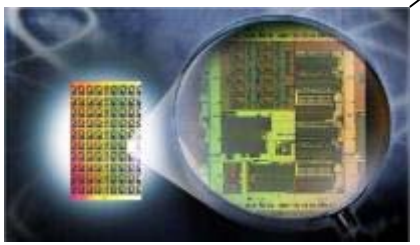


ring interconnect keeps the caches for each chip coherent

22nm Knights Corner (2012) 50+ cores,

Single-chip Cloud Computer (2009-) 48 cores

Knights Ferry (2010)
32 cores, 1.2 GHz, 750 GFLOPS, 2 GB GDDR5, ~300 W

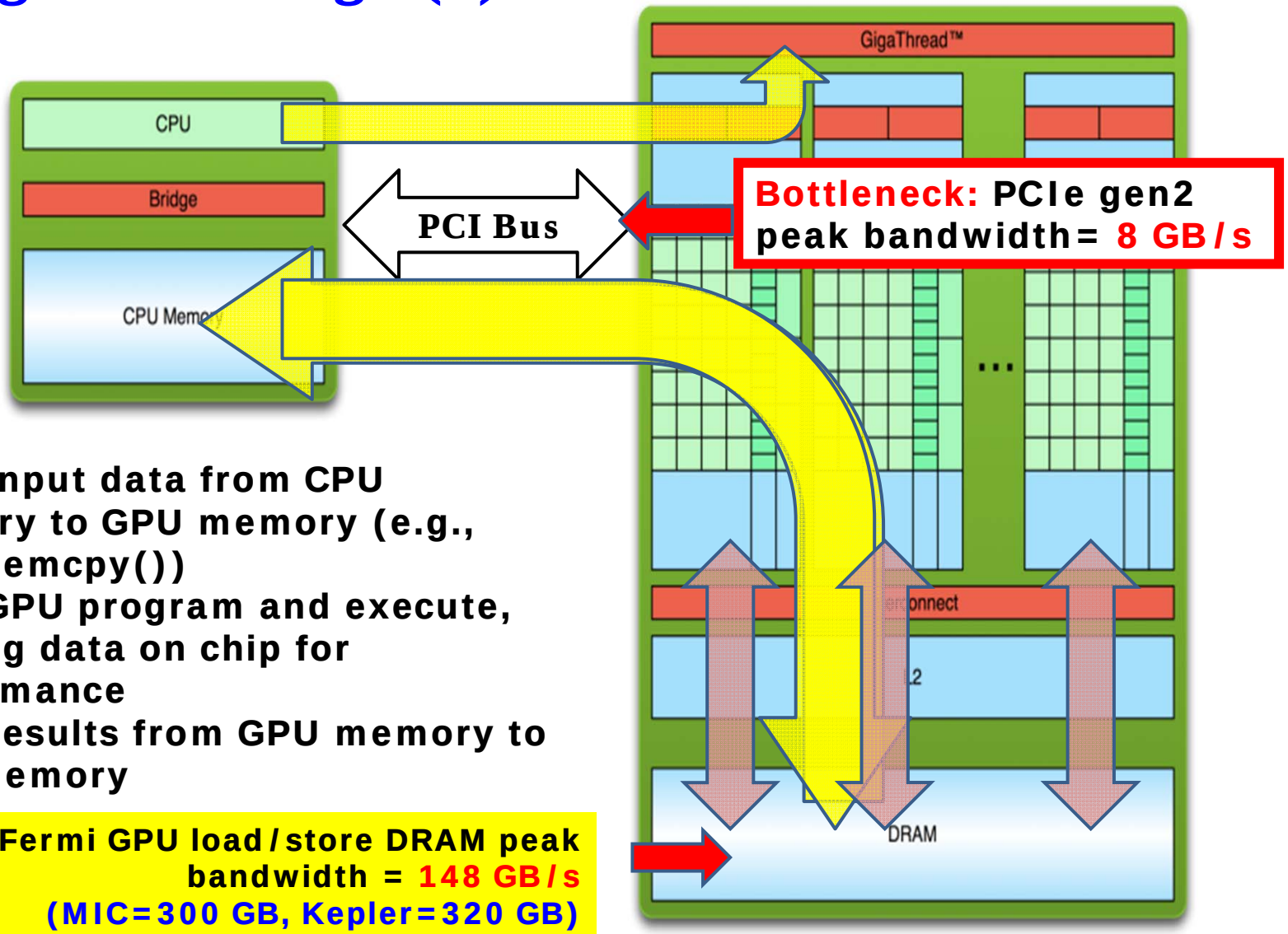


Teraflops Research Chip (2007)
***80 cores**, 3.16GHz, 1.01 Tflops, 62W*

The MIC chip has a superscalar x64 core (without **the out-of-order execution** of Xeons) and a **512-bit vector** math unit that can do **16 floating point** operations per clock with single precision math.

'Knight's Corner' chips (branded as 'Xeon Phi')- 6/2012 -- 64 x86 cores (256 threads) + a 512-bit vector unit @2GHz, **1 Teraflops**

Design Challenge (1)



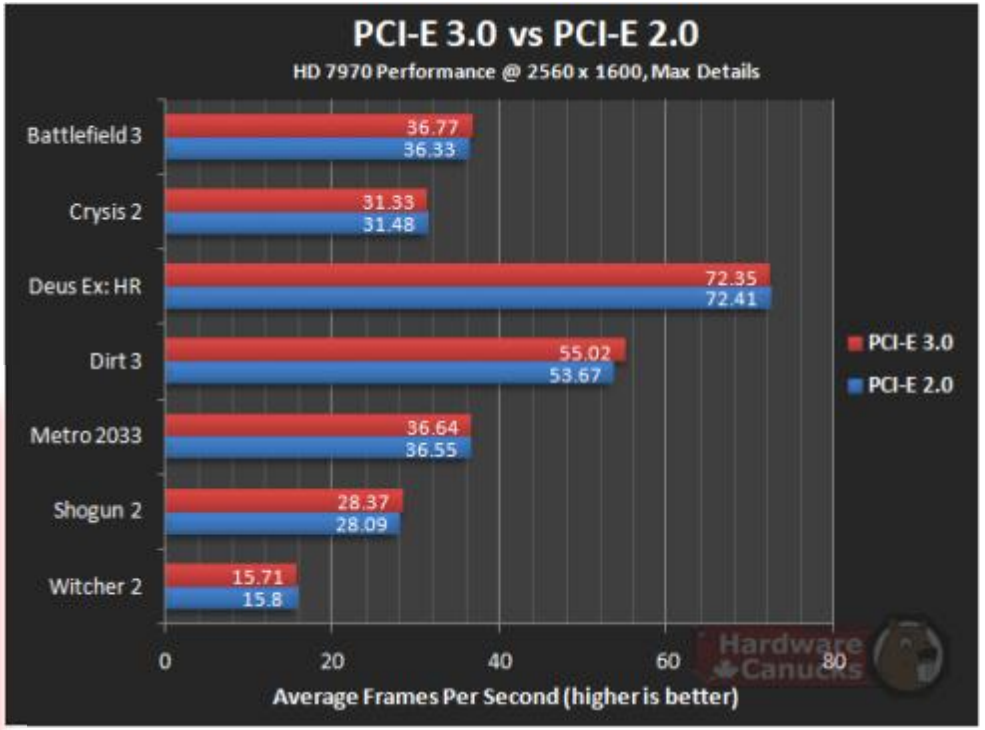
1. Copy input data from CPU memory to GPU memory (e.g., cudaMemcpy())
2. Load GPU program and execute, caching data on chip for performance
3. Copy results from GPU memory to CPU memory

Fermi GPU load/store DRAM peak bandwidth = 148 GB/s
(MIC=300 GB, Kepler=320 GB)

Does PCIe 3.0 help?

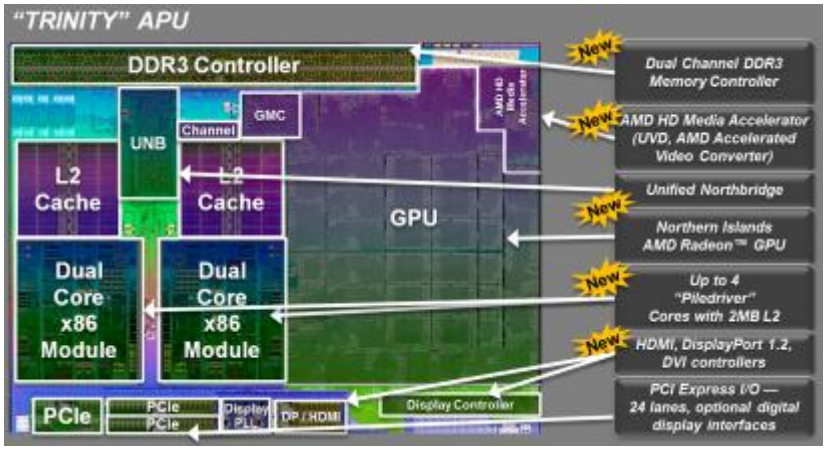
PCIe Bandwidth Comparison (Each Direction)			
	PCIe 1.x	PCIe 2.x	PCIe 3.0
x1	250MB/sec	500MB/sec	1GB/sec
x2	500MB/sec	1GB/sec	2GB/sec
x4	1GB/sec	2GB/sec	4GB/sec
x8	2GB/sec	4GB/sec	8GB/sec
x16	4GB/sec	8GB/sec	16GB/sec

Informal testing results: No appreciable difference in performance between PCIe 3 x16 (16GB/sec) and PCIe 2 (8GB/sec)

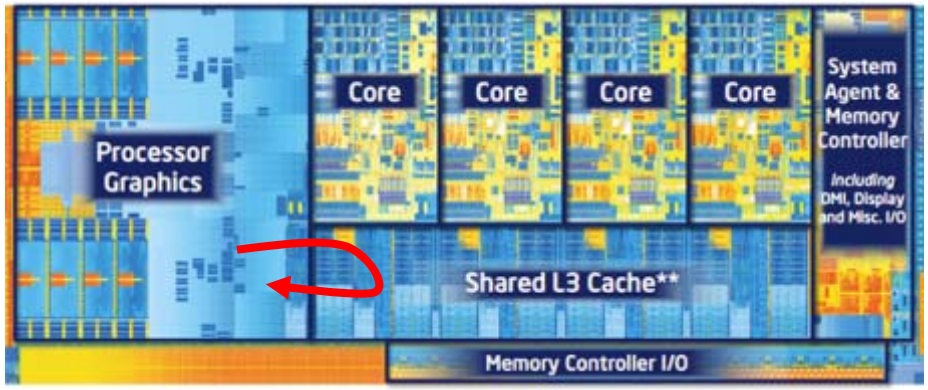


Require much higher "flops per byte" – i.e., applications with "High Arithmetic Intensity" (HAI)

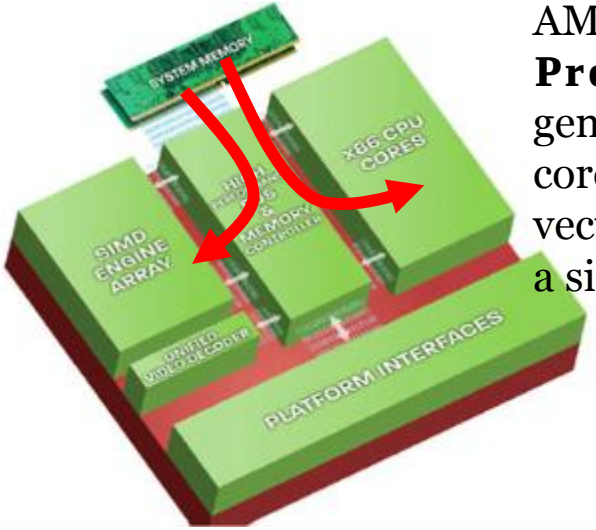
Soluion? : CPU-GPU mashups



AMD "Trinity".



Intel Ivy Bridge (22nm)



AMD's new **Accelerated Processing Units** combine general-purpose x86 CPU cores with programmable vector processing engines on a single silicon die

Ivy Bridge GPU incorporates a high bandwidth **L3 cache** that is shared by the entire shader array.

Design Challenge (2): GPU Can't Handle Dynamic Loops

GPU = SIMD / Vector

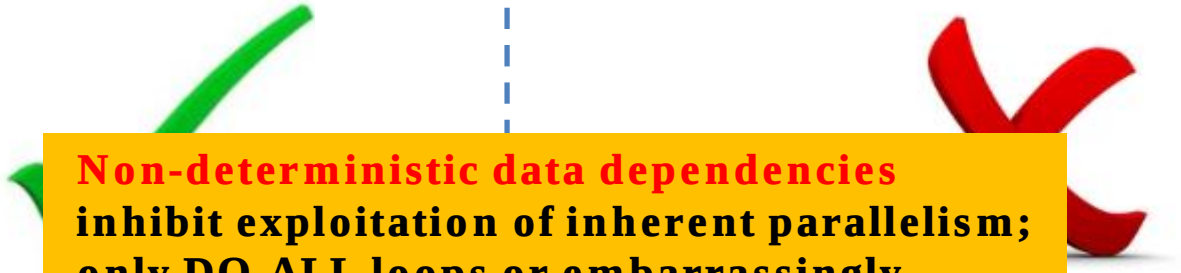
Data Dependency Issues (RAW, WAW) → Solutions?

