

The University of Hong Kong



WHAT IS BIG DATA?

Big Data: The “4Vs” Model

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




30 billion pieces of content were added to Facebook this past month by 600 million plus users.


32 billion searches were performed last month... on Twitter.


More than 2 billion videos were watched on YouTube... yesterday.

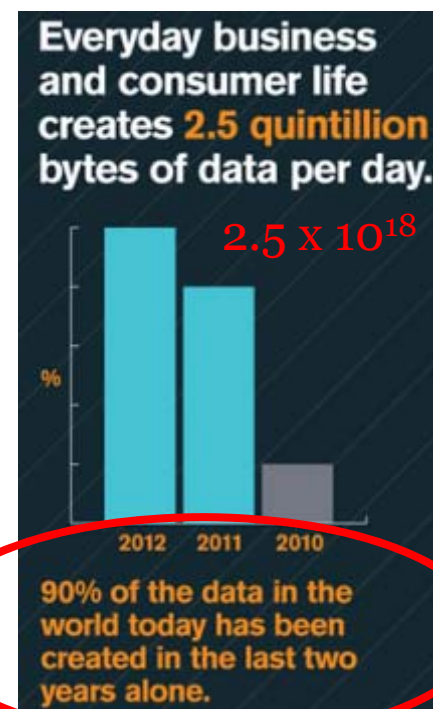

Zynga processes 1 petabyte of content for players every day; a volume of data that is unmatched in the social game industry.

By 2015, nearly
3 billion people



will be online, pushing the data created and shared to nearly
8 zettabytes.

Worldwide IP traffic will
quadruple by 2015.



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: (12/2012)

Big Data vs. Data Analytics vs. Cloud Computing

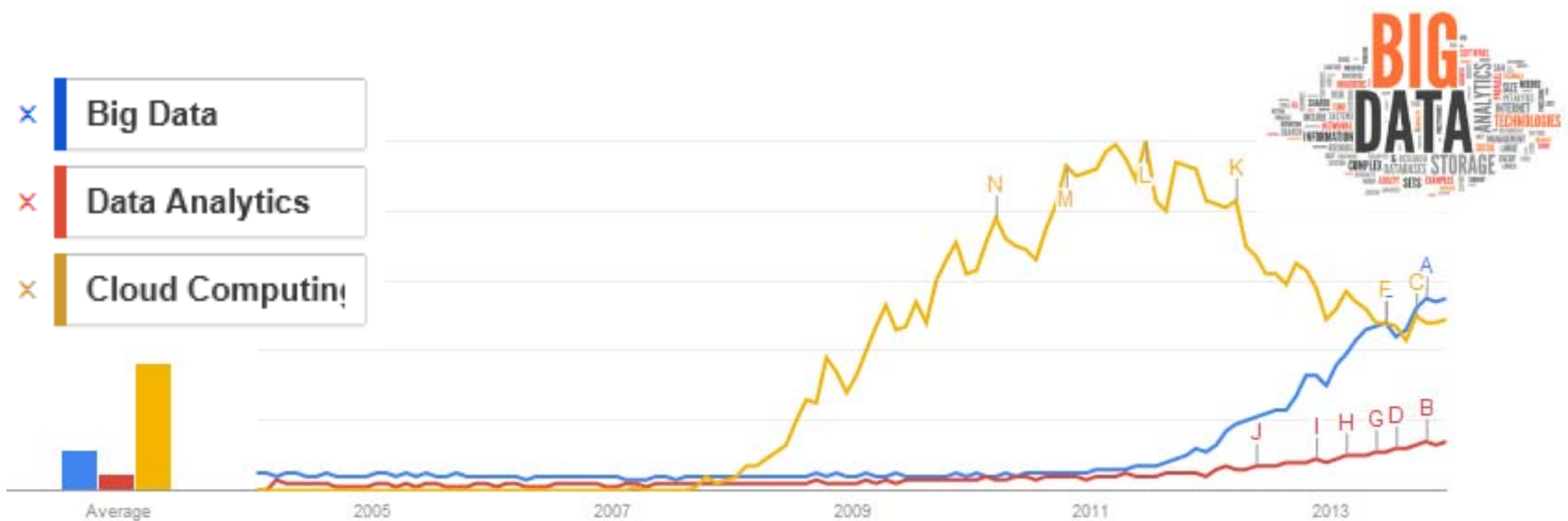


- **McKinsey Global Institute (MGD) :**

- 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).

Google Trend: 12/2013

Big Data vs. Data Analytics vs. Cloud Computing



“Big Data” in 2013



Outline

- **Part I: Multi-granularity Computation Migration**
 - "A Computation Migration Approach to Elasticity of Cloud Computing" (previous work)
- **Part II: Big Data Computing on Future Maycore Chips**
 - **Crocodiles: Cloud Runtime with Object Coherence On Dynamic tILES** for future **1000-core** tiled processors" (ongoing)



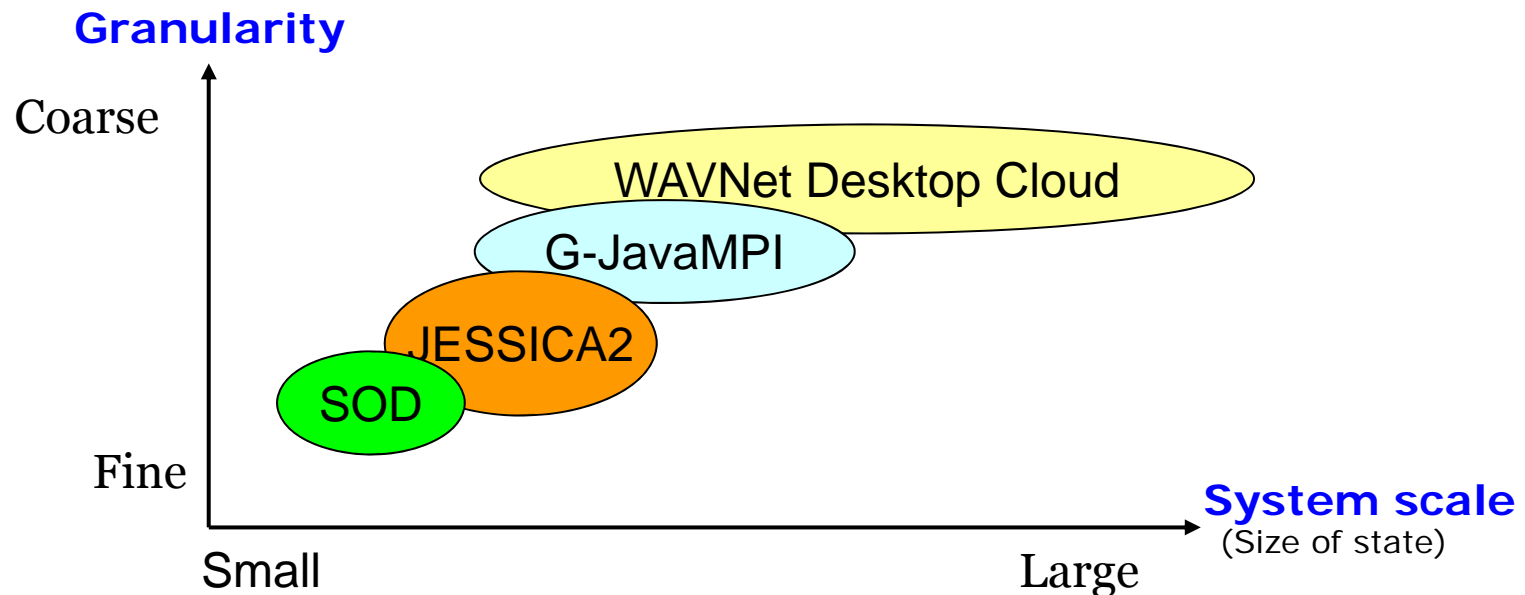
Part I

Multi-granularity Computation Migration

Source: Cho-Li Wang, King Tin Lam and Ricky Ma, "A Computation Migration Approach to Elasticity of Cloud Computing", Network and Traffic Engineering in Emerging Distributed Computing Applications, IGI Global, pp. 145-178, July, 2012.

Multi-granularity Computation Migration

	<u>Granularity</u>	<u>Migration Technique (System)</u>	<u>Target System Type (Area)</u>
→	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)

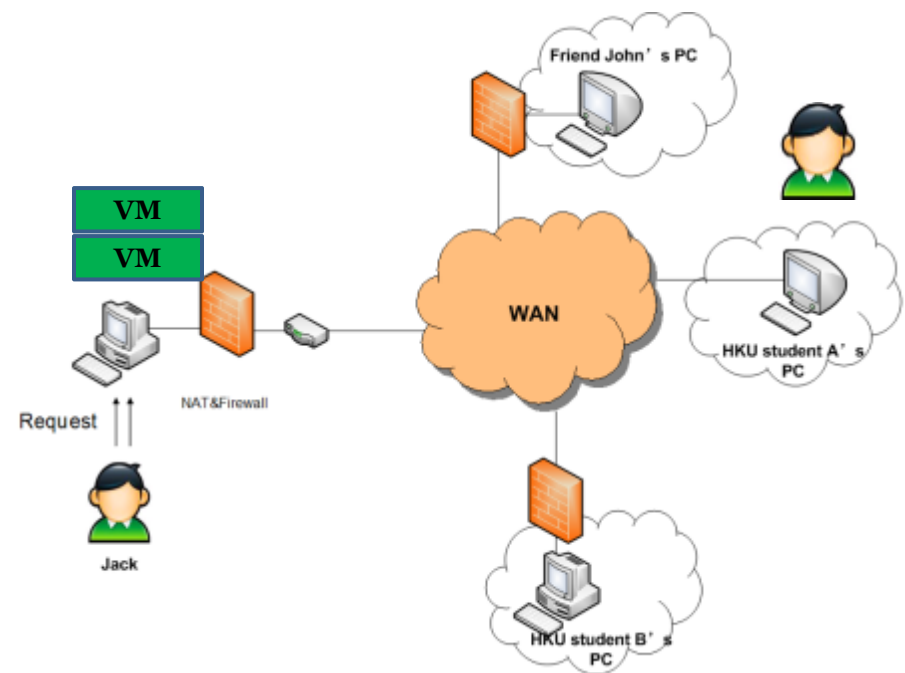
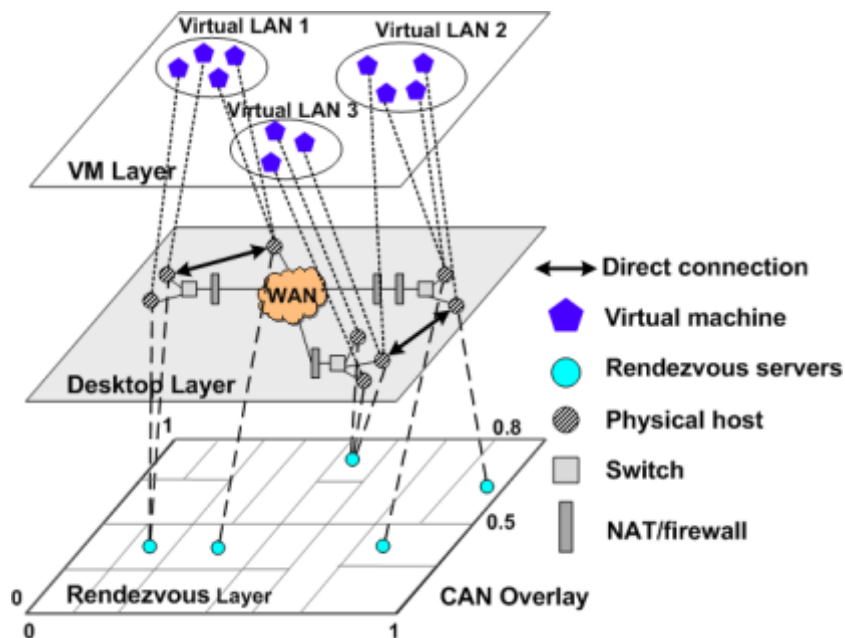