Static loops

```
for(i=0;  i<N; i++)
{
      C[i] = A[i] + B[i];
}
```

Dynamic loops

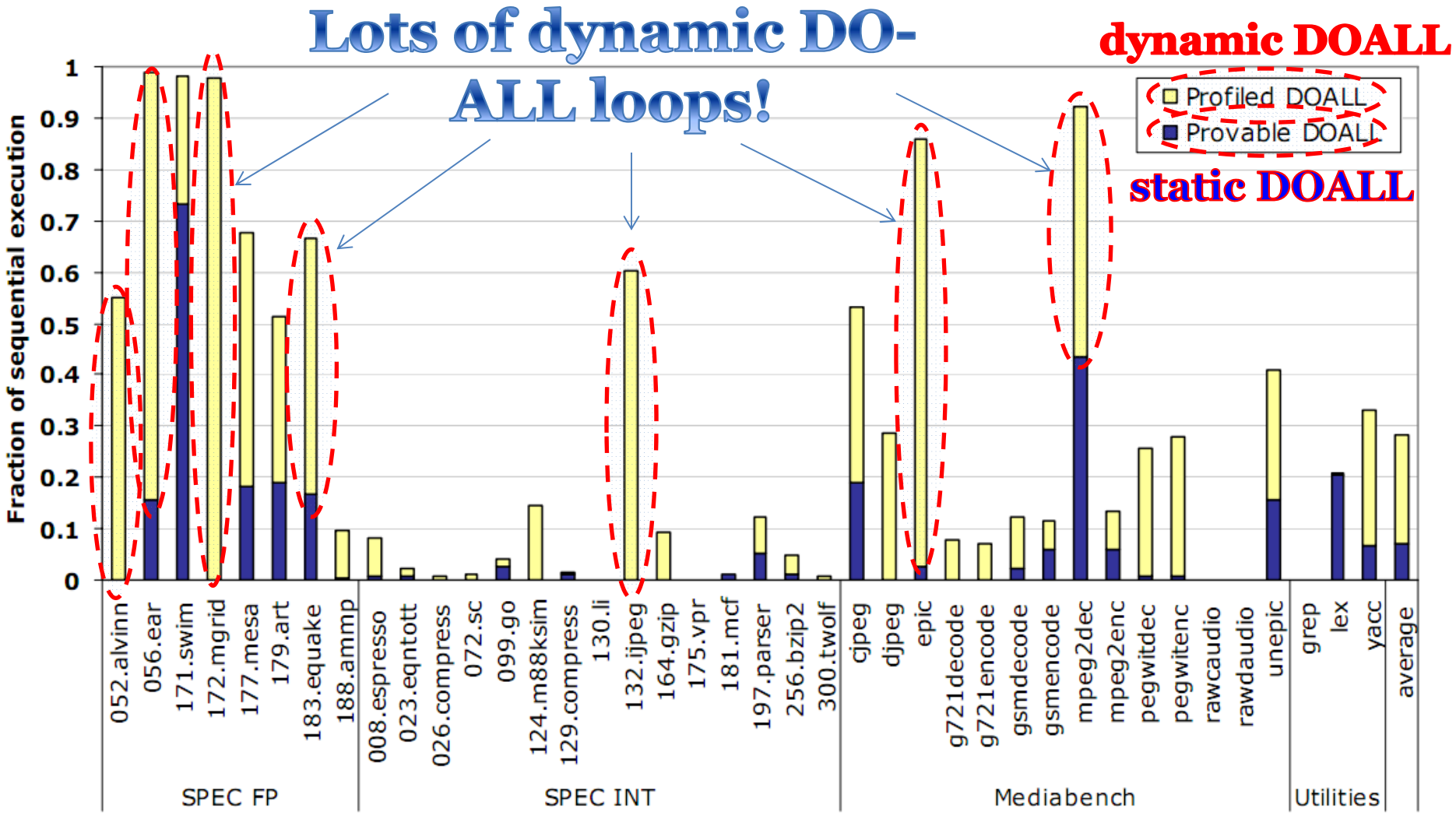
```
for(i=0;  i<N; i++)
{
      A[ w[i] ] = 3 * A[ r[i] ];
}
```



Non-deterministic data dependencies inhibit exploitation of inherent parallelism; only DO-ALL loops or embarrassingly parallel workload gets admitted to GPUs.

26

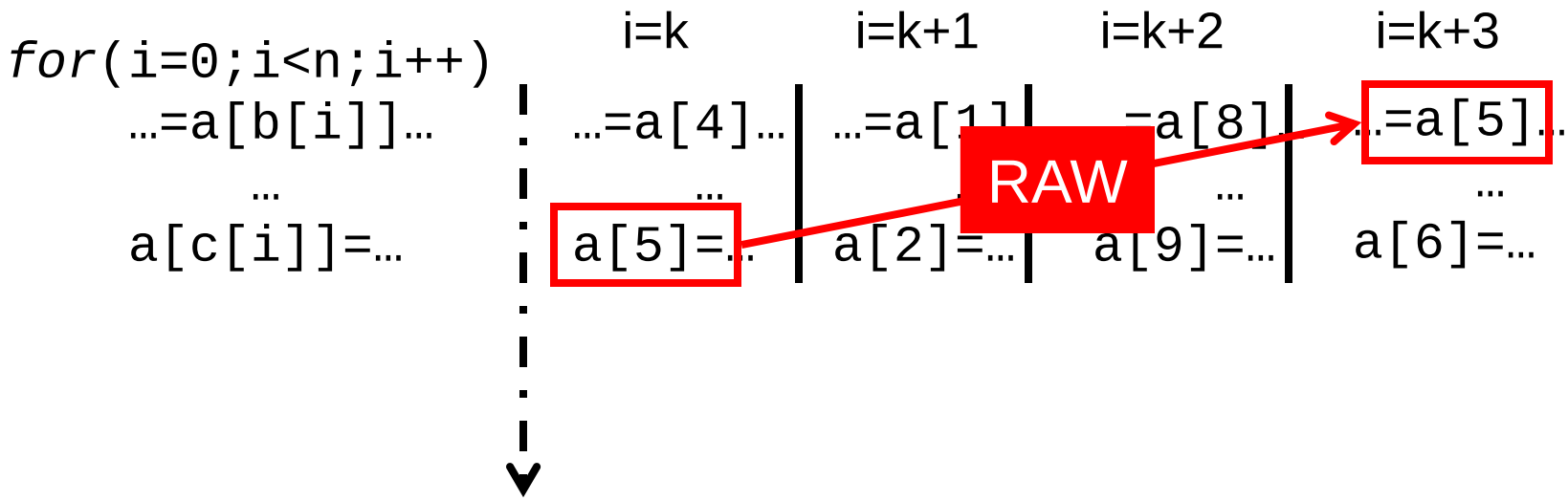
Dynamic loops are common in scientific and engineering applications



Source: Z. Shen, Z. Li, and P. Yew, "An Empirical Study on Array Subscripts and Data Dependencies"

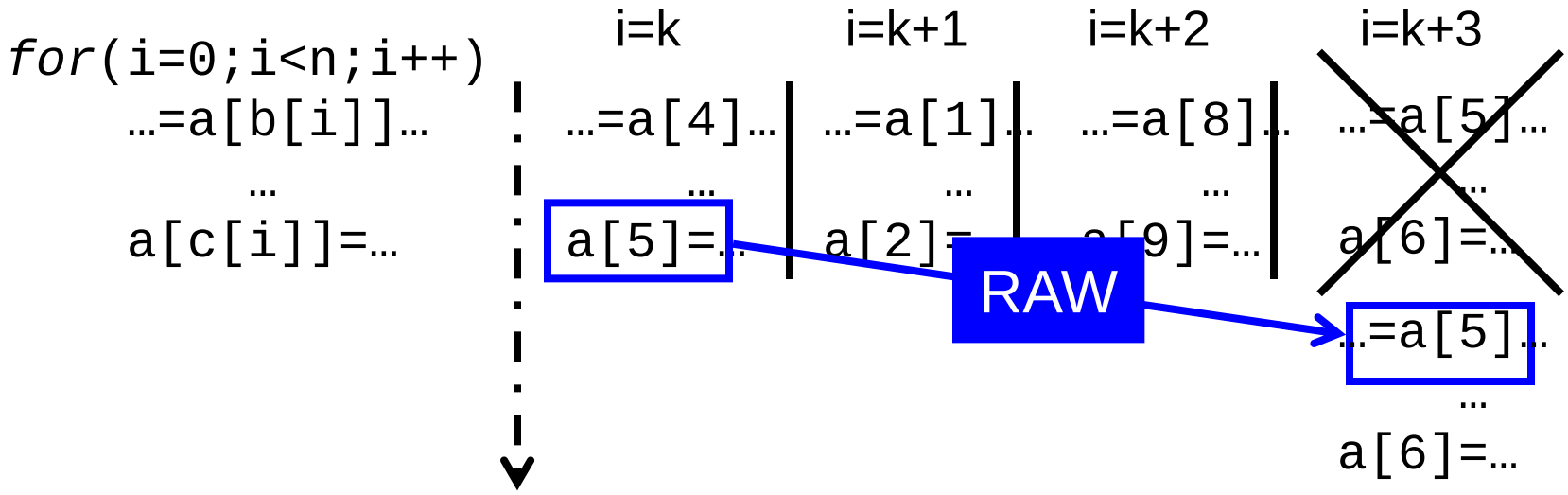
Thread Level Speculation (*TLS*)

- Execute hard-to-analyze codes speculatively (or optimistically) in parallel.
  - Assume no dependences and execute in parallel
  - Track memory accesses and detect violations
  - Squash and restart offending threads



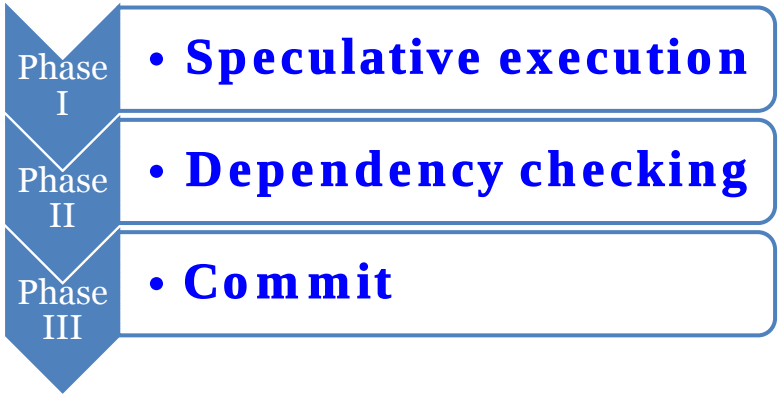
Thread Level Speculation (*TLS*)

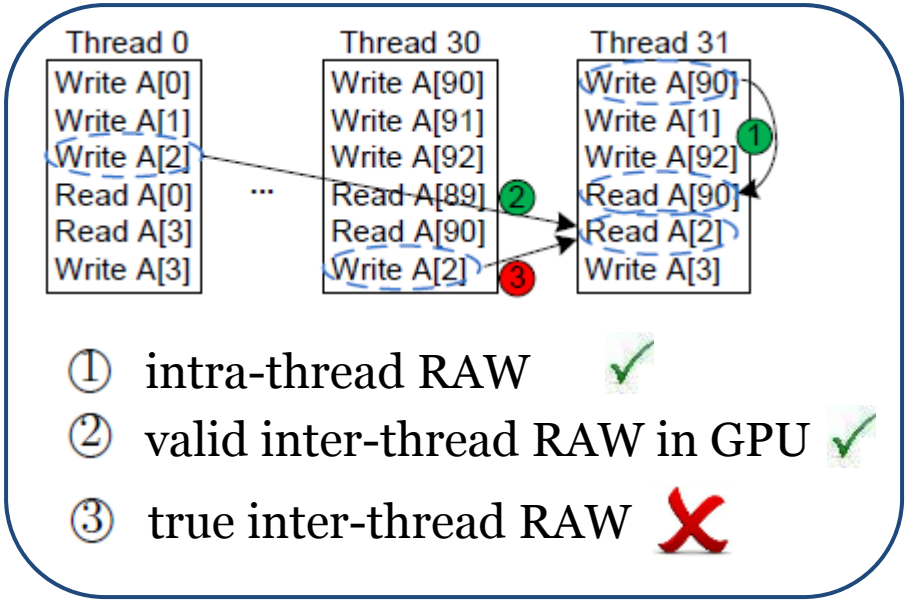
- Execute hard-to-analyze codes speculatively (or optimistically) in parallel.
  - Assume no dependences and execute in parallel
  - Track memory accesses and detect violations
  - Squash and restart offending threads



GPU-TLS : Thread-level Speculation on GPU

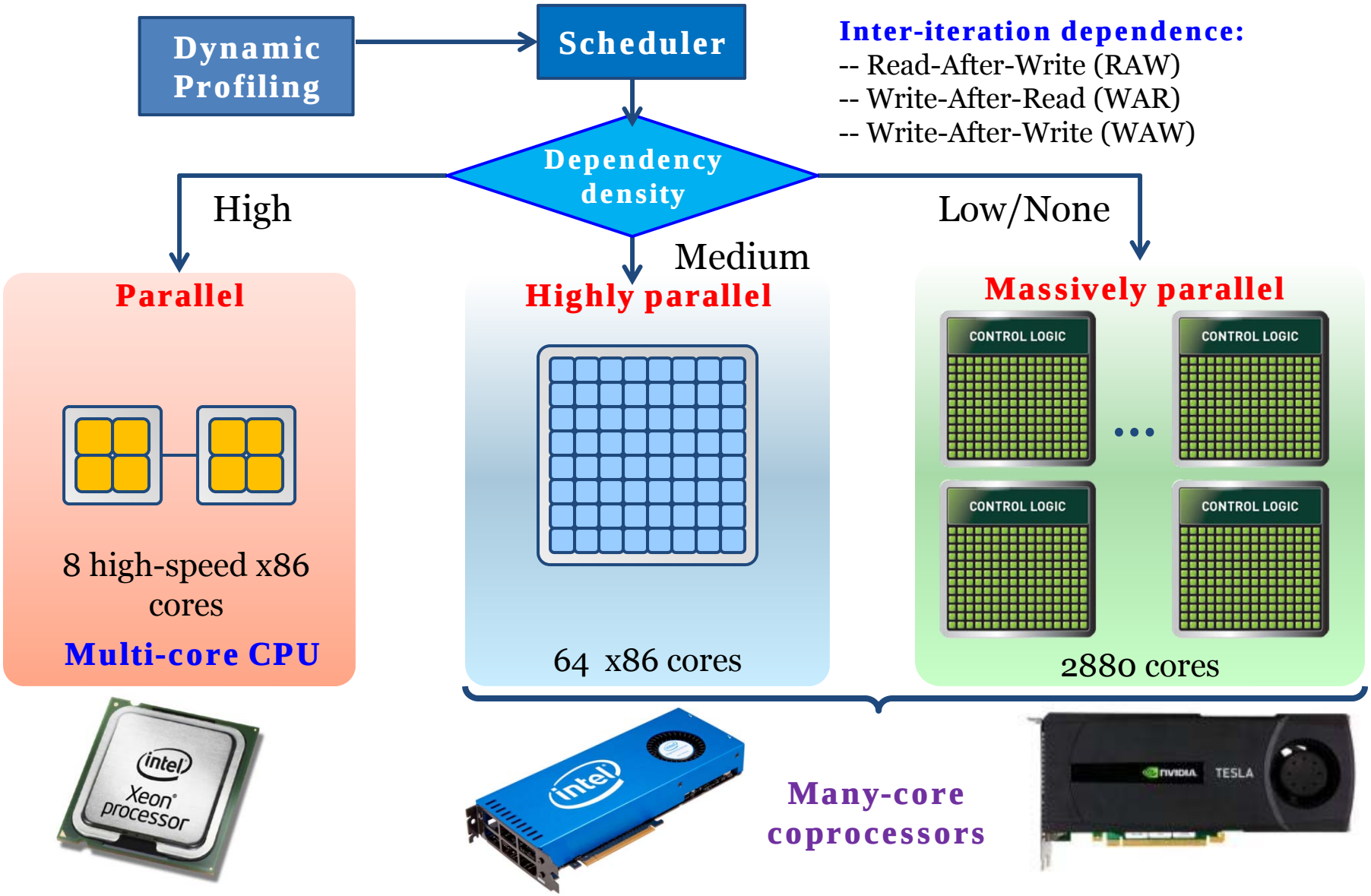
- **Incremental parallelization**
  - sliding window style execution.
- **Efficient dependency checking schemes**
- **Deferred update**
  - Speculative updates are stored in the write buffer of each thread until the commit time.
- **3 phases of execution**



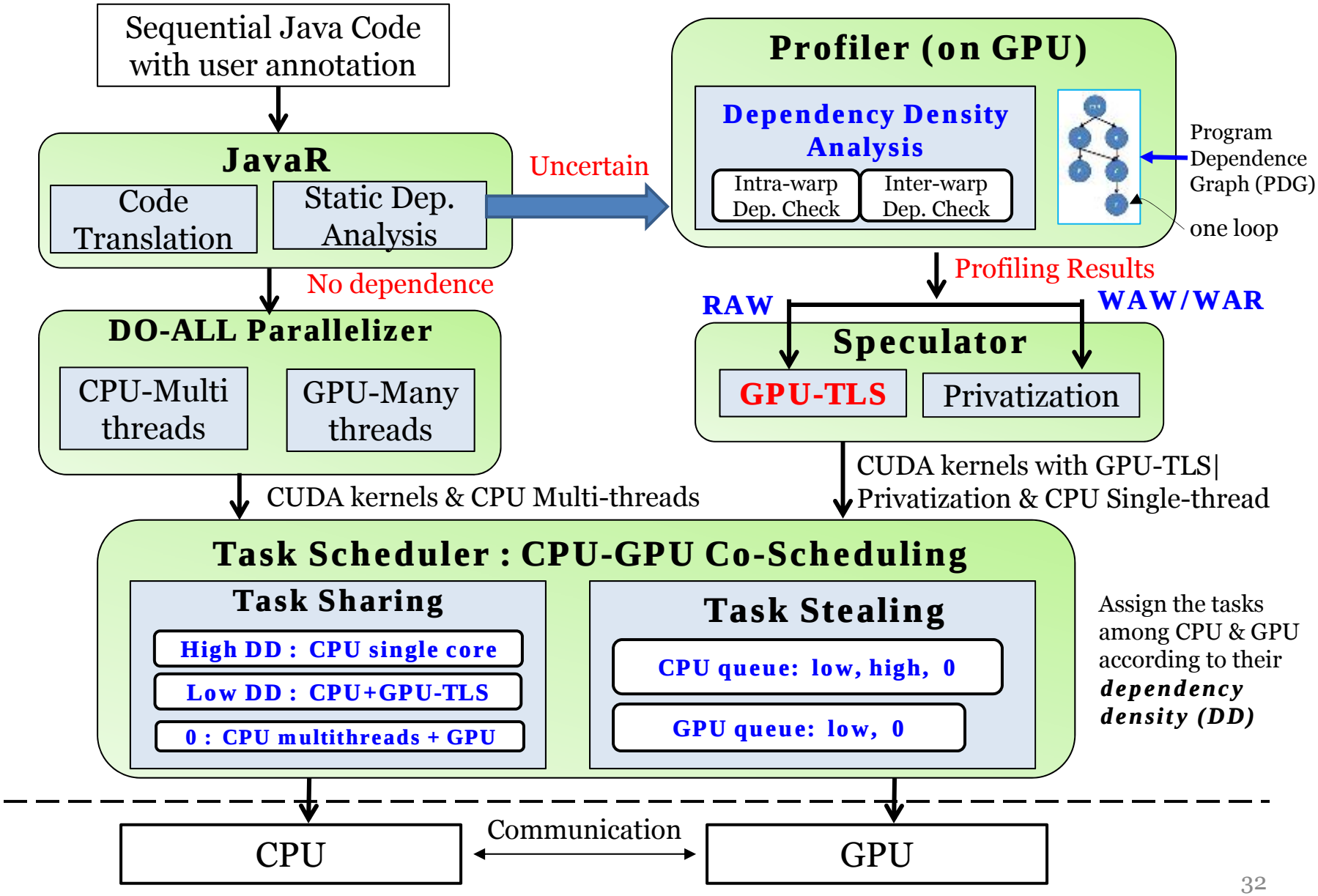


GPU: **lock-step execution** in the same warp (32 threads per warp).

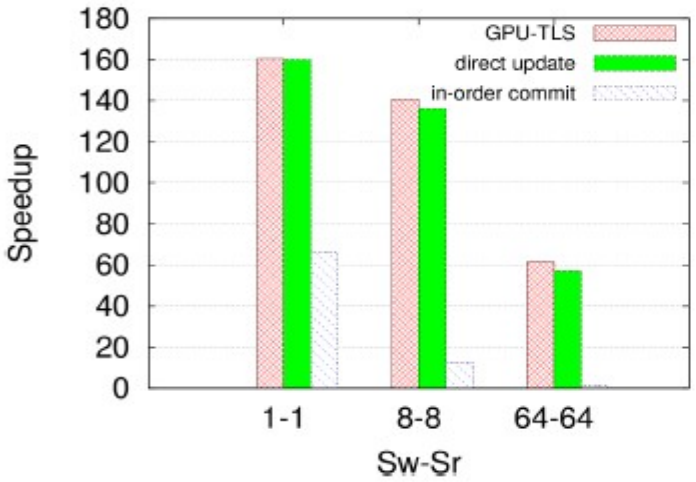
JAPONICA : Profile-Guided Work Dispatching



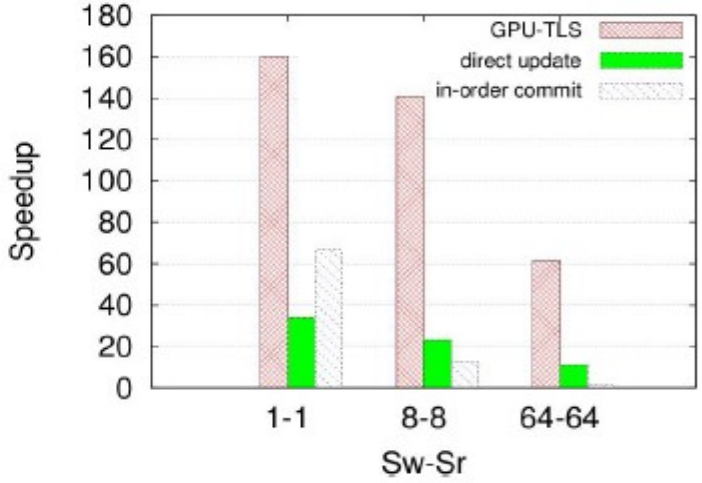
JAPONICA : System Architecture



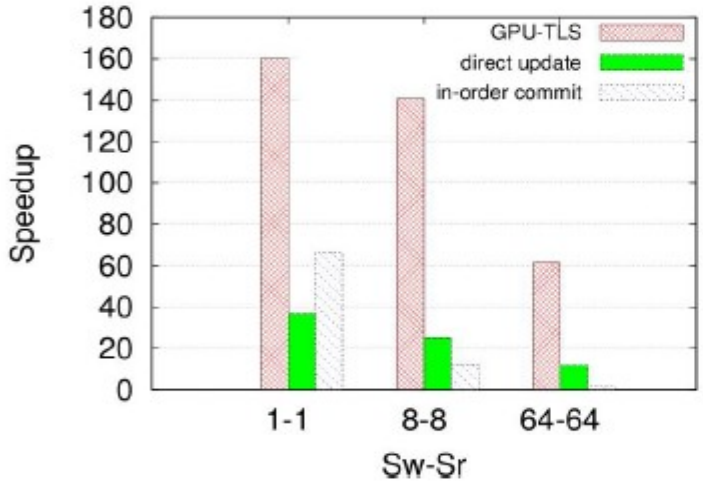
GPU-TLS: Performance Evaluation



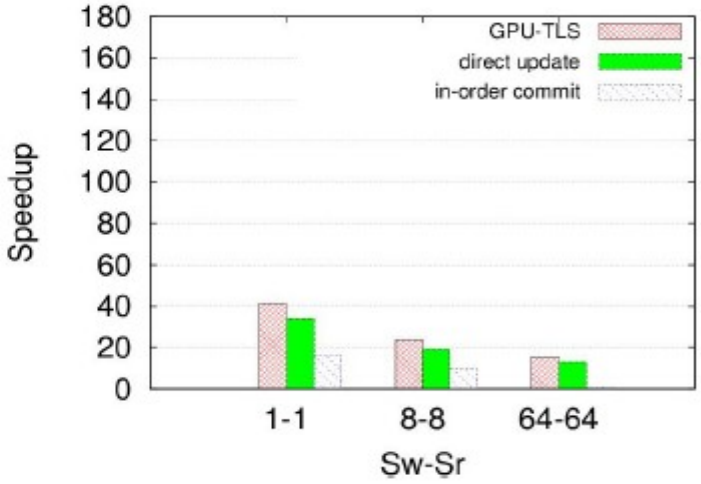
(a) Speedups for DOALL loops



(b) Speedups for WAW loops



(c) Speedups for WAR loops

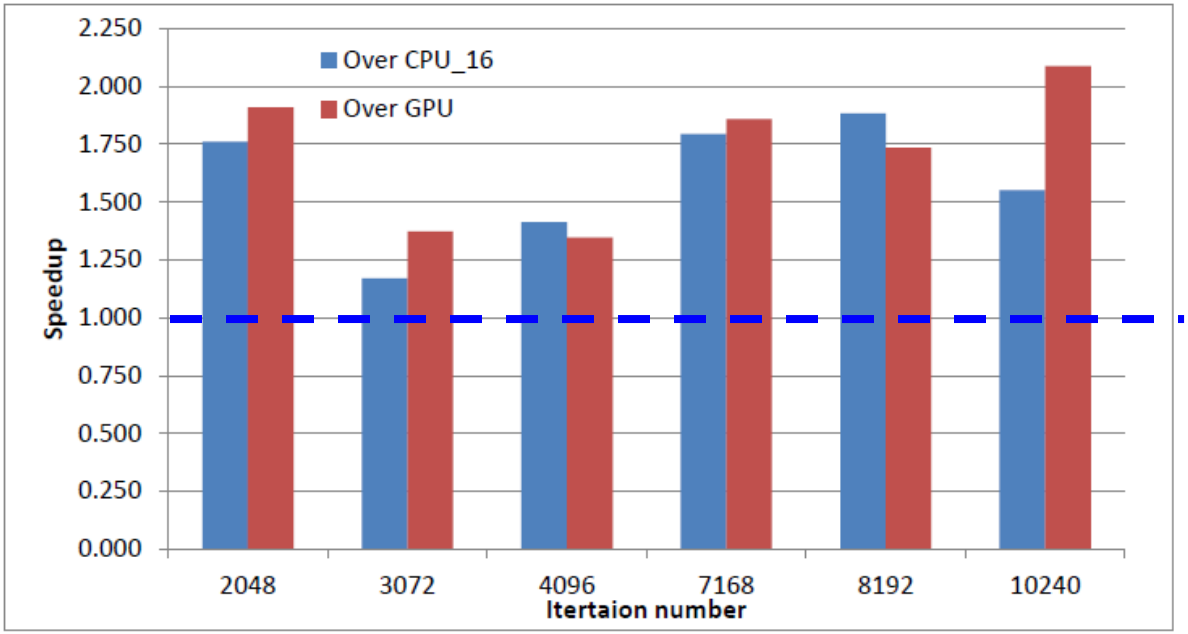


(d) Speedups for RAW loops

33

JAPONICA Evaluation: Bi-Conjugate Gradient (BICG)**

Number of iterations	CPU	CPU_16	GPU	CPU+GPU	Workload of CPU
2048	44.792	5.091	5.521	2.891	50%
3072	100.016	12.269	14.391	10.482	50%
4096	179.699	19.791	18.856	14.004	50%
7168	544.005	50.841	52.668	28.332	50%
8192	718.596	65.32	60.205	34.691	50%
10240	1109.6	100.871	135.728	65.044	62.5%