(1) WAVNet: Live VM Migration over WAN

- 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



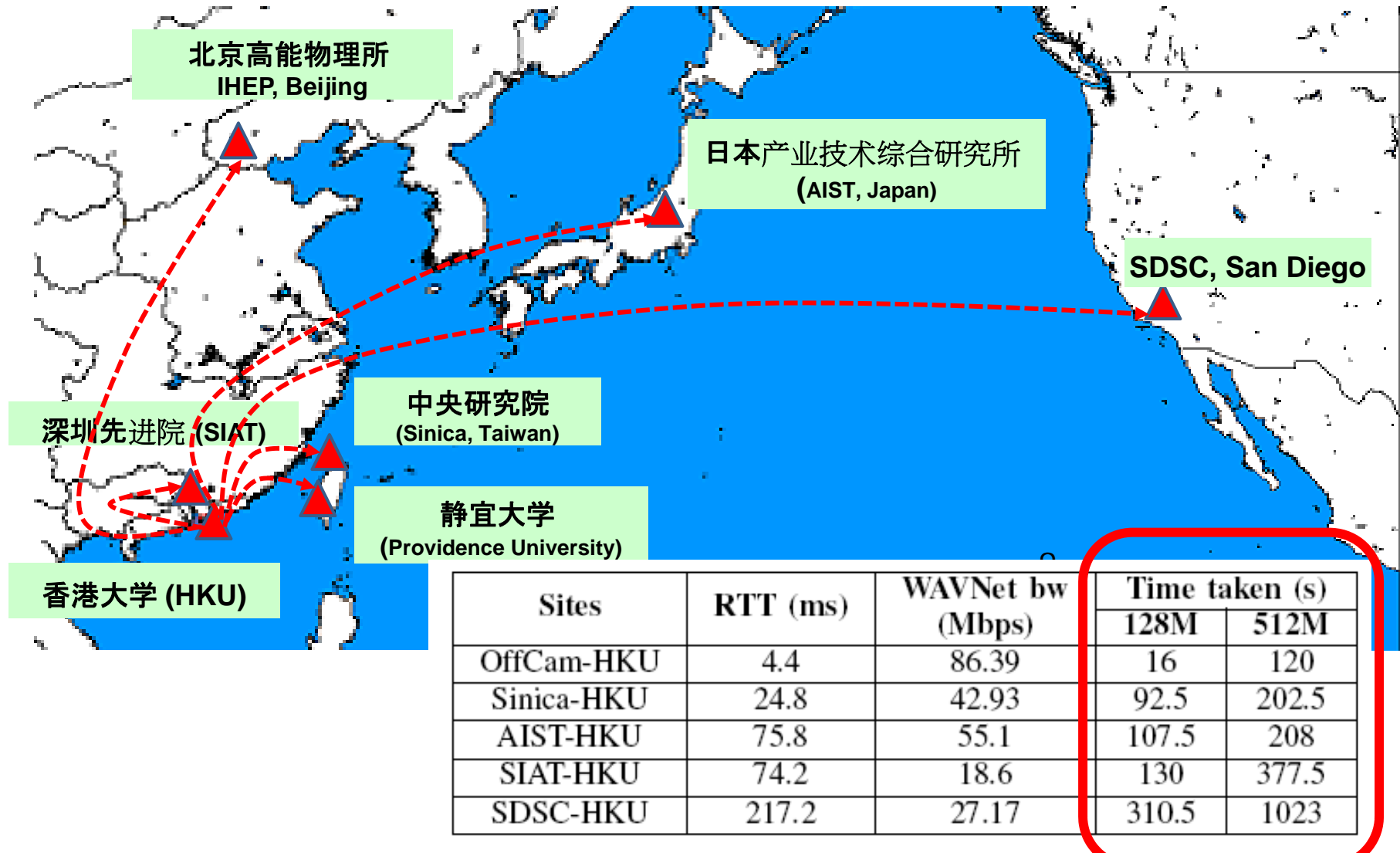
Key Members


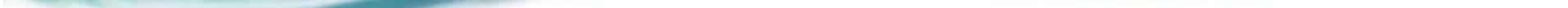
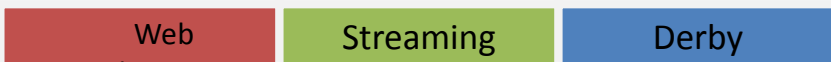


Zheming Xu, Sheng Di, Weida Zhang, Luwei Cheng, and Cho-Li Wang, WAVNet: Wide-Area Network Virtualization Technique for Virtual Private Cloud, 2011 International Conference on Parallel Processing ([ICPP2011](#))

WAVNet: Live VM Migration over WAN

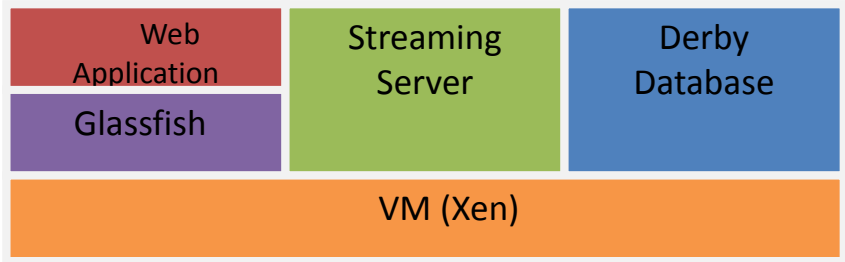
◦ Experiments at Pacific Rim Areas



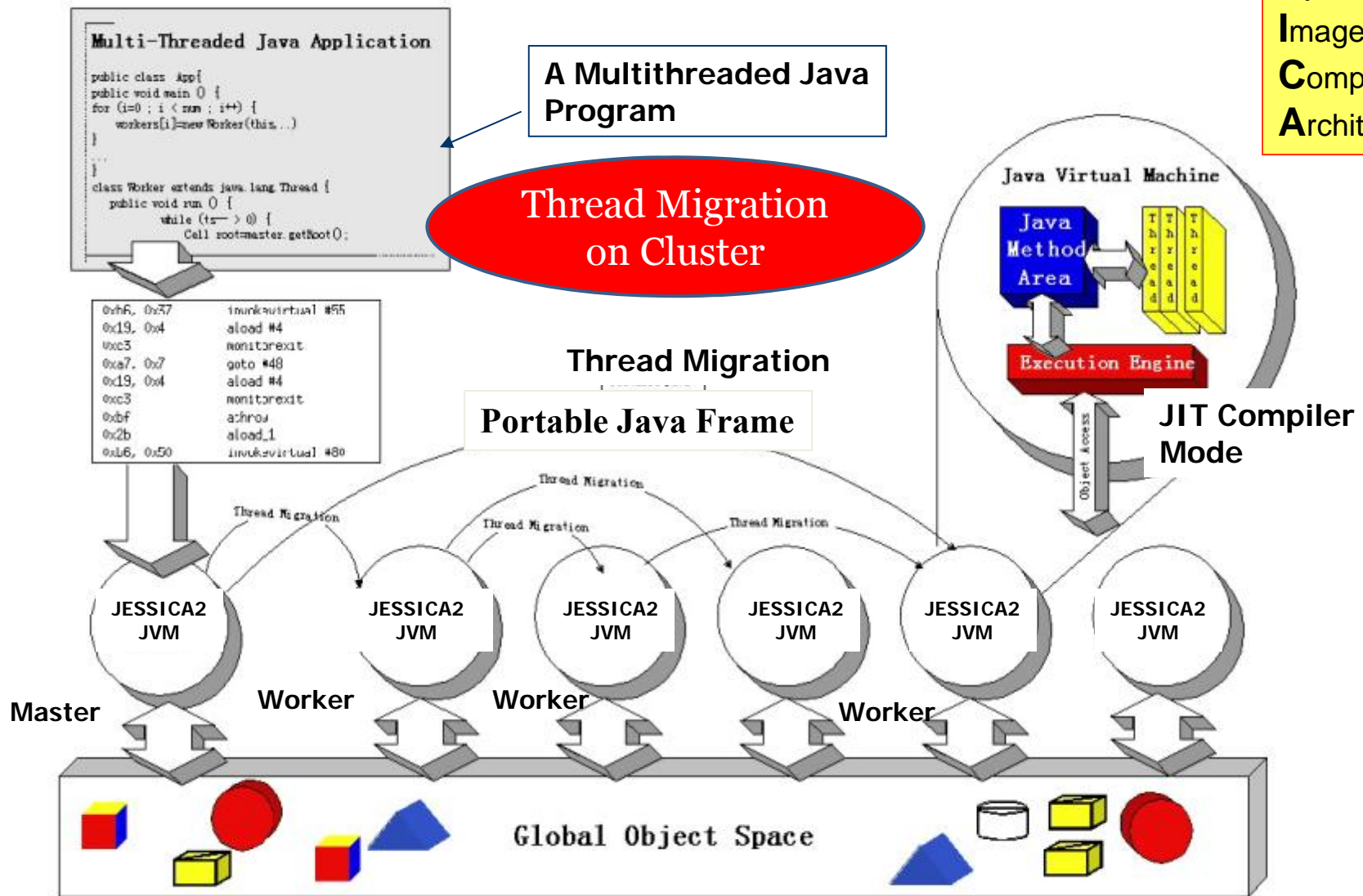
```

graph TD
    subgraph Applications
        direction LR
        WS[Web Application]
        WS --- GS[Glassfish]
        WS --- SS[Streaming Server]
        WS --- DD[Derby Database]
    end
    Applications --- VM[VM Xen]
  
```

A red starburst graphic with a white outline and a dark blue border. Inside the starburst, the text "Share Your Story" is written in a white, bold, sans-serif font, angled diagonally upwards from left to right.