**from the Polybench

General Observations and Prediction

- **Lowering clock rate** but many more cores.
  - Kepler 1 Ghz (3072) vs Fermi 1.3 Ghz (512)
- **More power efficient** (increasing perf/watt)
- **Increasing bandwidth** (> 300 GB/s, e.g., Kepler)
  - getting readier for data intensive workloads.
- **More dynamic workflow**:
  - Kepler's Dynamic Parallelism : GPU kernel can spawn new work onto the GPU
- **Intel MIC**, using x86 cores, is stealing the limelight.
  - We foresee it will be a norm in the coprocessor world.
  - Deliver similar flops (1 Tflops) but easier programming

Part III

Big Data Future 1000-core "General Purpose" Maycore Chips

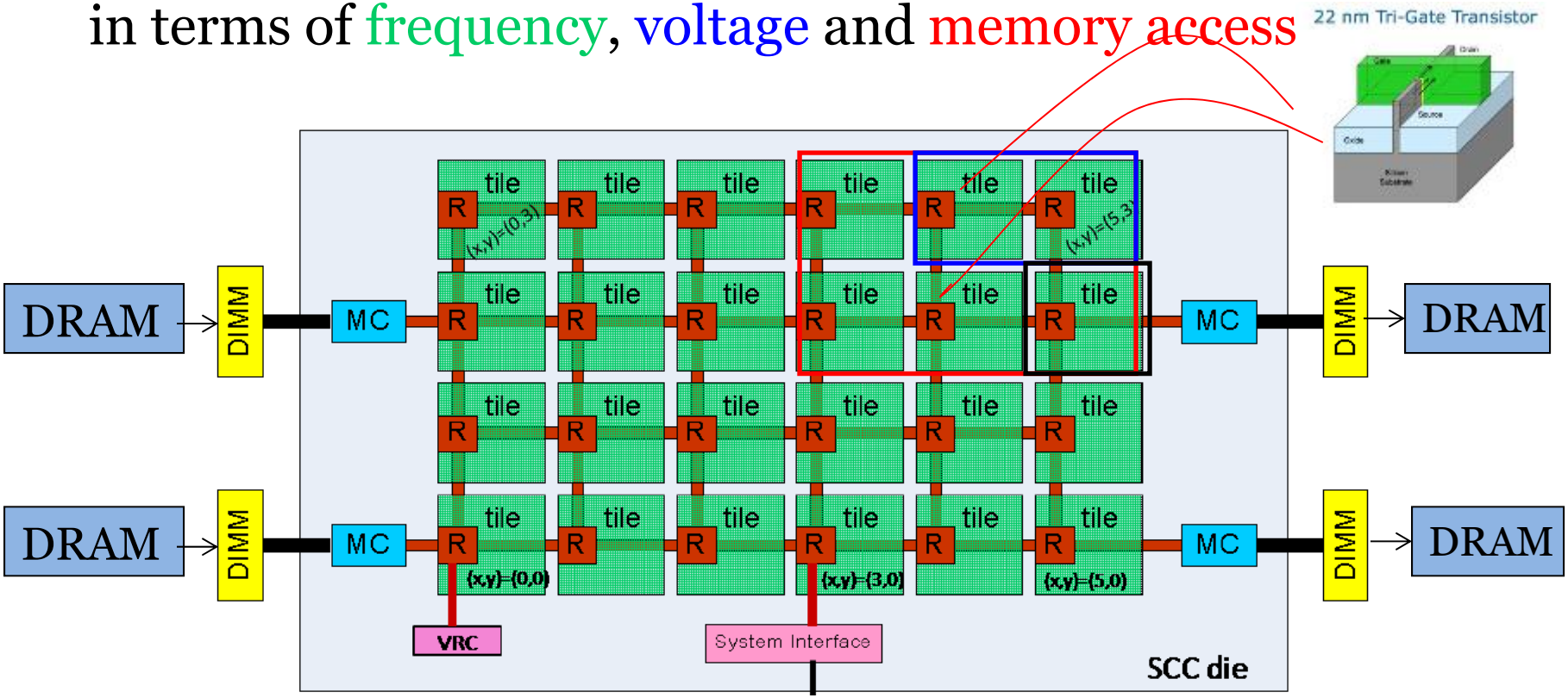
"General Purpose" Manycore

Micro-architecture	# of cores	On-Chip Network (Link Bandwidth)	H/W Coherence	L1$/core	L2$/core	L3$	DDR Controller
Teraflops Research Chip	80 (4.0 GHz)	2D Mesh (256Gb/s)	No	5KB	256KB	NA	3D stacked memory
MIT's ATAC (2008)	1000 (simulation)	2D (optical) Mesh (32Gb/s)	Yes	NA	NA	NA	NA
Single-Chip Cloud (2009)	48 (1.0 GHz)	2D Mesh (512Gb/s)	No	32KB	256KB + 8KB MPB	Nil	4
Tilera Tile-GX (2009)	100 (1.5 GHz)	2D Mesh (320Gb/s)	Yes	64KB	256KB	26MB (shared)	4
Godson-T (FPGA, 2011)	64 (1.0 GHz)	2D Mesh	Yes	32KB	128KB x 16 shared	Nil	4

Tile-based architecture: Cores are connected through a 2D network-on-a-chip

Tiled Manycore Architectures

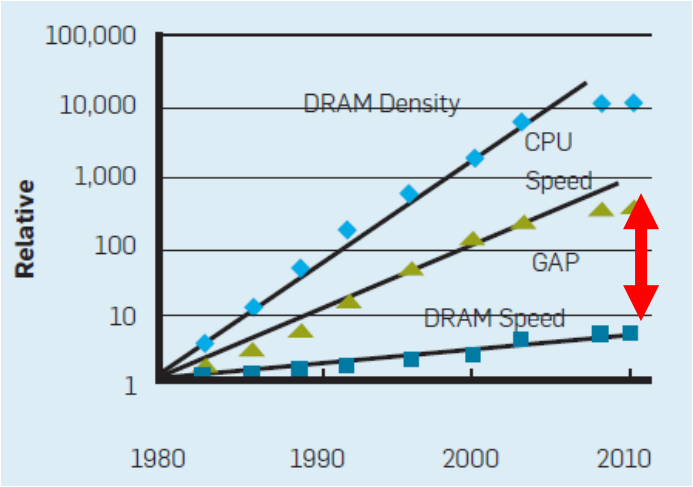
- The cores of the SCC are grouped into multiple domains in terms of frequency, voltage and memory access



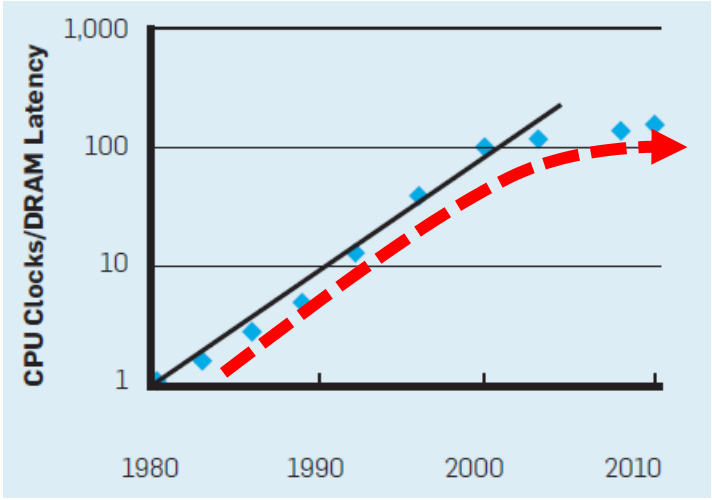
Multiple cores per tile, connected by an on-die 2D mesh network (network-on-chip).

Design Challenge (1): "Off-chip Memory Wall" Problem

– DRAM performance (latency) improved slowly over the past 40 years.



(a) Gap of DRAM Density & Speed



(b) DRAM Latency Not Improved

Memory density has doubled nearly every two years, while performance has improved slowly (e.g. still 100+ of core clock cycles per memory access)

Design Challenge (2): "Coherency Wall" Problem

Overhead of enforcing cache coherency across 1,000 cores at hardware level will put a hard limit on scalability

1. Performance overhead: Coherence uses **20%** more traffic per miss than a system with caches but not coherence
2. Die space overhead: cache directory, read/write log increase
3. Not always needed: Only around **10%** of the application memory references actually require cache coherence tracking
4. Verification complexity and extensibility: require dealing with subtle races and many transient states

Intel's SCC and Teraflops Research Chip decided to give up coherent caches.

Laser-Powered Chip in 2017??

HP Corona : 10-Teraflop Manycore Chip (expected 2017)

- 256 cores, each supporting up to four threads
- **Optical interconnect** : a **20 TB/sec** DWDM crossbar
- **Optically connected memory (OCM) @ 10 TB/sec**
  - **80 GB/sec** : 8-core Intel E5-2600 Xeons
  - **64 GB/sec** : SPARC64 VIIIfx CPU of K computer
  - **177 GB/sec** : NVIDIA M2090,
- **Energy efficiency**: 6.4 watts @ 10 GB/sec of data to DRAM, which is 25 x less than electrical interconnect (160 watts)
- **MOESI directory cache coherency protocol**
- Aim at big data applications
- **Other projects: Intel's Runnemede**, **MIT's Angstrom**, **NVIDIA's Echelon**, and **Sandia's X-calibur.**

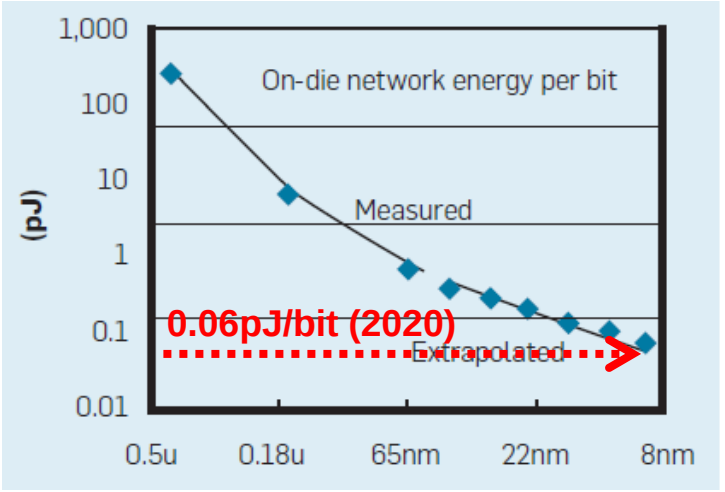
Design Challenge (3): "Power Wall" Problem

- **Computation costs much less energy than moving data to and from the computation units**
- **As the energy cost of computation is reduced by voltage scaling, the cost of data movement starts to dominate.**

If only 10% of the operands move over the network, 10 hops in average, at 0.06pJ/bit, the network would consume 35 watts of power, > 50% of the power budget of the processor.

- **Bill Dally, Chief Scientist of nVIDIA**
  - 1 pJ for an integer operation
  - 20 pJ for a floating-point operation
  - 26 pJ to move an operand over 1mm of wire to local memory (1000x)
  - 1 nJ to read an operand from on-chip memory located at the far end of a chip (1600x)
  - 16 nJ to read an operand from off-chip DRAM

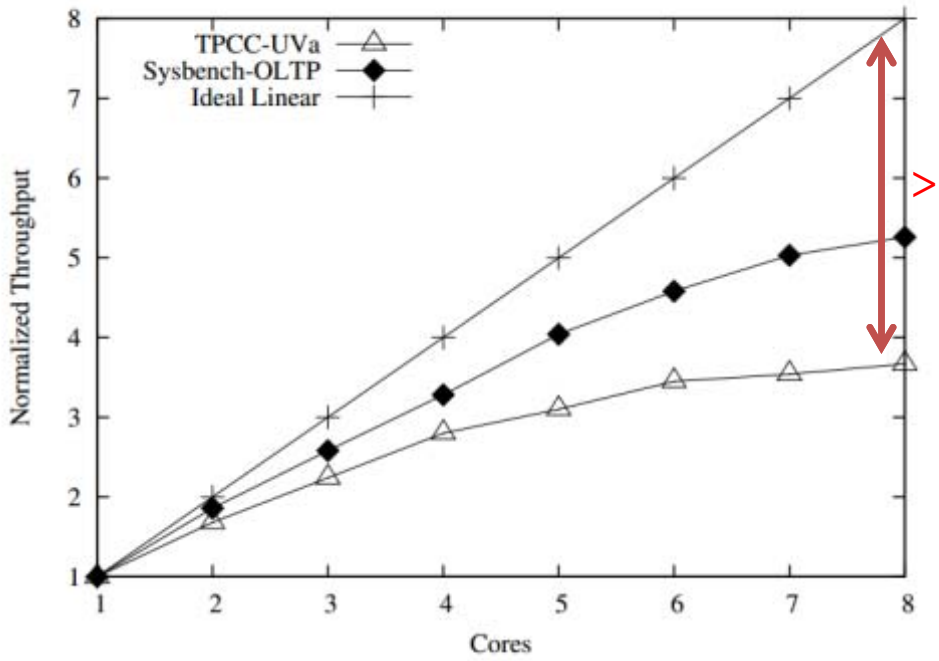
You cannot break the laws of physics - and 7nm is the limit



On-die network energy consumption per bit

picojoule (pJ) = 10^{-12} J
nanojoule (nJ) = 10^{-9} J

Design Challenge (4): OS Scalability



Intel 8 core system
4G RAM, 500G HDD
Linux kernel 2.6.25

>2X Gap

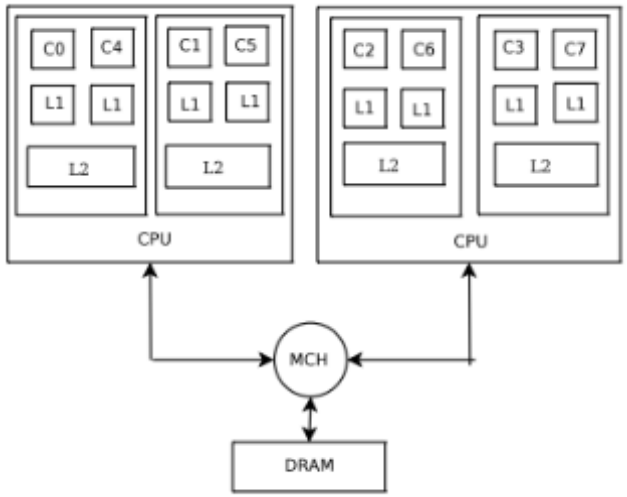


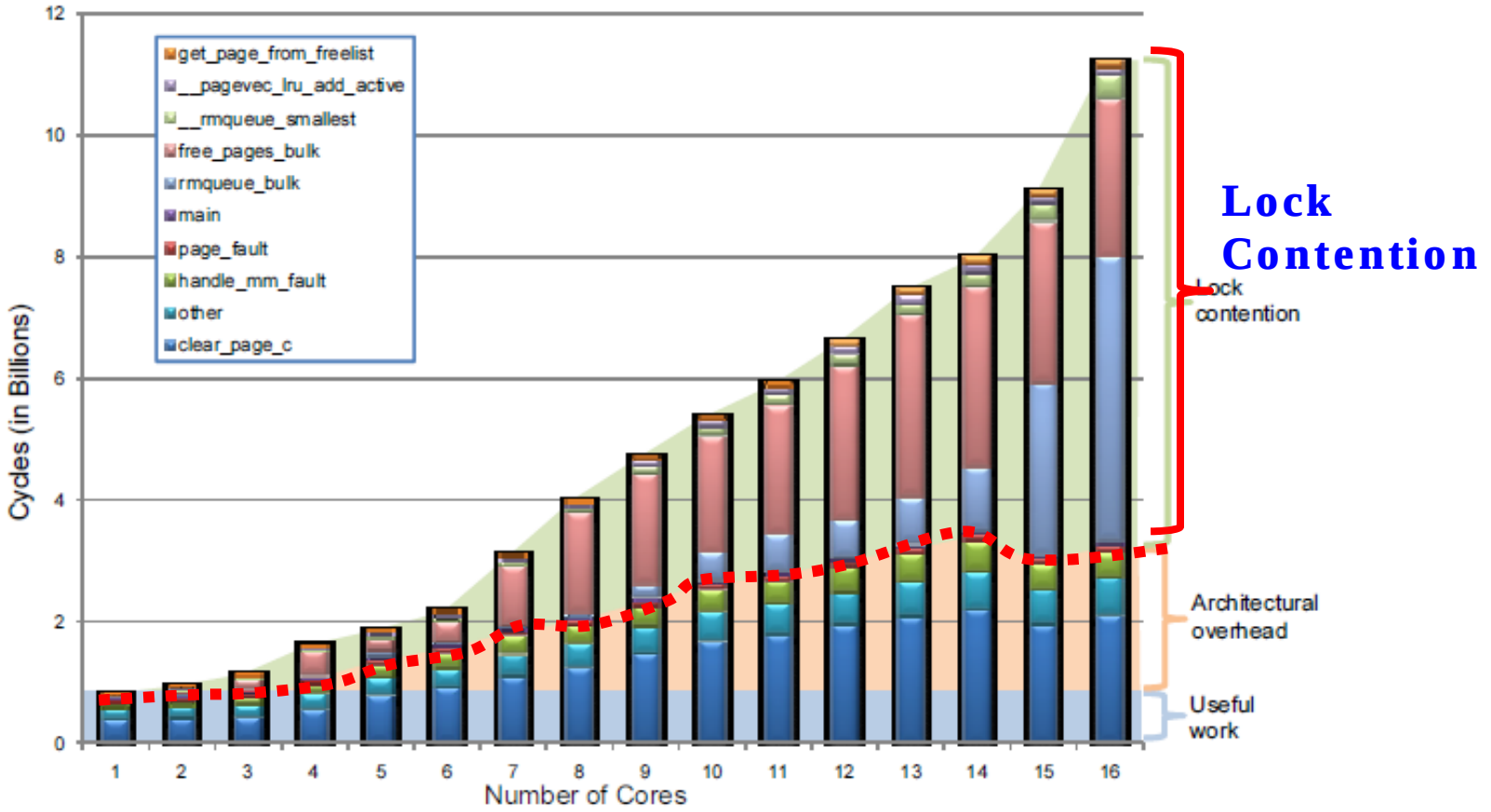
Fig. 4. Normalized transaction throughput scalability of Sysbench-OLTP and TPCC-UVa with the number of cores.

On **L**ine **T**ransaction **P**rocessing

Y. Cui, *et al*, Scaling OLTP Applications on Commodity Multi-Core Platforms, ISPASS10

Lock Contention in Multicore System

- **Physical memory allocation performance sorted by function. As more cores are added more processing time is spent contending for locks.**



Linux Atomic Operations

- x86 LOCK prefix makes many read-modify-write instructions atomic.
- Most general instruction is cmpxchg, used to implement locks

```
29 static __inline__ void atomic_add(int i, atomic_t *v)
30 {
31         __asm__ __volatile__(
32                 LOCK "addl %1,%0"
33                 :"=m" (v->counter)
34                 :"ir" (i), "m" (v->counter));
35 }
```

```
37 static __inline__ void atomic_sub(int i, atomic_t *v)
38 {
39         __asm__ __volatile__(
40                 LOCK "subl %1,%0"
41                 :"=m" (v->counter)
42                 :"ir" (i), "m" (v->counter));
43 }
```

x86: LOCK prefix "locks" the memory bus for the destination memory address to allow the processor has exclusive use of any shared data.

How often is ‘cmpxchg’ used in Linux kernel?

$ cat vmlinux.asm | grep cmpxchg

```
c01046de:       f0 0f b1 15 3c 99 30    lock cmpxchg %edx,0xc030993c
c0105591:       f0 0f b1 15 3c 99 30    lock cmpxchg %edx,0xc030993c
c01055d9:       f0 0f b1 15 3c 99 30    lock cmpxchg %edx,0xc030993c
c010b895:       f0 0f b1 11             lock cmpxchg %edx,(%ecx)
c010b949:       f0 0f b1 0b             lock cmpxchg %ecx,(
c0129a9f:       f0 0f b1 0b             lock cmpxchg %ecx,(
c0129acf:       f0 0f b1 0b             lock cmpxchg %ecx,(
c012d377:       f0 0f b1 0e             lock cmpxchg %ecx,(
c012d41a:       f0 0f b1 0e             lock cmpxchg %ecx,(
c012d968:       f0 0f b1 16             lock cmpxchg %edx,(
c012e568:       f0 0f b1 2e             lock cmpxchg %ebp,(
c012e57a:       f0 0f b1 2e             lock cmpxchg %ebp,(
c012e58a:       f0 0f b1 2e             lock cmpxchg %ebp,(
c012e83f:       f0 0f b1 13             lock cmpxchg %edx,(
c012e931:       f0 0f b1 0a             lock cmpxchg %ecx,(%edx)
c012ea94:       f0 0f b1 11             lock cmpxchg %edx,(%ecx)
c012ecf4:       f0 0f b1 13             lock cmpxchg %edx,(%ebx)
c012f08e:       f0 0f b1 4b 18          lock cmpxchg %ecx,0x18(%ebx)
c012f163:       f0 0f b1 11             lock cmpxchg %edx,(%ecx)
c013cb60:       f0 0f b1 0e             lock cmpxchg %ecx,(%esi)
c0148b3c:       f0 0f b1 29             lock cmpxchg %ebp,(%ecx)
c0150d0f:       f0 0f b1 3b             lock cmpxchg %edi,(%ebx)
c0150d87:       f0 0f b1 31             lock cmpxchg %esi,(%ecx)
c0199c5e:       f0 0f b1 0b             lock cmpxchg %ecx,(%ebx)
c024b06f:       f0 0f b1 0b             lock cmpxchg %ecx,(%ebx)
c024b2fe:       f0 0f b1 51 18          lock cmpxchg %edx,0x18(%ecx)
c024b321:       f0 0f b1 51 18          lock cmpxchg %edx,0x18(%ecx)
c024b34b:       f0 0f b1 4b 18          lock cmpxchg %ecx,0x18(%ebx)
c024b960:       f0 0f b1 53 18          lock cmpxchg %edx,0x18(%ebx)
(more)
```

cmpxchg

Defined as a preprocessor macro in:

- linux/arch/arm/include/asm/system.h, line 413
- linux/arch/x86/include/asm/cmpxchg_64.h, line 163
- linux/arch/x86/include/asm/cmpxchg_32.h, line 113
- linux/arch/x86/include/asm/cmpxchg_32.h, line 279
- linux/include/asm-generic/system.h, line 153
- linux/include/asm-generic/cmpxchg.h, line 19

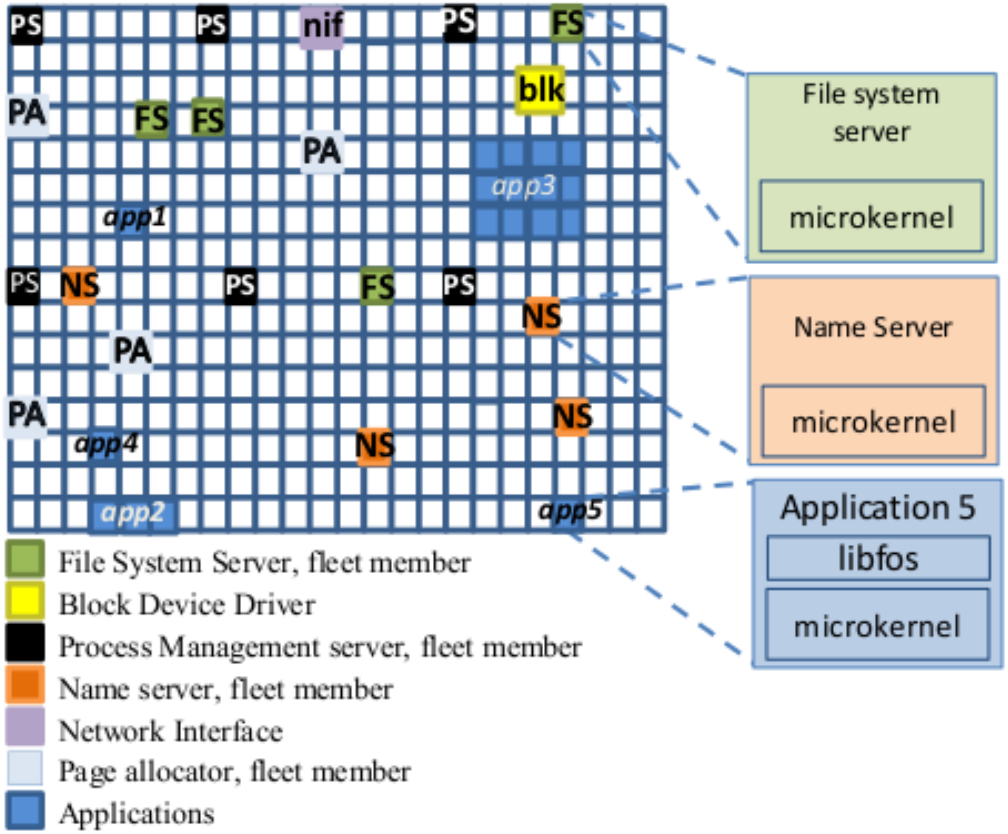
Referenced in **25 files** total (***2.6.31.13)*** !