Prototyped by Eric Wu, Queena Fung

(2) JESSICA2 : Thread Migration on Clusters



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~)**
 - **Japanica**: Automatic loop parallization and speculative execution on GPU and multicore CPU.
Handle dynamic loops with runtime dependency checking



Past Members



King Tin LAM,



Chenggang Zhang



Kinson Chan

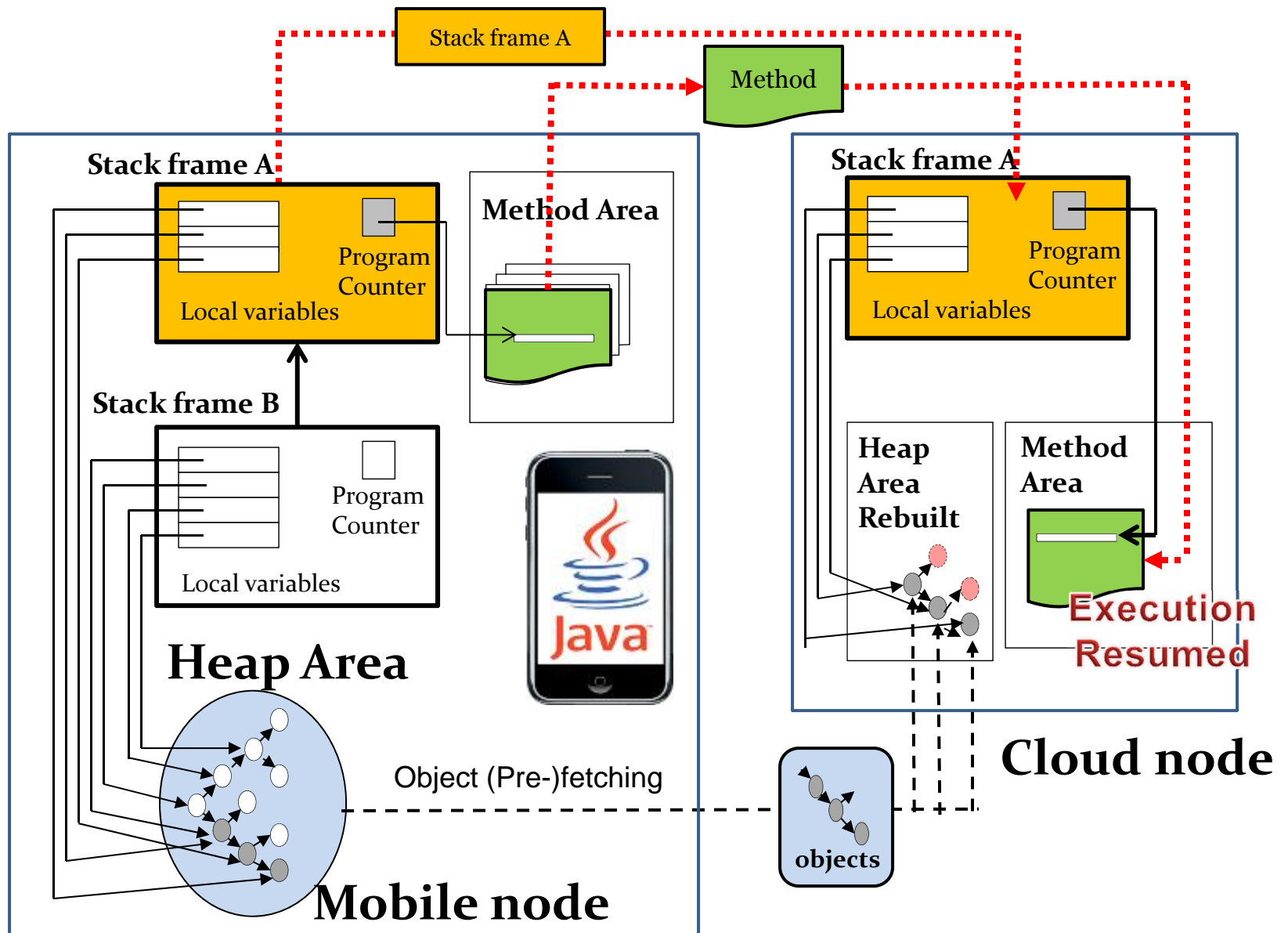


Ricky Ma

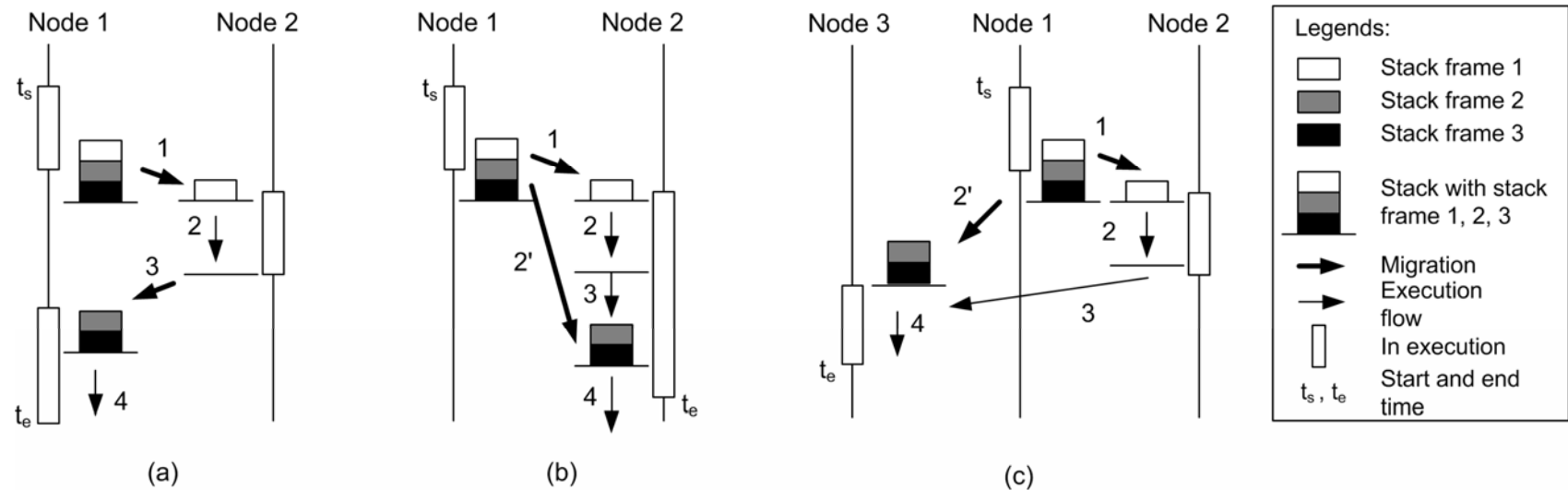
Download JESSICA2:

<http://i.cs.hku.hk/~clwang/projects/JESSICA2.html>

(3) Stack Migration: “Stack-on-Demand” (SOD)



SoD enabled the “Software Defined” Execution Model



(a) “Remote Method Call”

(b) Thread migration

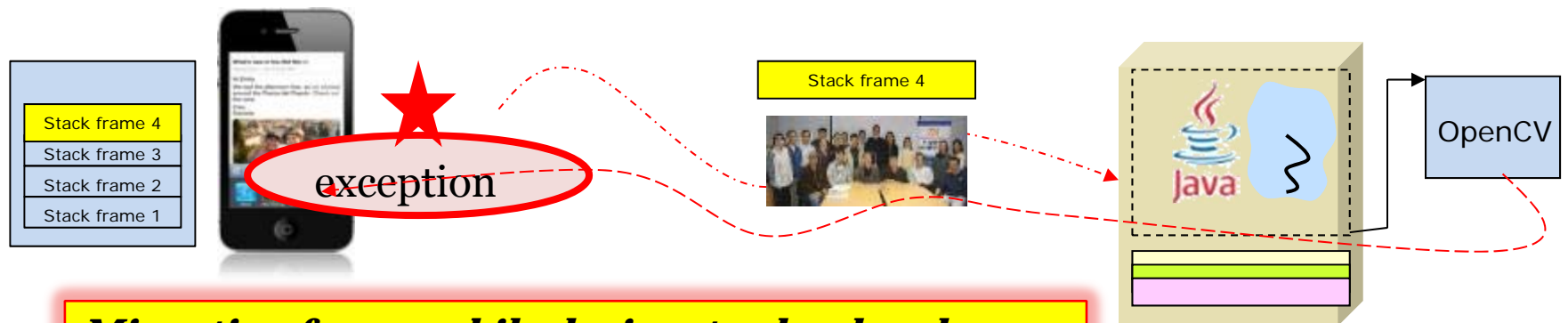
(c) “Task Roaming” or “Workflow”

With such flexible or *composable* execution paths, SOD enables agile and elastic exploitation of distributed resources (storage) → **Exploit Data Locality in Big Data Computing !**

SOD : Face Detection on Cloud



apps	capture time (ms)	transfer time (ms)	restore time (ms)	total migration latency (ms)
FaceDetect	103	155	7	265



Migration from mobile devices to cloud node

SOD: “Mobile Spider” on iPhone

Bandwidth (kbps)	Capture time (ms)	Transfer time (ms)	Restore time (ms)	Migration time (ms)
50	14	1674	40	1729
128	13	1194	50	1040
384	14	728	29	772
764	14	672	31	717

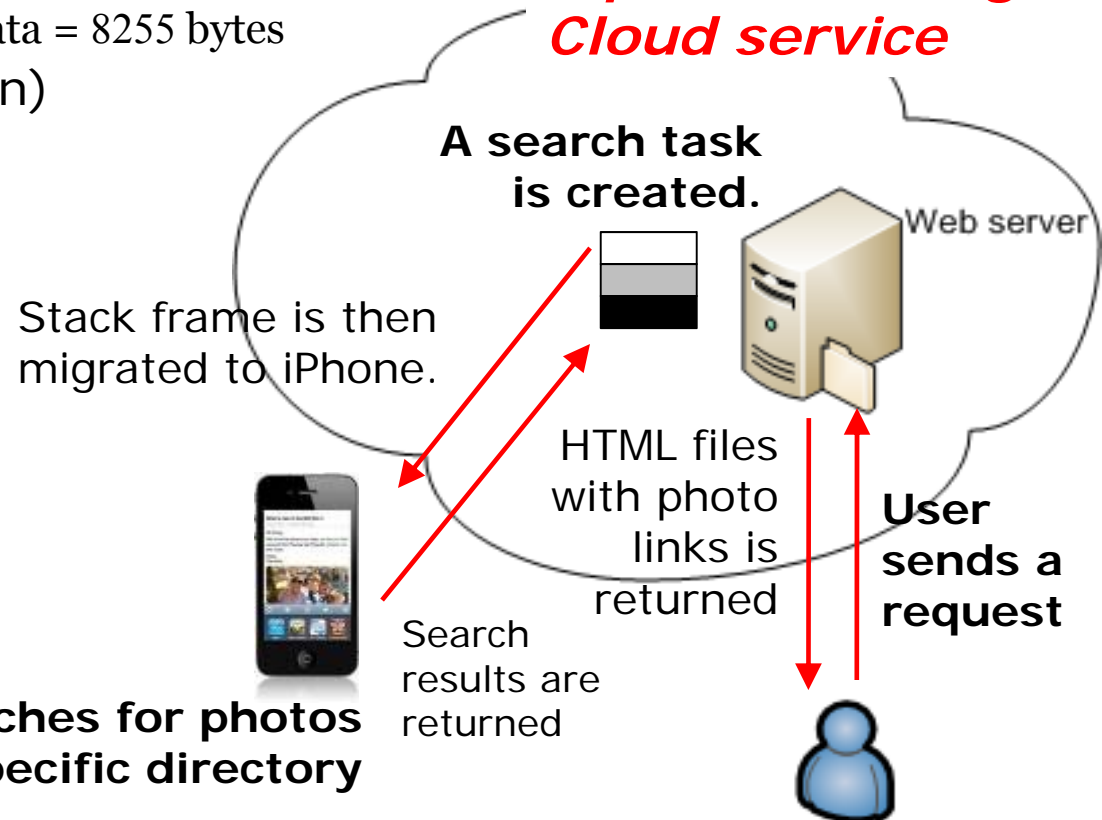
Size of class file and state data = 8255 bytes
(with Wi-Fi connection)