Operating Systems for Many-core (1)

- **MIT Factored Operation System (fOS): 2009**
  - Target 1,000 core multicore chip
  - Space sharing replaces time sharing
- **Berkeley Tessellation (2009)**
  - "Cell" replace processes for performance isolation and QoS guarantees
- **Microsoft: Barrelfish**
  - **Multikernel** design: Build OS as a distributed system over all cores. Message passing among cores.
- **Berkeley ROS** (2010)
  - Space and time partitioning
  - 'many-core' process (MCP) abstraction

MIT fos: a Factored Operating System

- Space sharing replaces time sharing to increase scalability
- Mimic distributed Internet services
- fos's system servers communicate via message passing



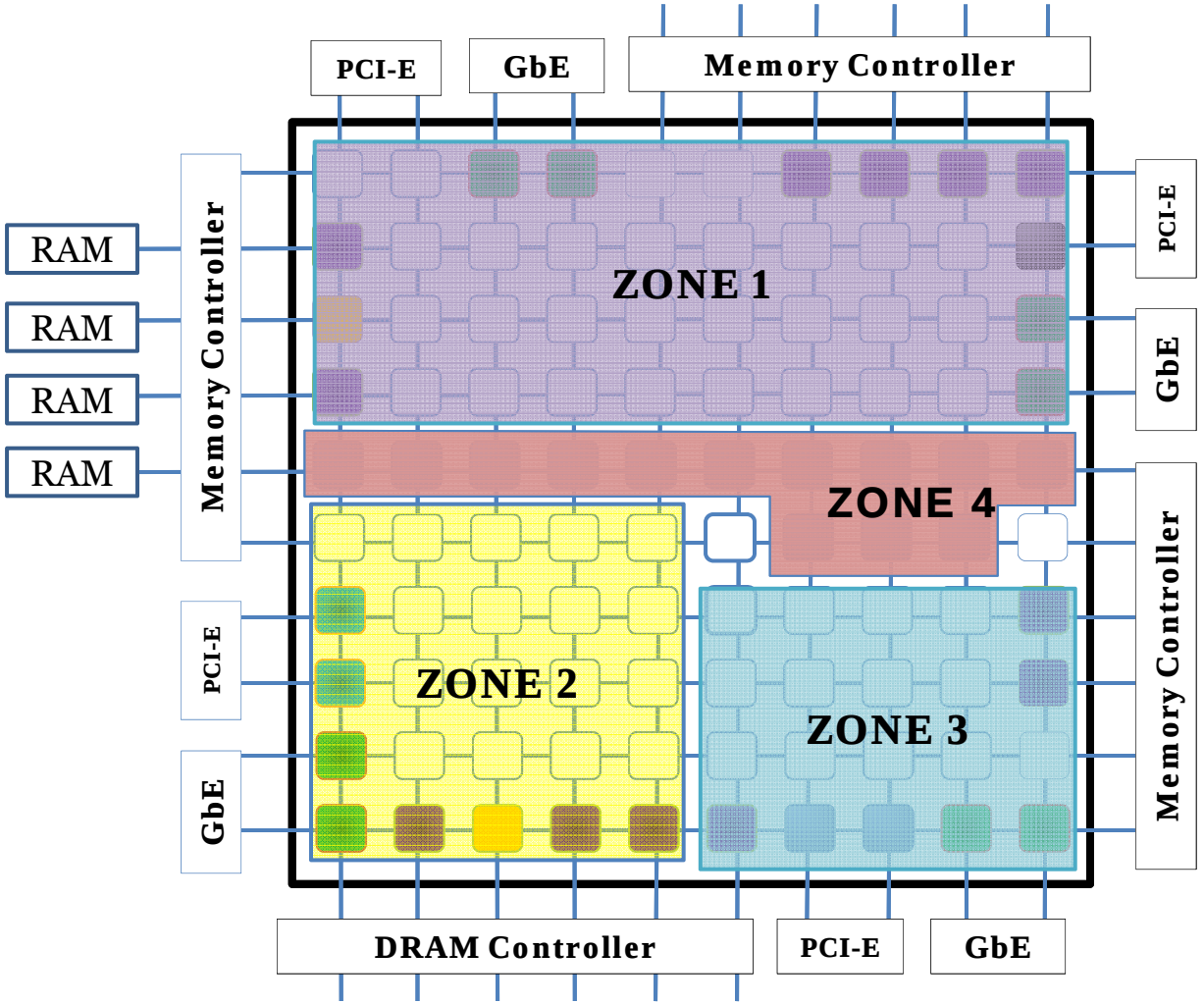
"Internet on a Chip"

Operating Systems for Many-core (2)

- **Microsoft Helios (2009)**
  - running on **heterogeneous hardware**, based on Singularity OS
  - satellite kernels, remote message passing, affinity
- **K42 (Since 1996): IBM, U of Toronto**
  - microkernel architecture, **object-oriented design**, research purposes
- **Corey (2008) :** MIT & Fudan &Microsoft Research Asia
  - exo-kernel, re-implementing OS data structures (file descriptor table, mm_struct) and user APIs
- **µKMC (2012-): : U. of Tokyo**
  - **light-weight micro kernels on Intel MIC**, starts from July 2012.
  - accelerator abstraction layer (AAL), inter-kernel communication layer (IKCL)
- **Berkeley Akaros (2010-2013)**
  - **Asymmetric OS structure** to scale to thousands of cores.
  - per-core private memory, syscalls are ”context switch free”

鰐魚 @ HKU (01/2013-12/2015)

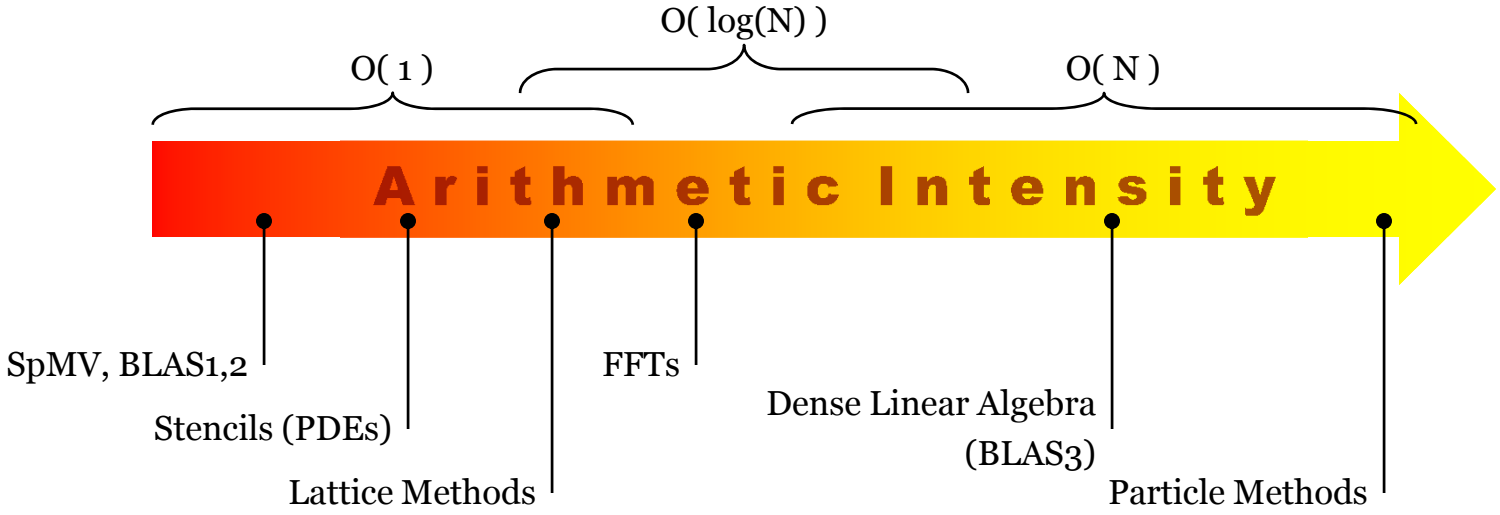
- **Crocodiles**: Cloud Runtime with Object Coherence On Dynamic tILES for future 1000-core tiled processors”



Challenges and Potential Solutions (1)

- **Stop moving so much data around**
  - Data Locality/Working Set getting critical!
  - 3D stacked memory (TSV technology) helps !
  - Compiler or runtime techniques to improve data reuse and increase arithmetic intensity (next slide)
  - Cache-aware design (temporal locality becomes more critical)
  - Migrating "code & state" instead of data → Thread migration among cores (+ large 3D stacked memory !).
- **Stop multitasking**
  - Context switching breaks data locality
  - No Time Sharing → Space Sharing

Arithmetic Intensity

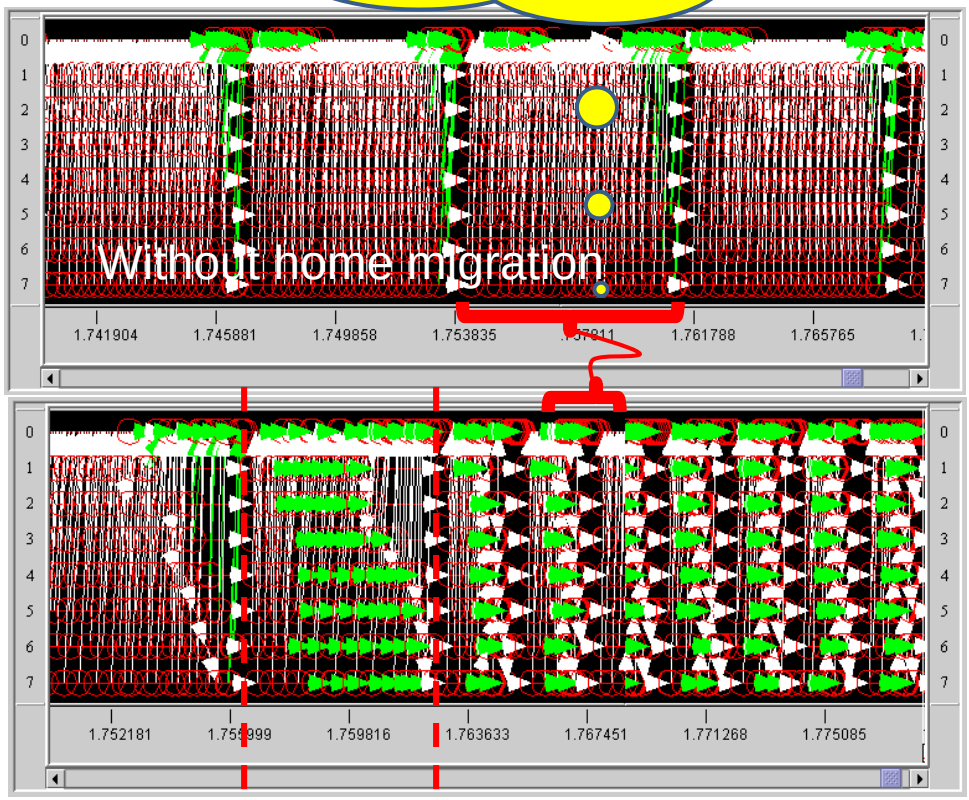


- **Arithmetic intensity is defined as the number of operations performed per word of memory transferred**
- It is important for Big Data applications to have high arithmetic intensity, otherwise the memory access latency will limit computational speedup

Challenges and Potential Solutions (2)

- **Software-managed cache coherence**
  - Leverage programmable on-chip memory (e.g., MPB on Intel SCC)
  - **Scope consistency (ScC)** : minimizing on-chip network and off-chip DRAM traffic
  - **Migrating-home ScC Protocol (MH-ScC) → improve data locality**

With home migration, each phase took much less execution time.

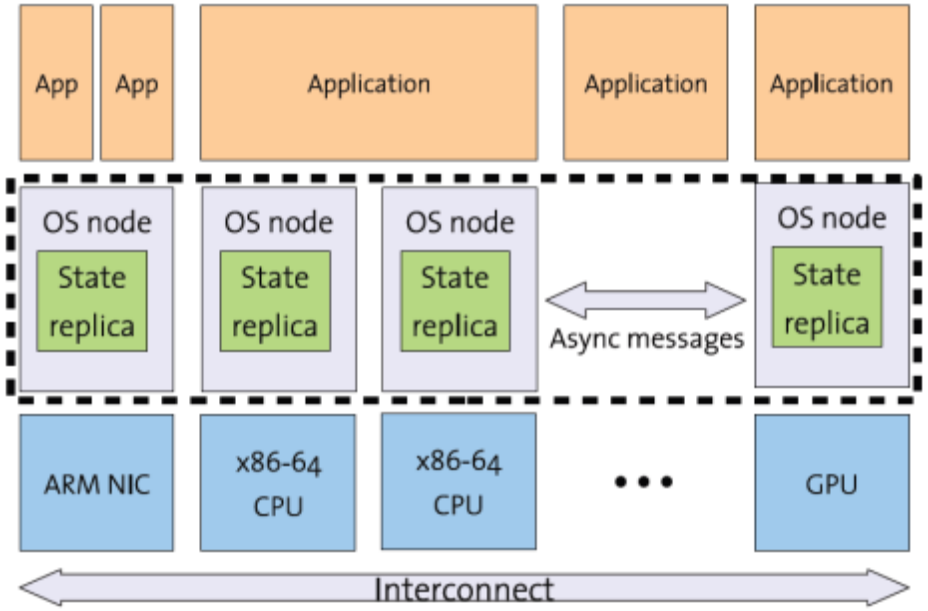


Before Home Migration | *Migrating phase* | *After Home migration*

Simulation results obtained in a 8-node cluster (SOR program)

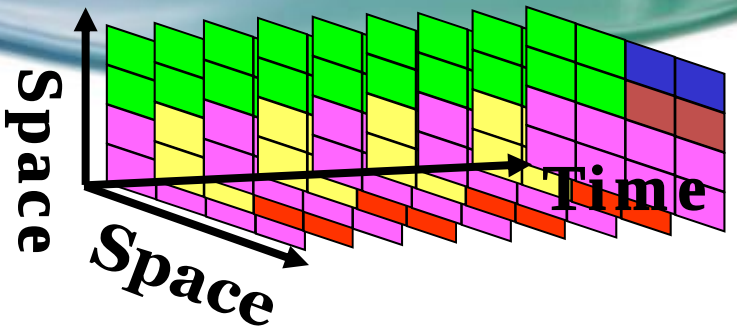
Challenges and Potential Solutions (3)

- **Scalability (up to 1000 cores?)**
  - **Adopt multikernel operating system (e.g., Barrelfish) to reduce contentions on shared structures in OS kernel**
  - **Shared memory → message passing**
    - **Barrelfish : *"Compact message cheaper than many cache lines-- even on a cache-coherent machine."***

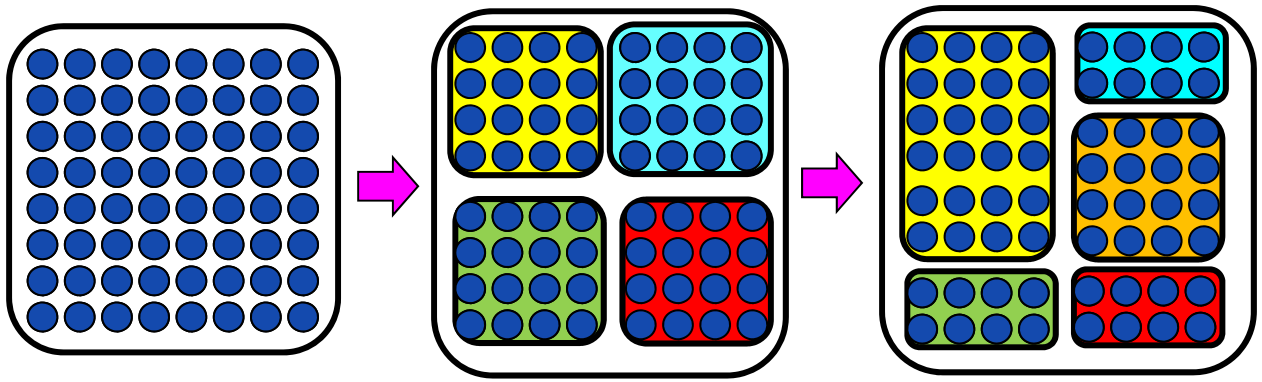


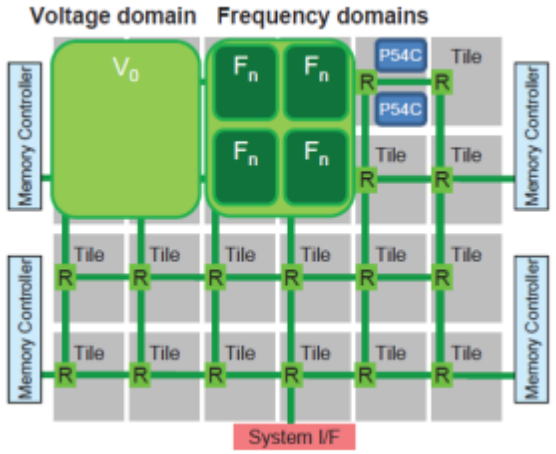
View state as replicated: Maintain state through replication rather than shared memory (improved locality)

Challenges and Potential Solutions (4)



- **Dynamic Zoning for Elasticity of Demand**
  - "Zoning" (Spatial Partition) → Performance isolation
  - "Dynamic Zoning": on-demand scaling of resources (e.g., # of cores, DRAM,..) for each zone.
  - Partitioning varies over time, mimic multi-tenant Cloud Architecture → "Data center on a Chip"
  - Fit well with the domain-based power management (e.g., Intel SCC)





Conclusion

- **GHz game is over→ Go for Manycore**
  - World has gone to manycore to continue Moore's Law
  - "General-purpose" 100-core chip is available (e.g., Tilera TILE-Gx), 1000-core chip is expected soon (2017?)
  - Intel MIC to be used in China's 100 petaflops machine?
- **PCIe bottleneck problem?**
  - CPU-GPU mashup (e.g., APU)
- **Big data computing on 1000-core chip is tough**
  - **Locality is critical** (compute is "free", avoid moving data around)
  - Power efficiency is the key challenge (flops/watt)
  - Low AI problem: Data reuse techniques for high **flops/byte**

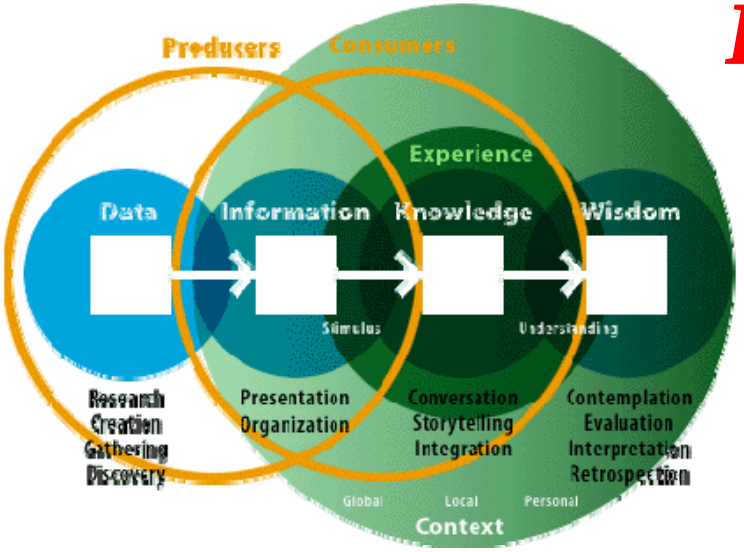
Conclusion(Cont'd)

- **Scalability issues in all layers:**
  - Hardware (NoC), OS, software cache coherency, programming model
- ***"DON'T MOVE THE DATA!"***
  - **Implication:** ***moving code & state instead***
  - **Try "Multi-granularity Computation Migration"**
- **Research in system software is hard.**
  - There are rarely clearly right or clearly wrong solutions. No "***one-size-fits-all" solution.***
  - Difficult to compare: No standard interfaces
  - **Pressures from academic publication volume or deliverables**

Questions?

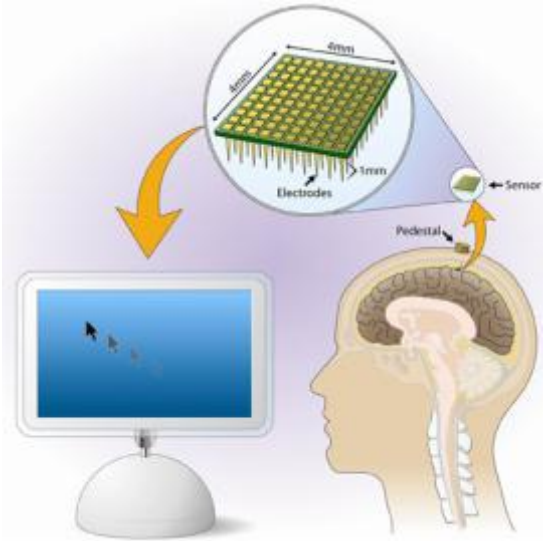
Part IV

From Data to Intelligence -- *Context Reasoning*



Context Reasoning

- ***Data is only valuable when you can gain insights from it to make decisions***
- ***Context Reasoning:***
  - deducing new and relevant information to the use of application(s) and user(s) from the various sources of context-data.
- These tasks include: (1) *context data pre-processing, (2) sensor data fusion and* ***(3) mapping lower level context*** **into higher level context (which is also known as *context inference*).**



Context Reasoning : Significant Places Detection

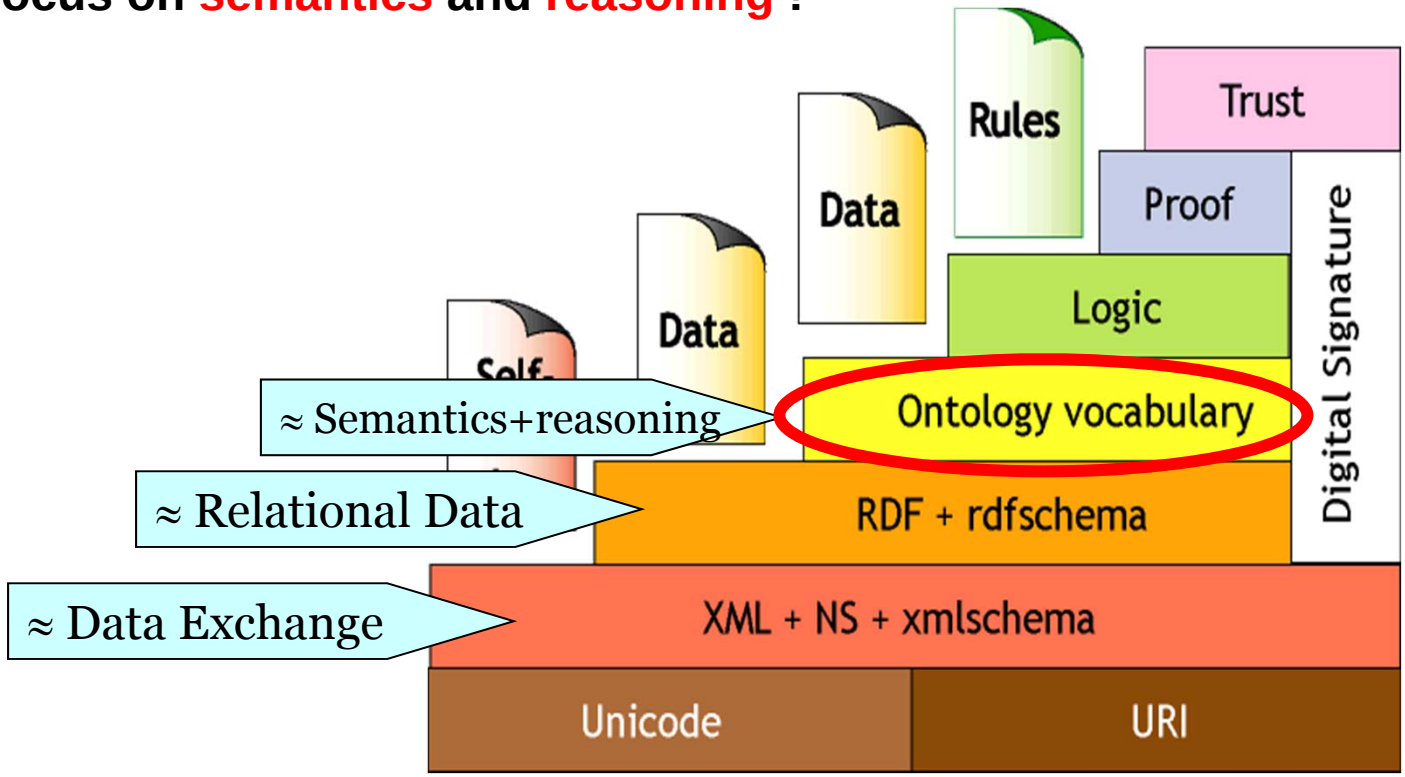
From lower-level raw data to *meaningful* higher-level context

(a) Seven extracted places:

a: King George V Memorial Park
b: 7-Eleven convenience store
c: Pizza-Box store
d: Bus station
e: Flora Ho Sports Centre
f: Pokfulam Road Playground
g: a restaurant

Ontology-based Context Modeling

- Ontologies provide a **vocabulary** of terms
  - Meaning (*semantics*) of such terms is formally specified
  - New terms can be formed by combining existing ones
- **Focus on semantics and reasoning !**



Resource Description Framework (RDF)

post-Hadoop era (取代GFS 和 MapReduce)

- **Google Caffeine (2010)**
  - 主要为Google网络搜索引擎提供支持 (2010)
  - 将索引放置在由Google的分布式数据库BigTable上
- **Google Pregel (SIGMOD 2010)**
  - **Large-scale graph processing (**图形数据库)
- **Google Dremel (VLDB 2010)**:
  - interactive ad-hoc query
  - 可以在几秒的时间处理PB级的数据查询 (BigQuery)
- **Google Percolator**:
  - for incremental processing (Bigtable)
- **Apache Giraph** (Open Source)
  - HDFS + Zookeeper

ContextTorrent

semantically organize, search, and store various types of contexts and their semantic relationships using ontology-based semantic technologies

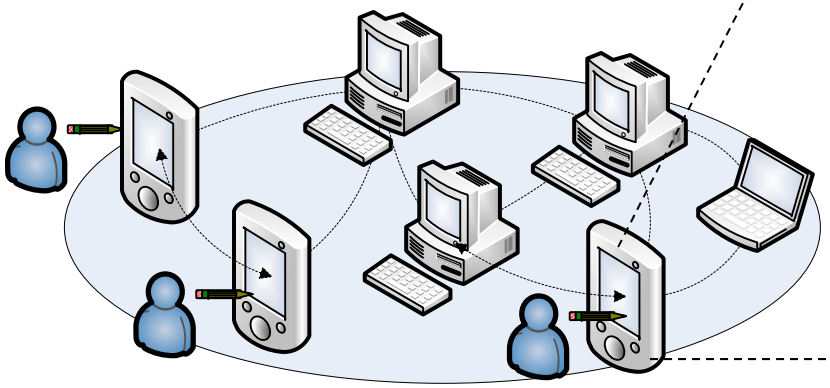
Ontology-based Context Modeler

RDF triple: (*subject, predicate, object*)