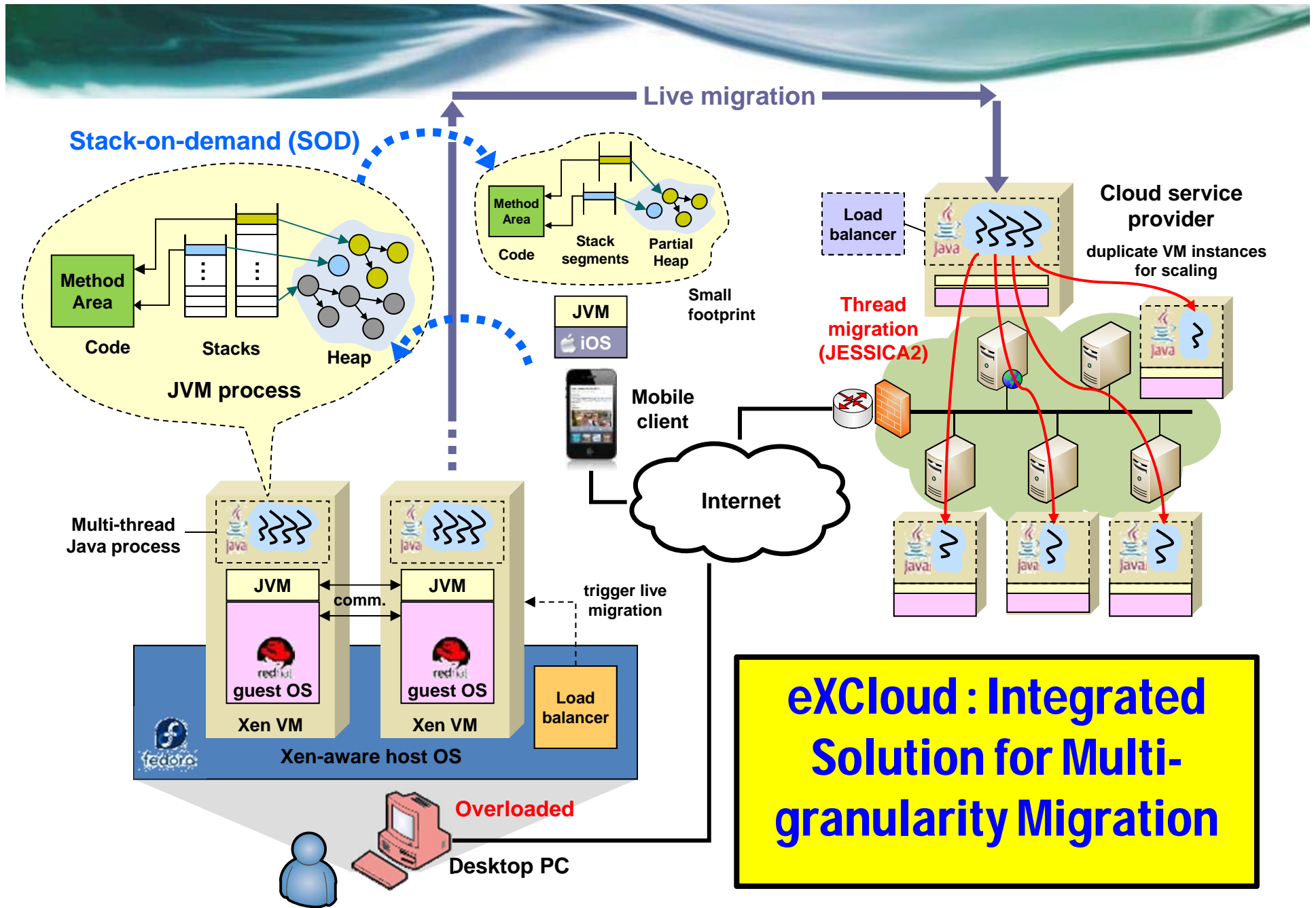
***Migration from
cloud node to
mobile devices***

***A photo sharing
Cloud service***

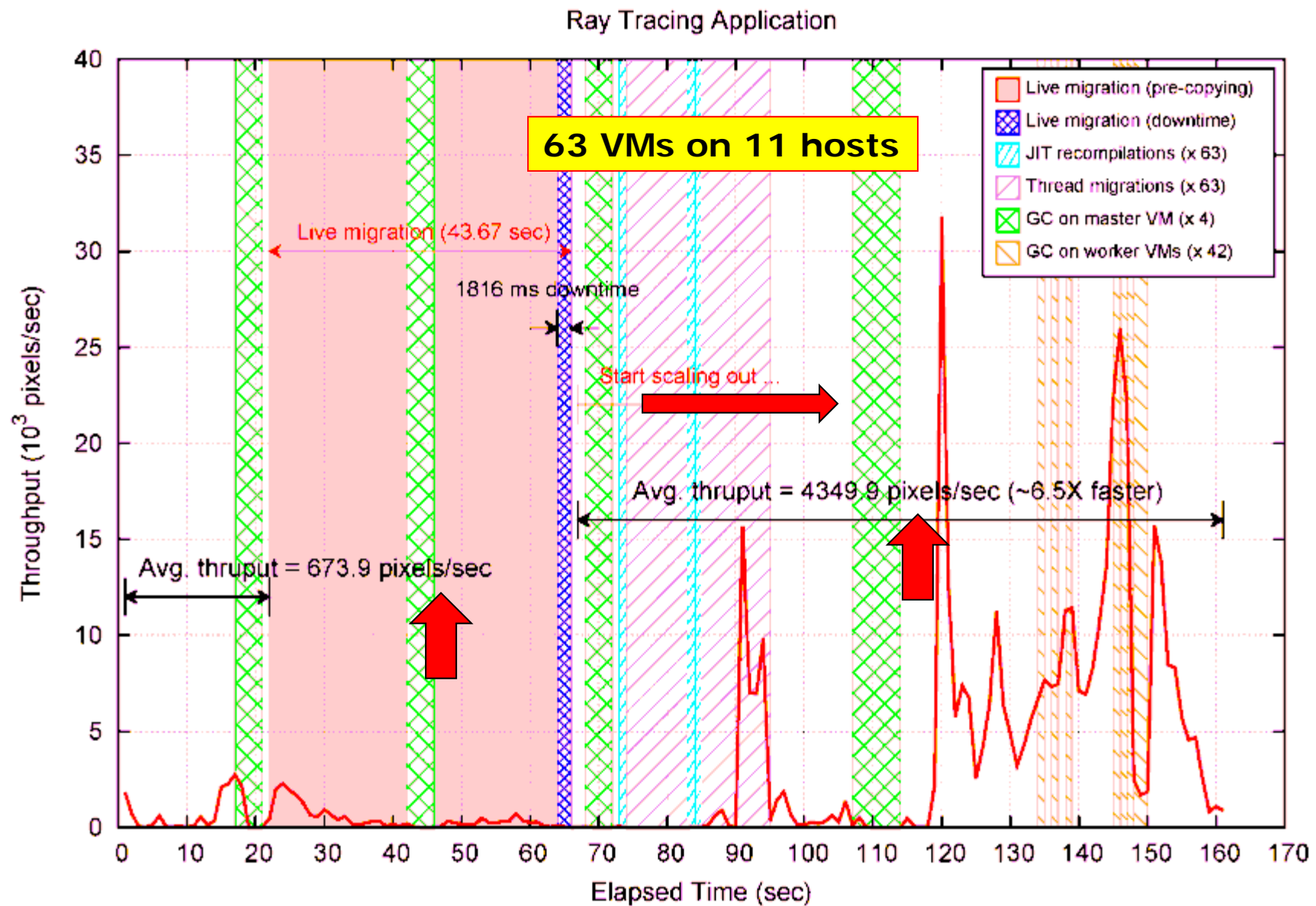


**The task searches for photos
available in the specific directory**





“JESSICA on Cloud”: VM Migration + Thread Migration



Comparison of Migration Overhead

Migration overhead (MO)

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

App \ Sys	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



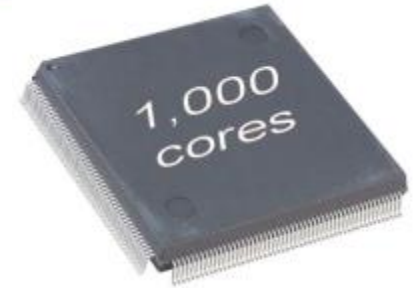
Part II

Big Data Computing on Future Maycore Chips



Crocodiles: Cloud Runtime with
Object Coherence On Dynamic **tILES**
for future **1000-core** tiled processors”
(1/2013-12/2015, HK GRF)

The 1000-Core Era



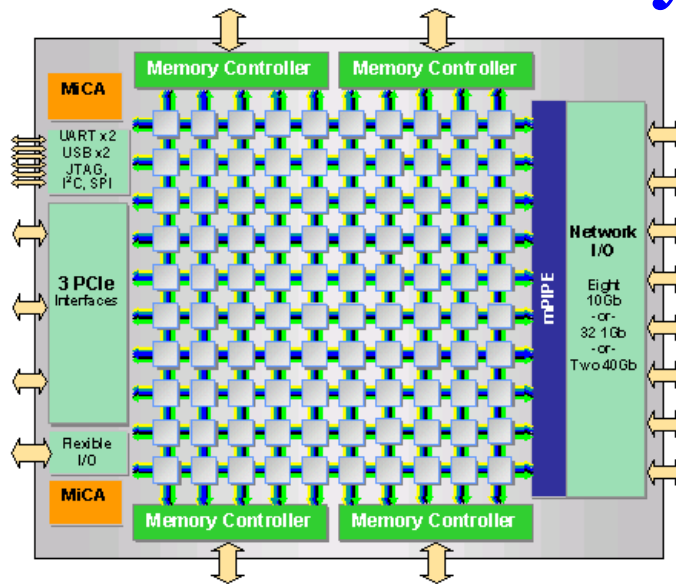
- Experts predict that by the end of the decade we could have as many as 1000 cores on a single die (S. Borkar, “Thousand core chips: a technology perspective”)
- **International Technology Roadmap for Semiconductors (ITRS) 2011 forecast:**
 - **By 2015: 450 cores**
 - **By 2020: 1500 cores**
- **Why 1000-core chip ?**
 - Densely packed servers cluster → Cloud Data Center in a Chip
 - Space efficiency + Power Efficiency (Greener)

Tiled Manycore Architectures

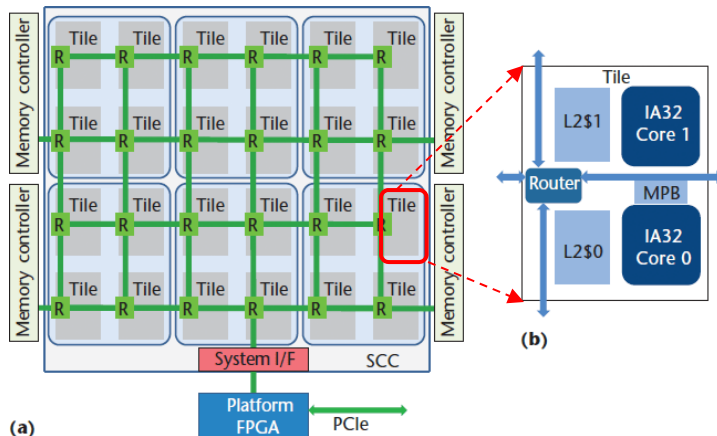
Micro-architecture	# of cores	On-Chip Network (Link Bandwidth)	H/W Coherence	L1\$/core	L2\$/core	L3\$	# of DDR Controller
Teraflops Research Chip	80 (4.0 GHz)	2D Mesh (256Gb/s)	No	5KB	256KB	NA	3D stacked memory
MIT's ATAC (2008)	1000 (simulated)	2D (optical) Mesh	Yes	NA	NA	NA	NA
MIT EM² (2013)	110	2D Mesh	Simplified (1 copy)	8KB+32 KB	NA	NA	2
Single-Chip Cloud (2009)	48 (1.0 GHz)	2D Mesh (512Gb/s)	No	32KB	(256 + 8) KB 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

All adopted tile-based architecture: Cores are connected through a 2D network-on-a-chip

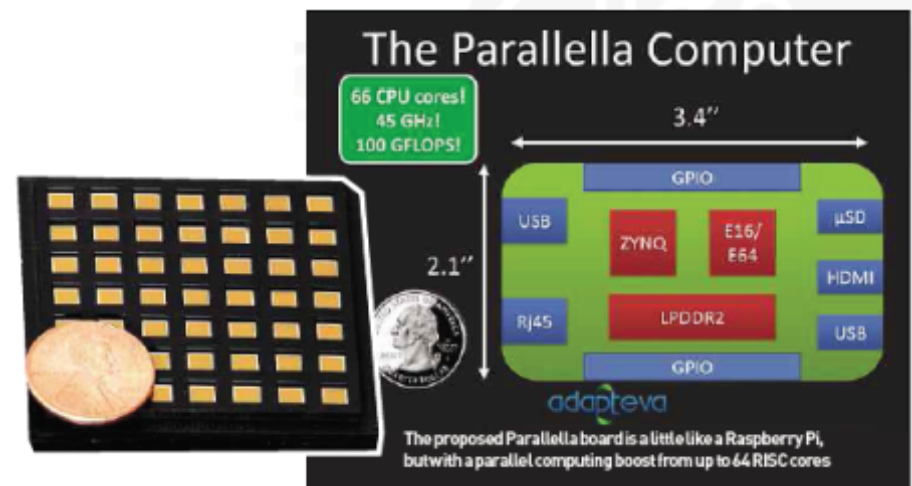
Tiled Manycore Architectures



Tiler Tile-Gx100 (100 64-bit cores)



Intel's 48-core Single-chip Cloud Computer (SCC)



Adapteva's Parallella: 64 cores for \$199



Intel Knights Landing processor (2014/15)



The Software Crisis in 1000-core Era

❑ New Challenges

1. “**Off-chip Memory Wall**” Problem
2. “**Coherency Wall**” Problem
3. “**Power Wall**” Problem

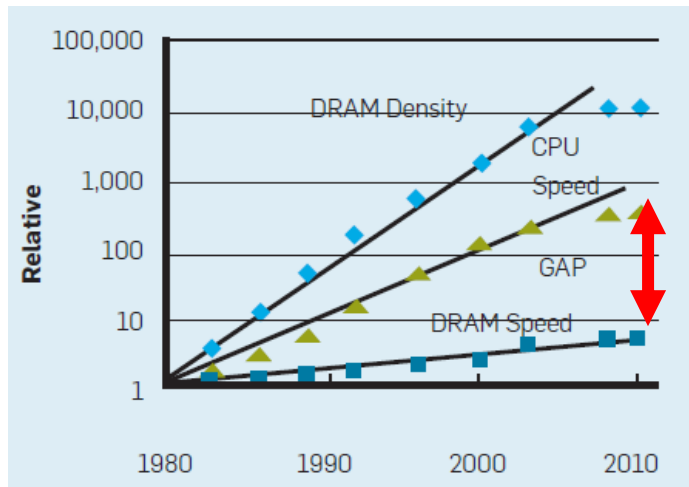
❑ Moving towards a parallelism with 1,000 cores requires a fairly **radical rethinking** of how to design system software.

• What we have done:

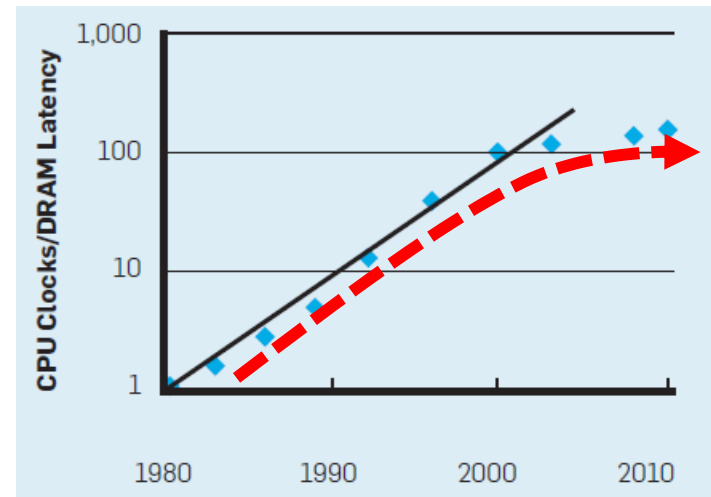
- ❑ Developed a **scalable** OS-assisted **shared virtual memory (SVM)** system on a **multikernel OS** (Barrelfish) on the Intel Single-chip Cloud Computer (SCC) which represents a likely future norm of many-core **non-cache-coherent NUMA** (NCC-NUMA) processor.
- ❑ A “zone-based” dynamic voltage and frequency scaling (DVFS) method for power saving

(1) “Off-chip Memory Wall” Problem

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



(a) Gap of DRAM Density & Speed



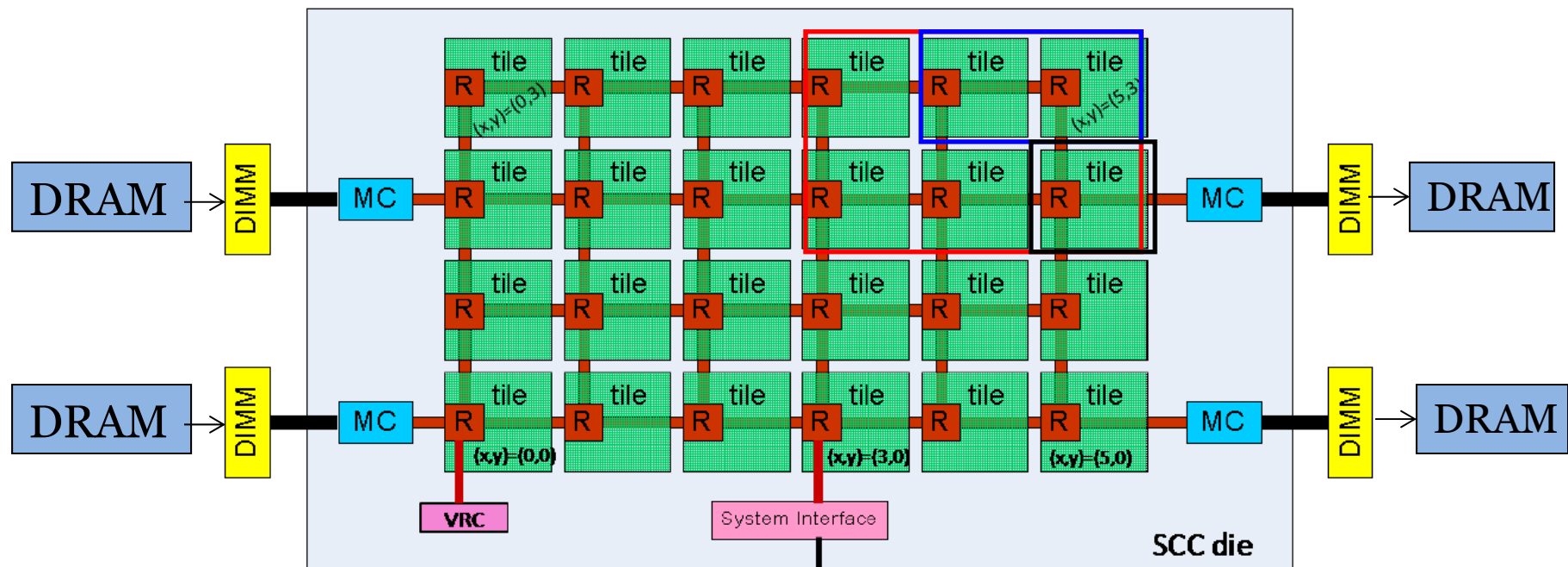
(b) DRAM Latency Not Improved

Memory density has doubled nearly every two years, while performance has improved slowly → Eliminating most of the benefits of processor improvement

Source: Shekhar Borkar, Andrew A. Chien, "The Future of Microprocessors", Communications of ACM, Vol. 54 No. 5, Pages 67-77, May 2011.

(1) “Off-chip Memory Wall” Problem

- **Smaller per-core DRAM bandwidth**
 - Intel SCC : only 4 DDR3 memory controllers → not scale with the increasing core density
 - 3D stacked memory (TSV technology) helps ?





New Design Strategies (Memory-Wall)

Data locality (working set) getting more critical!

- **Stop multitasking**
 - **Context switching breaks data locality**
 - **Space Sharing instead of Time Sharing**
- **“NoVM” : (or Space-sharing VM)**
 - **No support of VM because of weaker cores (1.0-1.5 GHz)**
 - **“Space Sharing” as we have many cores.**
- **Others**
 - **Maximize the use of on-chip memory (e.g., MPB in SCC)**
 - **Compiler or runtime techniques to improve data reuse (or increase **arithmetic intensity**) → **temporal locality** becomes more critical**



(2): “Coherency Wall” Problem

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

1. **Die space overhead**:

- Cache directory; read/write log increase
- **200%** overhead for storing the 1000 presence-bits → 128-byte directory vs. 64-byte cache line

2. **Performance overhead**:

- **20% more traffic** per miss than a system with caches but not coherence (e.g., locate other copies at hierarchical directories; issue invalidations to ALL sharers)

3. **Not always needed**:

- Only **10% of the application** memory references actually require cache coherence tracking (Nilsson, 2003)



(2): “Coherency Wall” Problem

4. Verification complexity and extensibility:

- **Multiple copies AND multiple paths through network** → require to avoid deadlock, livelock, starvation due to subtle races and many transient states.