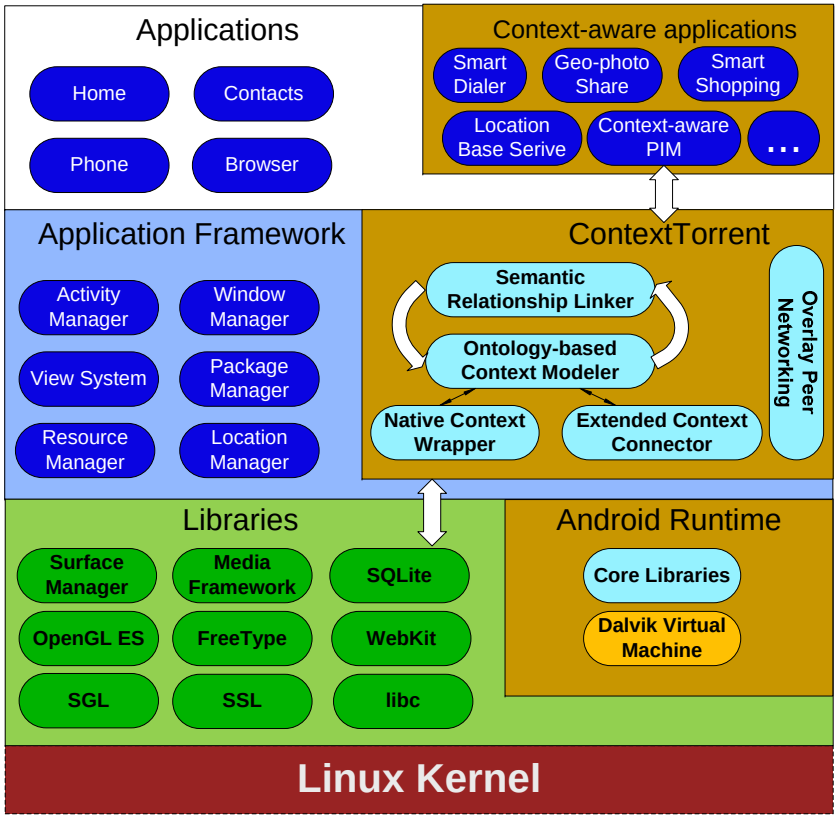
<person:"Dexter", hasAge, "25">,

<person: "Tom", hasLocation, (110.1, 20.3)>

<hk_weatherstation, hasPredictinon, "Sunny">



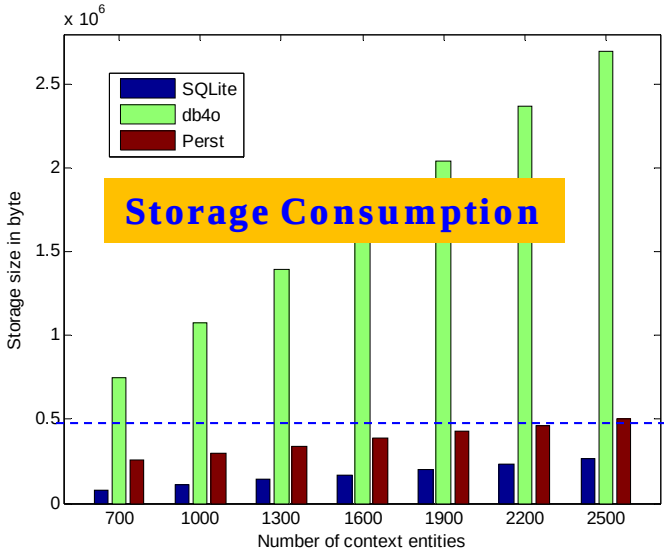
Peer to Peer Communication



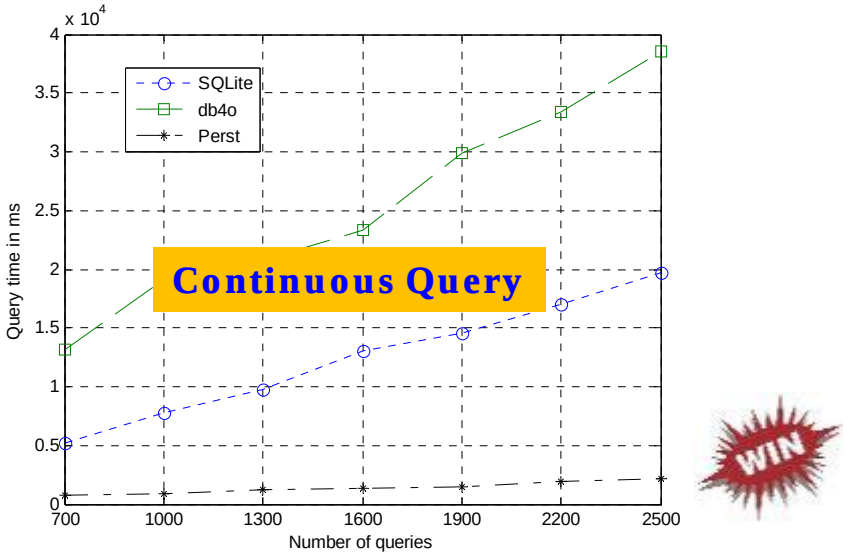
ODBMS (db4o, Perst): object-orientation analogous to ontological representation

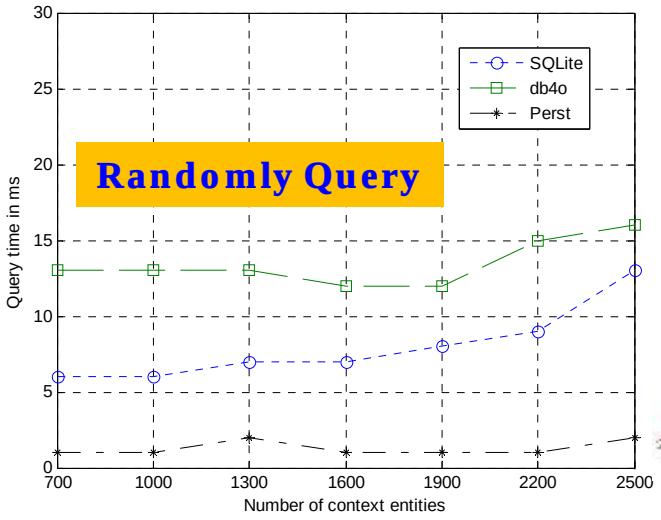
Dexter Hu, "ContextTorrent: a Context Provisioning Framework for Pervasive Applications", Ph.D Thesis, March 2011.

Evaluation

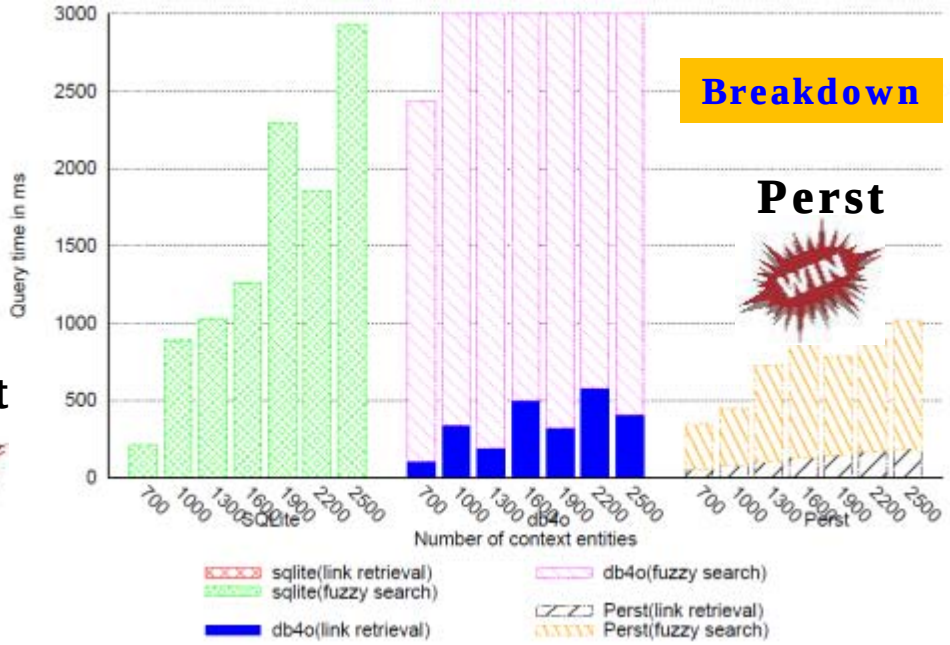


Perst

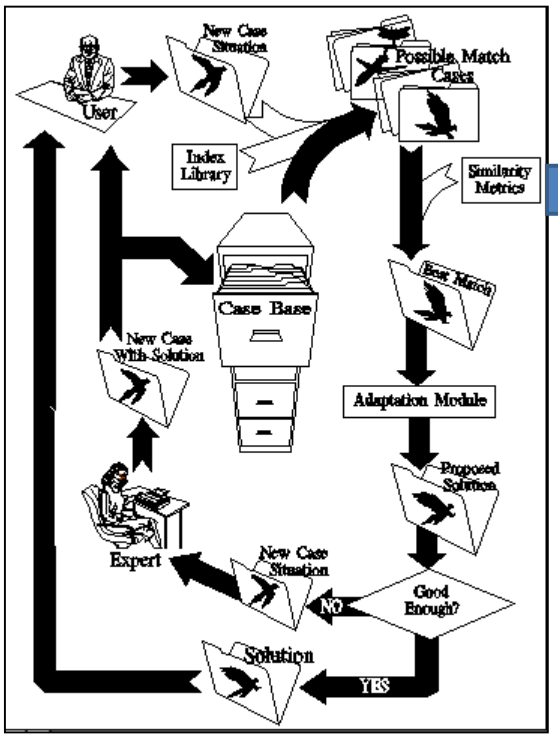




Perst

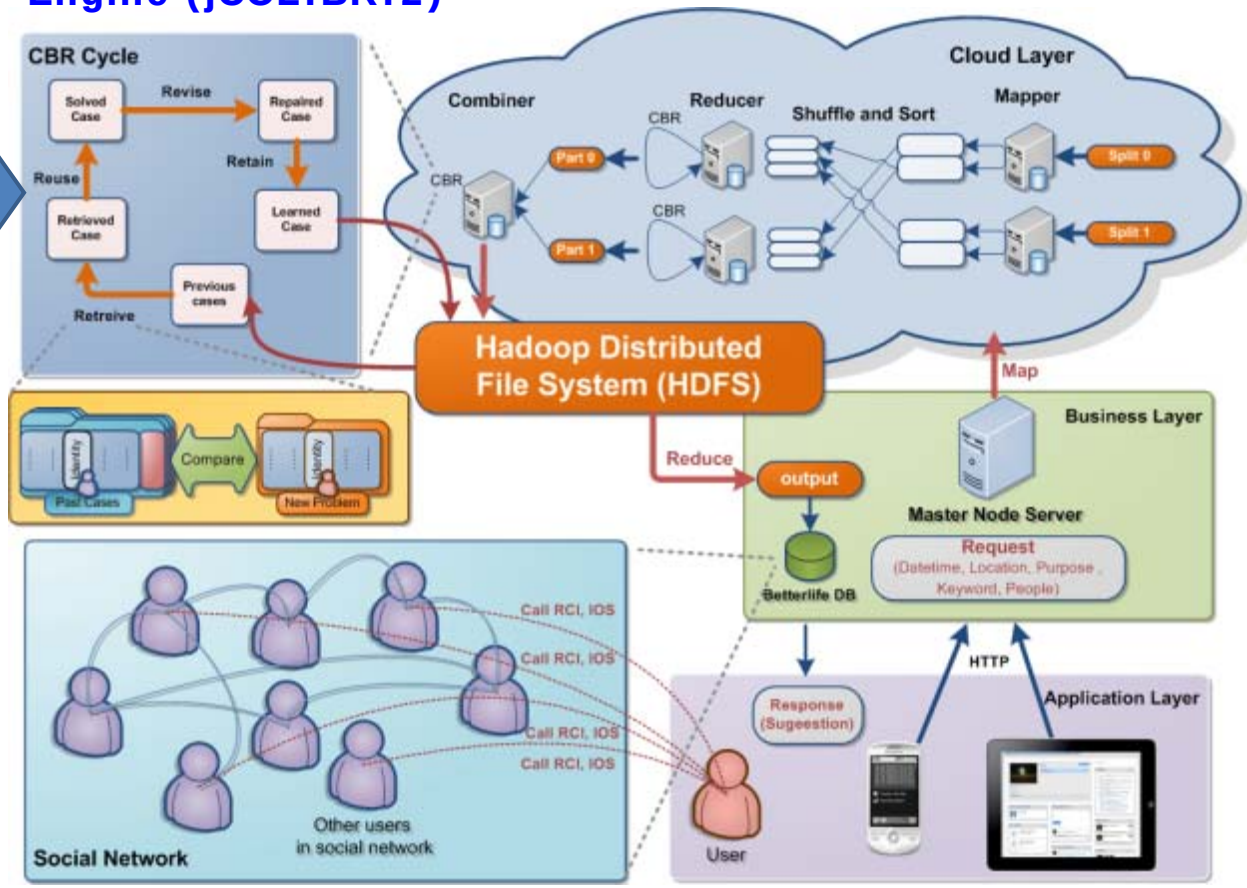


BetterLife 2.0: Large-scale Social Intelligence Reasoning



Case-based Reasoning

Case-based Reasoning Engine (jCOLIBRI2)

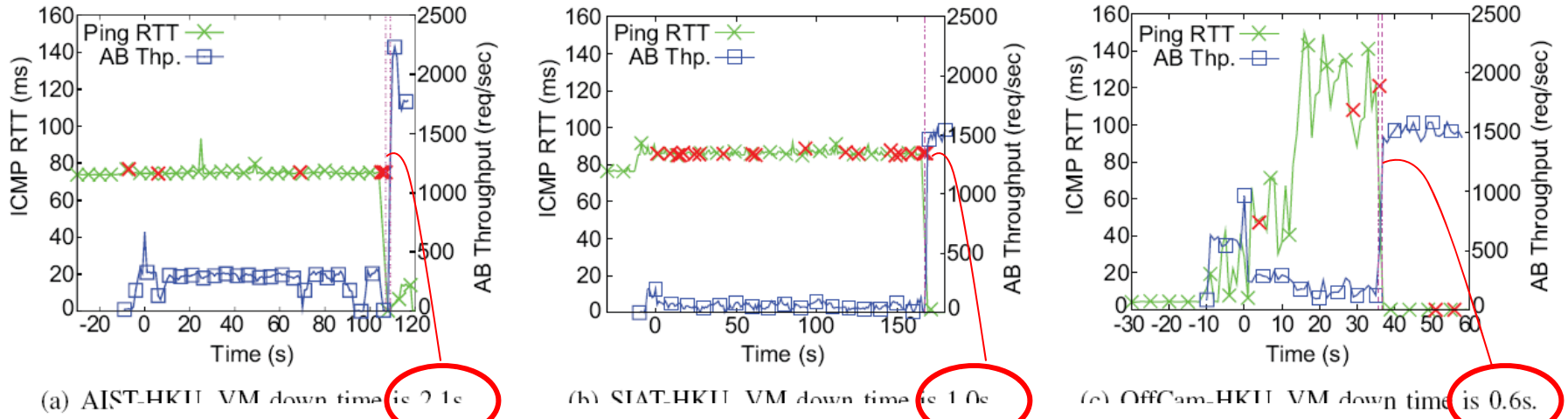


Social Network

Dexter H. Hu, Yinfeng Wang, Cho-Li Wang, "BetterLife 2.0: Large-scale Social Intelligence in Cloud Computing“ (CloudCom 2010)

WAVNet: Live VM Migration over WAN

Ping RTT drops after migration
Freeze Time: 0.6s ~ 2.1s



ICMP RTT and HTTP throughput during VM live migration
(x *represents ICMP packet loss)*