5. Energy overhead:

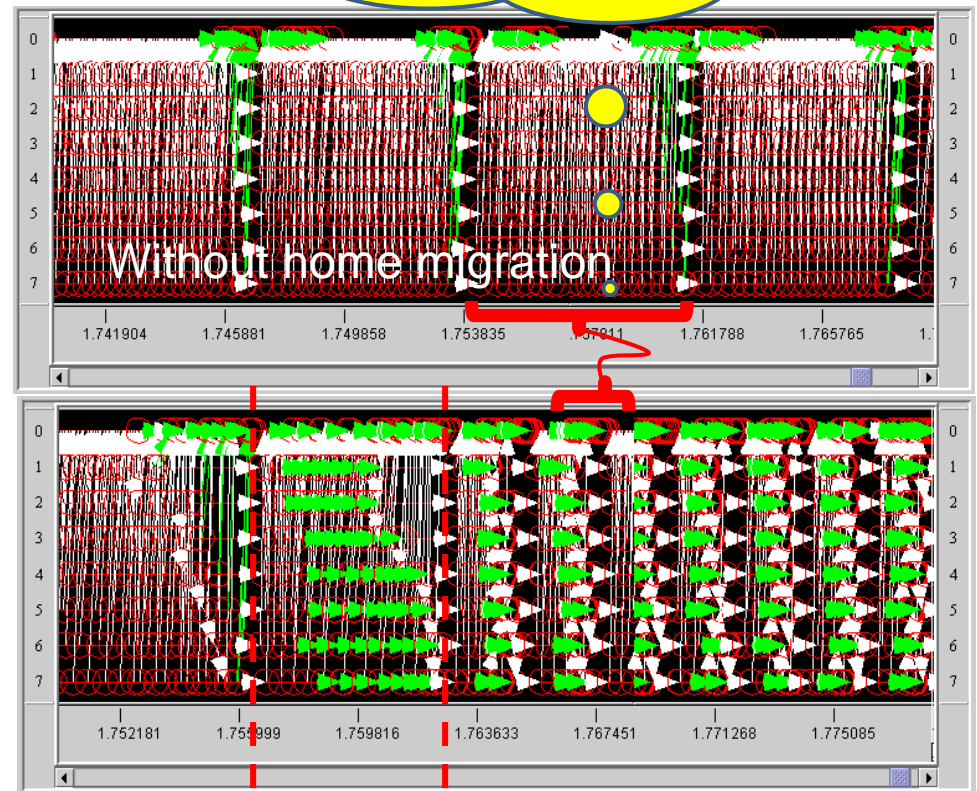
- **Unnecessary data movement and replication** consumes extra energy consumption on network and cache resources (Kurian ISCA13);
- Snoop-related cache activities can contribute up to **40%** of the total cache power (Ekman 2002, Loghi 2005)

Intel's SCC and Teraflops Research Chip decided to give up coherent caches. (History repeats itself : NCC-NUMA in 1990s: Cray T3D/ T3E)

New Design Strategies (“Coherency Wall”)

- **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)

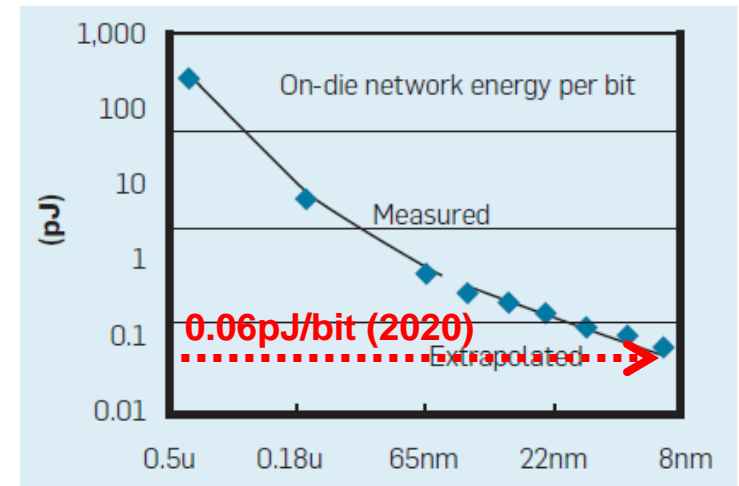
(3) “Power Wall” Problem

- Computation costs much less energy than moving data to and from the computation units

10% of the operands move over the network (10 hops at 0.06pJ/bit) → 35 watts of power → over 50% of the processor’s power budget.

- Bill Dally, Chief Scientist of nVIDIA

- 1 pJ for an integer operation
- 20 pJ for a floating-point operation
- 1000x ◦ 26 pJ to move an operand over 1mm of wire to local memory
- 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



On-die network energy consumption per bit

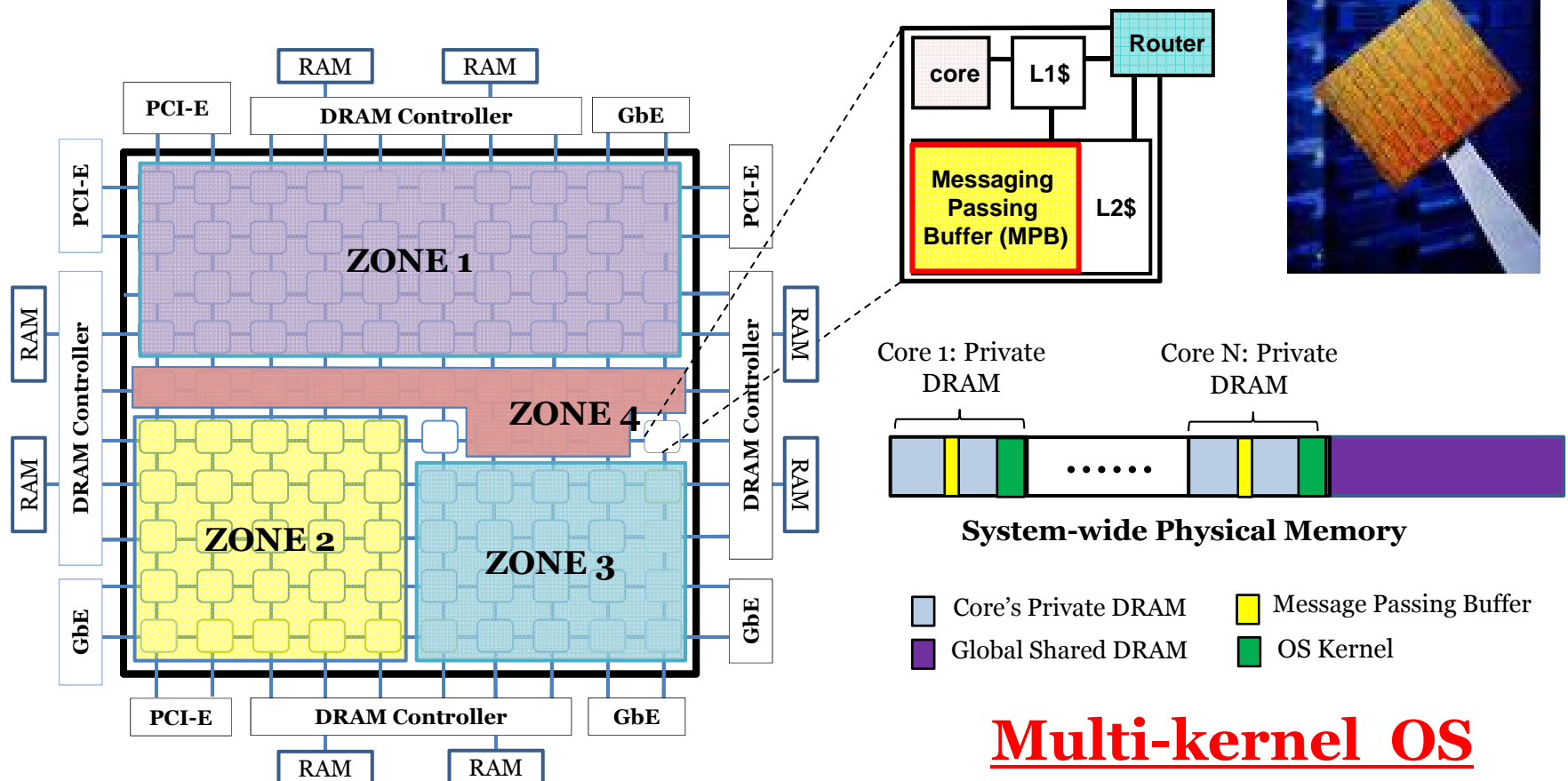
picojoule (pJ) = 10^{-12} J
nanojoule (nJ) = 10^{-9} J

New Design Strategies (“Power Wall”)

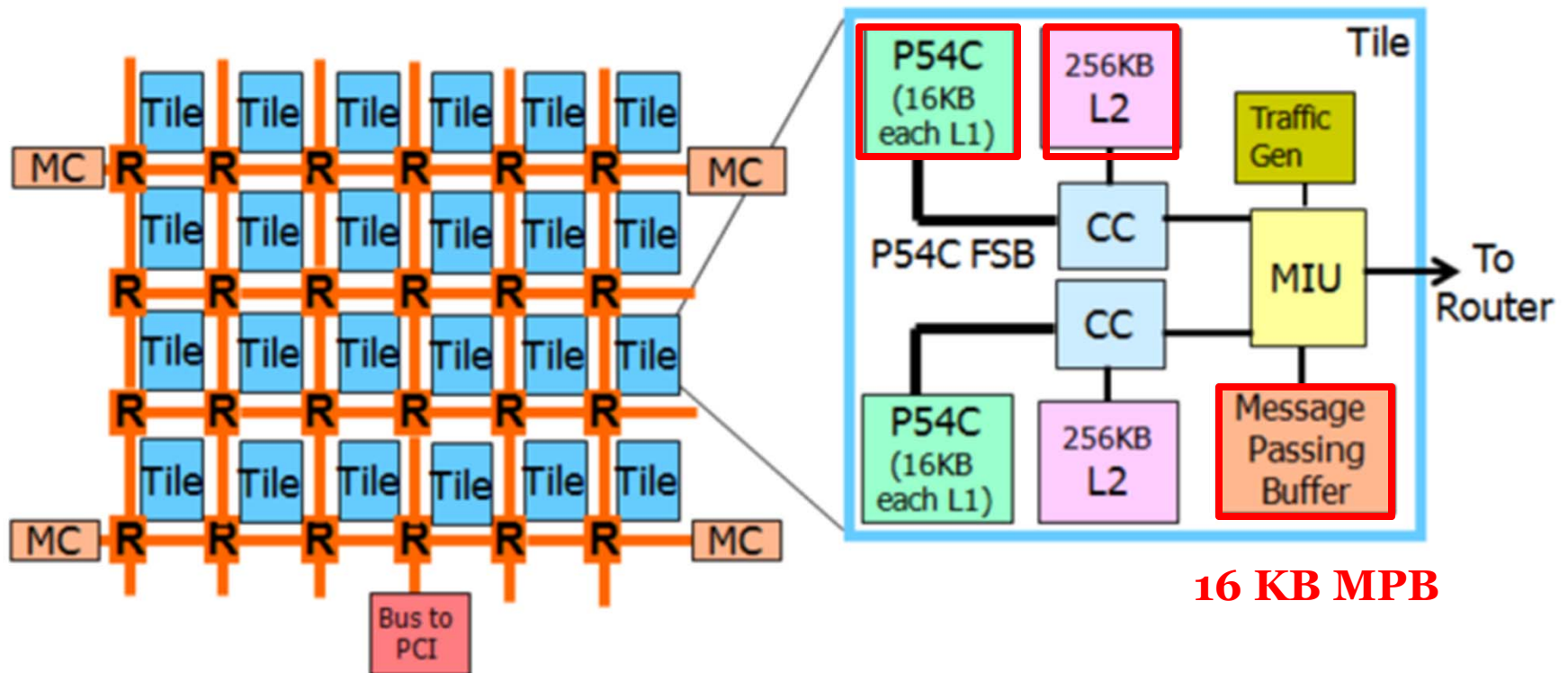
- **Stop moving so much data around**
 - Data Locality (Working Set) still critical!
 - **Distance-aware cache placement and home migration**
 - Migrating “code & state” instead of data
 - Relatively easier in shared memory manycore systems.
 - MIT EM² support hardware thread migration
 - Adopt **multikernel** operating system (e.g., **Barrelfish**)
 - **Message passing among kernels to avoid unnecessary NoC traffic**
- **Barrelfish** : “Compact message cheaper than many cache lines-- even on a cache-coherent machine.”



Crocodiles: Cloud Runtime with Object Coherence On Dynamic tILES for future 1000-core tiled processors” (HK GRF: 01/2013-12/2015)



Current Platform: INTEL 48-core SCC processor

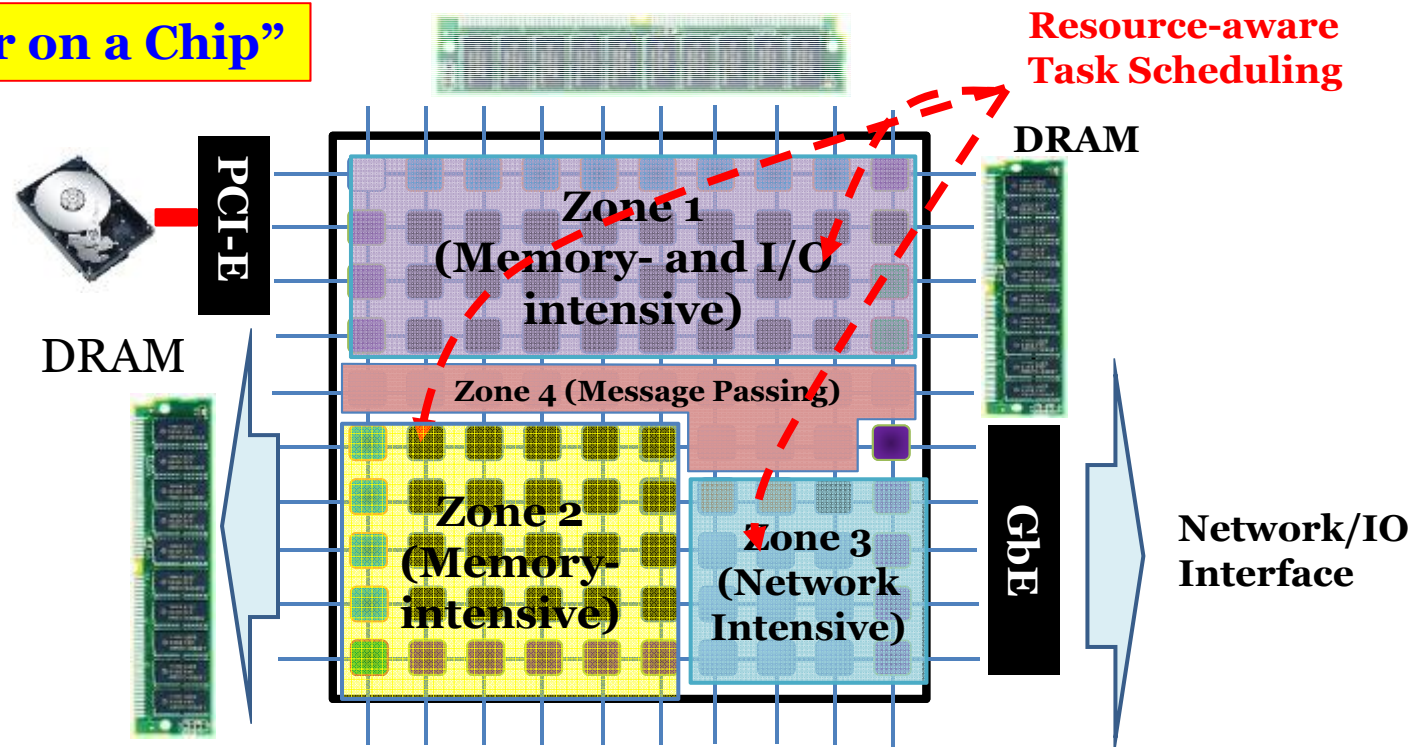


Routers (**R**), memory controllers (**MC**), mesh interface units (**MIU**), cache controllers (**CC**), front side bus (**FSB**). 45 nm CMOS technology, 1.3 billion transistors.

(1) Cloud-on-Chip (CoC) Paradigm

- Condensing a data center into a single many-core chip
 - “Zoning” (Spatial Partition)
 - Multiple isolated zones → Better performance isolation
 - Mimic multitenant Cloud computing without **time-sharing VMs** → avoid context switching

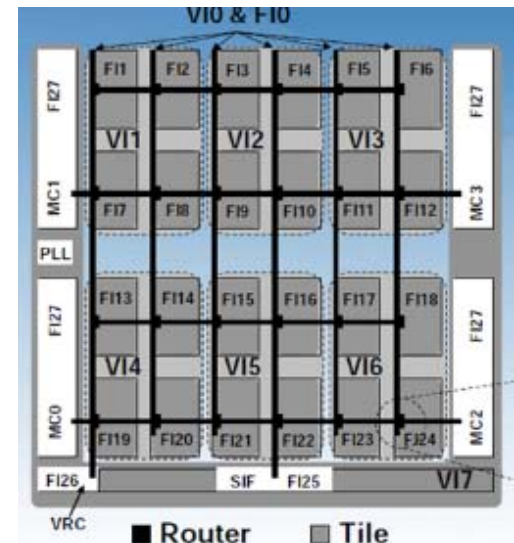
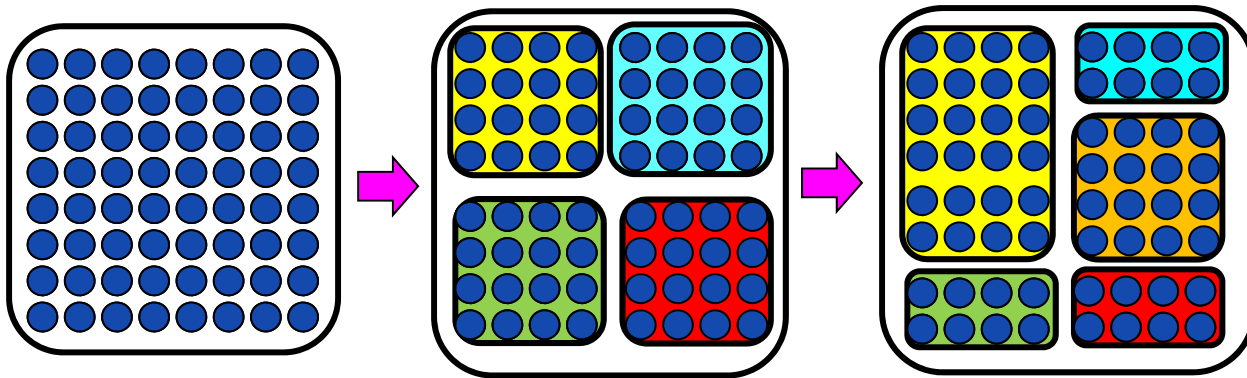
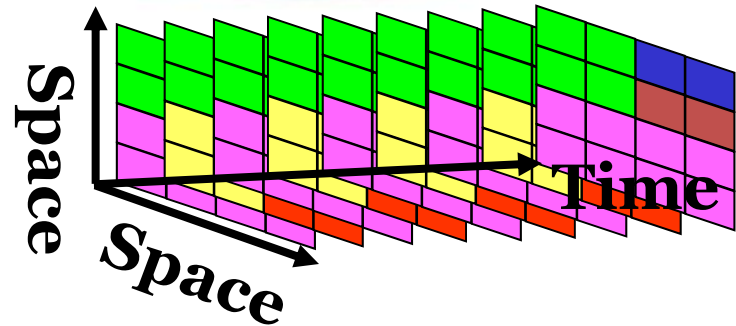
“Data Center on a Chip”



(2) Dynamic Zoning

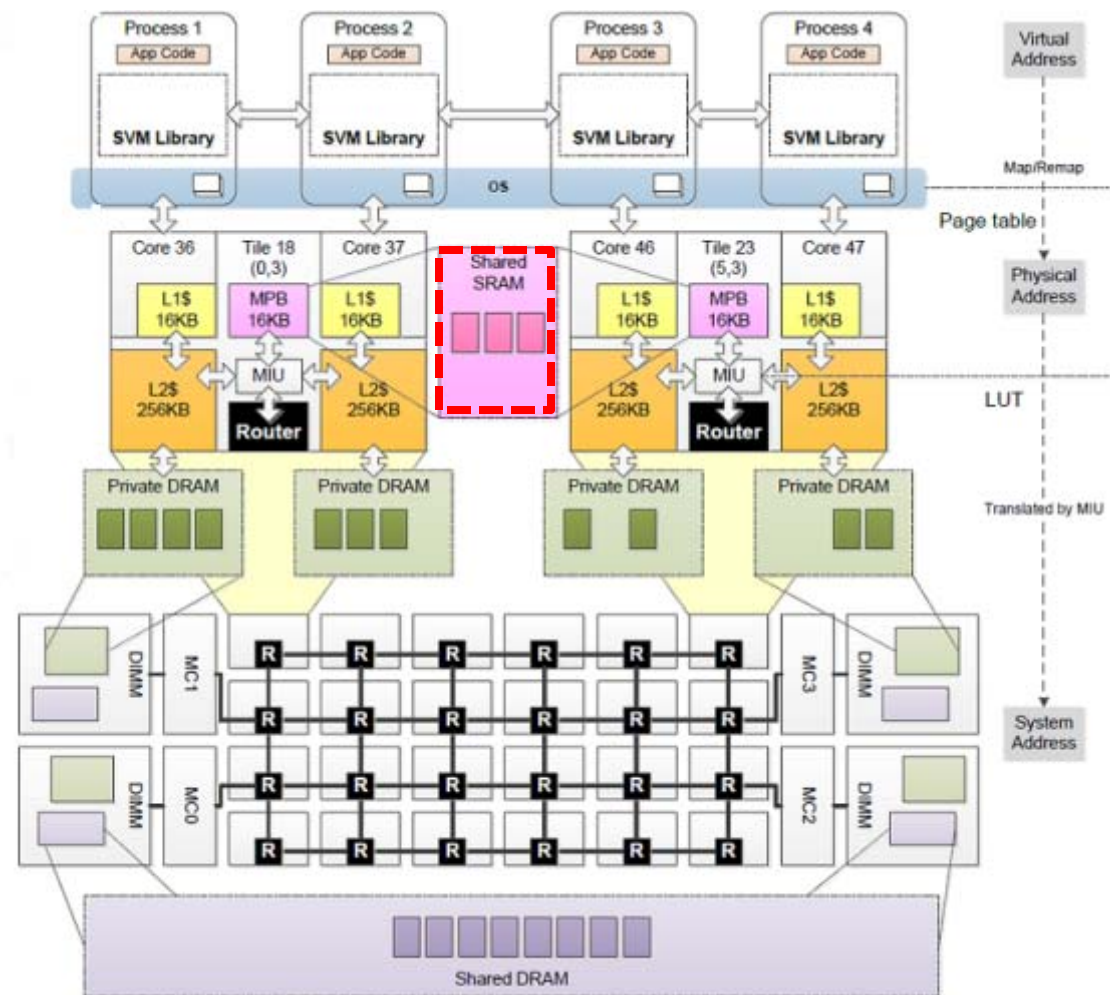
- **“Dynamic Zone Scaling”**:

- Partitioning varies over time.
- On-demand scaling of resources (e.g., # of cores, DRAM,..) for each zone.
- Fit well with the domain-based power management (e.g., Intel SCC)



(3) Software-Managed Cache Coherence: JumpSCC

- **Leverage programmable on-chip memory** (e.g., MPB on Intel SCC)
- **Scope Consistency (ScC)**: minimizing on-chip network and off-chip DRAM traffic
 - Existing systems using ScC: **Jiajia** (1998), **Nautilus** (1998), **HKU JUMP** (1999), **HKU LOTS** (2004), Godson-T (2009).



Intel SCC

JumpSCC: Hybrid Modes of Memory Sharing

- Data can be shared in a different way.

- Selectable on per-page basis

- **Two modes available:**

1. **Shared Physical Memory (SPM)**

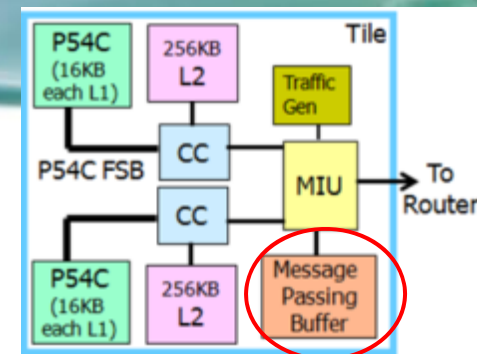
- Intel SMC's way
- All data kept as golden pages in shared DRAM
- Set MPBT to **bypass L2 cache**.
- **Use write-through.**
- Use **CL1INVMB** and **flush WCB** to ensure consistency

2. **Distributed Shared Memory (DSM)**

- For each user core, it will copy the golden page to a cached copy in private DRAM upon page faults (due to memory protected).
- Use **twin-and-diff** technique to avoid false sharing between multiple writers.

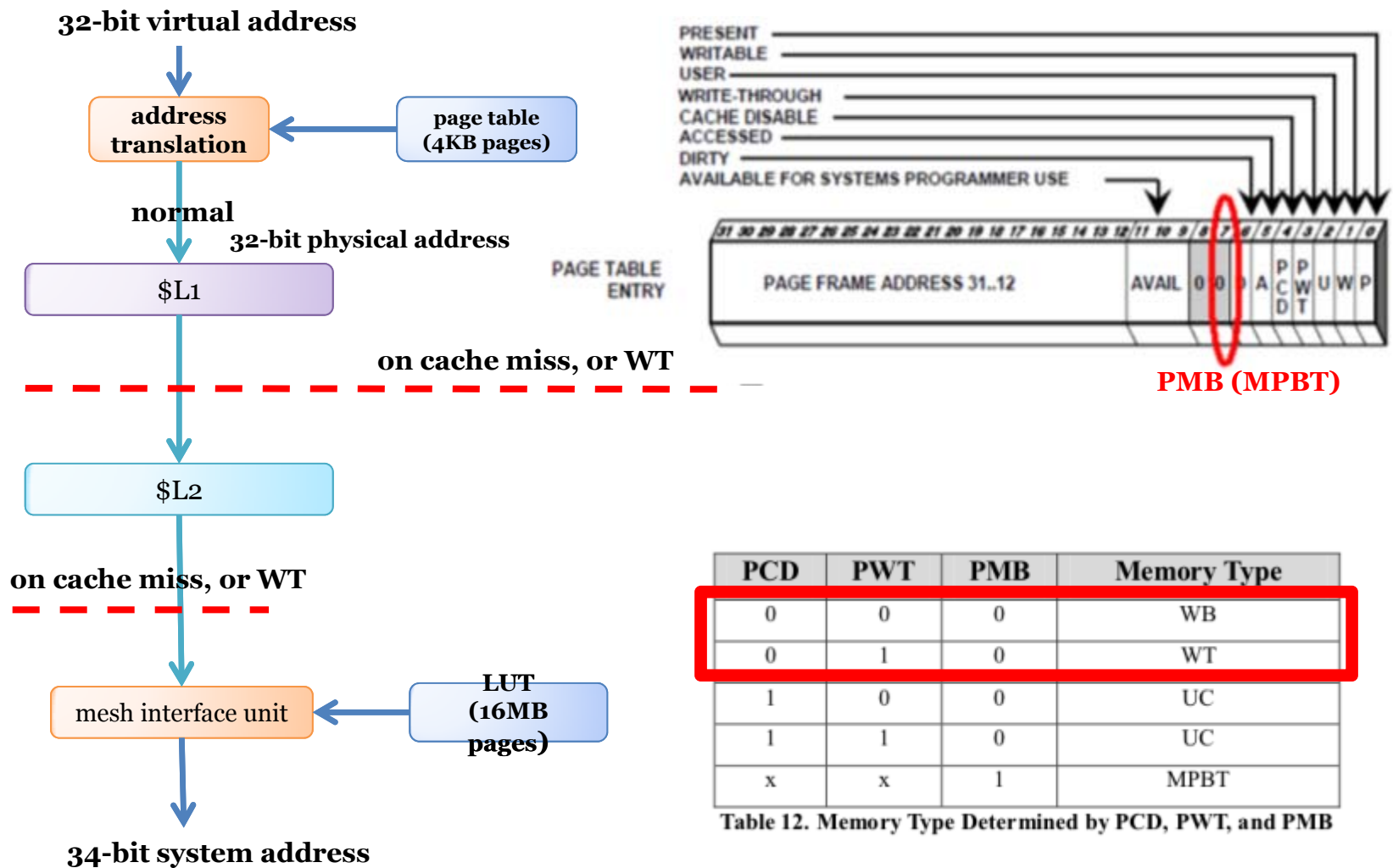


A New Memory Type

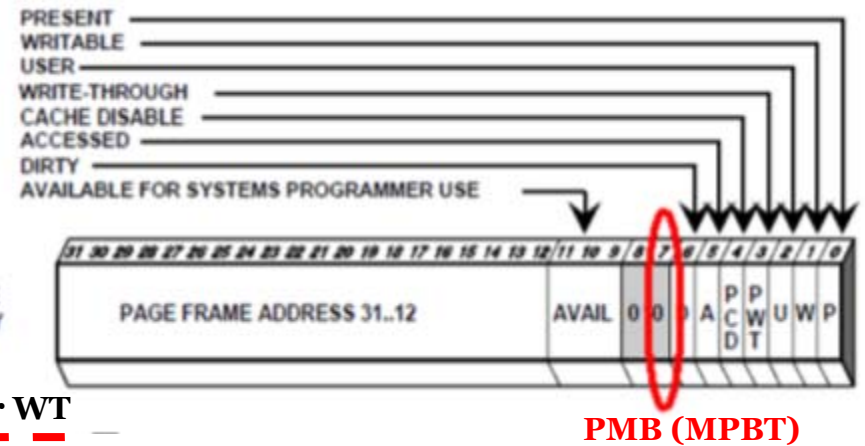
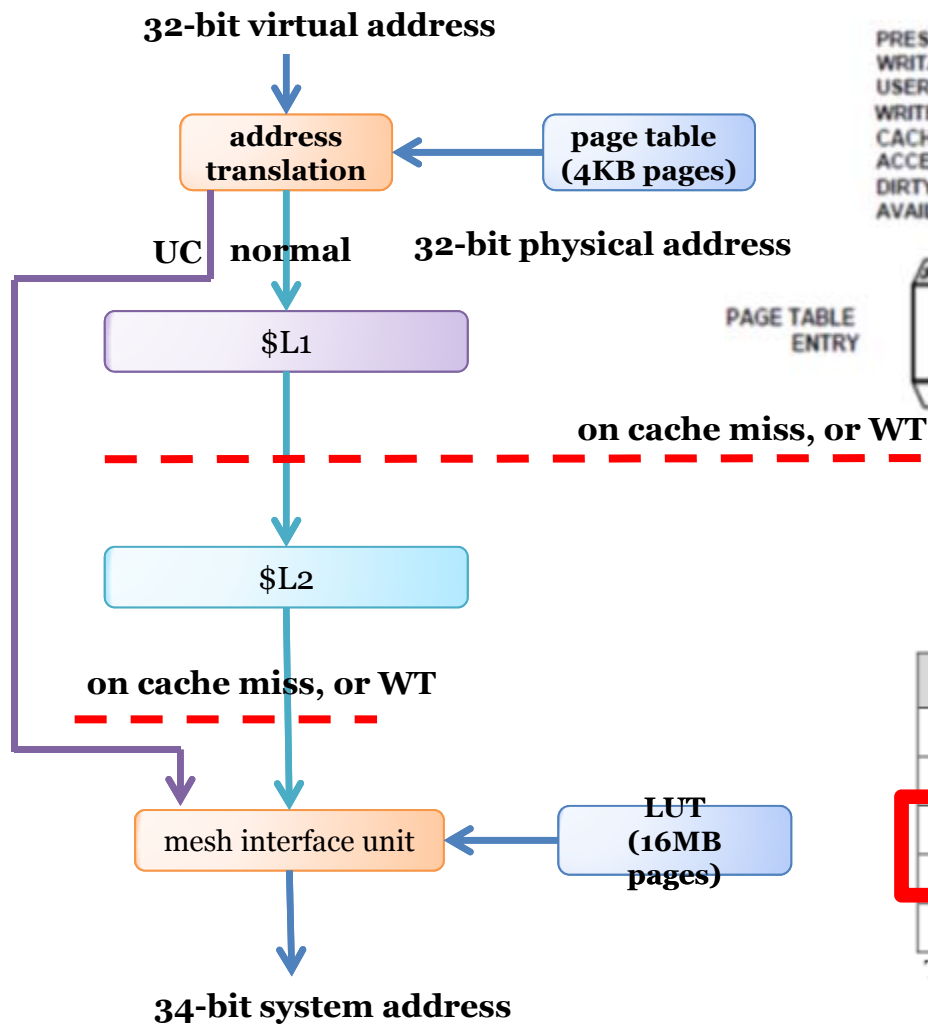


- **Message Passing Buffer Type (MPBT)**
- **MPBT (or PMB)** is a bit in page table entry
 - We can map a chunk in off-chip DRAM as MPBT
 - We can map a chunk in on-chip MPB as non-MPBT
 - We can modify it at runtime
- MPBT tag only takes effect upon
 - L1\$ write miss (where to write: **WCB or L2\$**)
 - L1\$ read miss (where to read: MEM or L2\$)
 - **CL1INVMB** (invalidate MPBT-tagged lines in L1\$)

SCC Cache Behavior - Normal



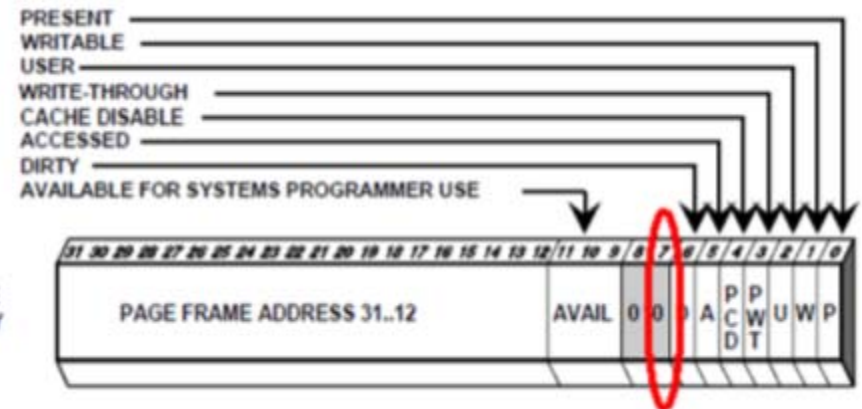
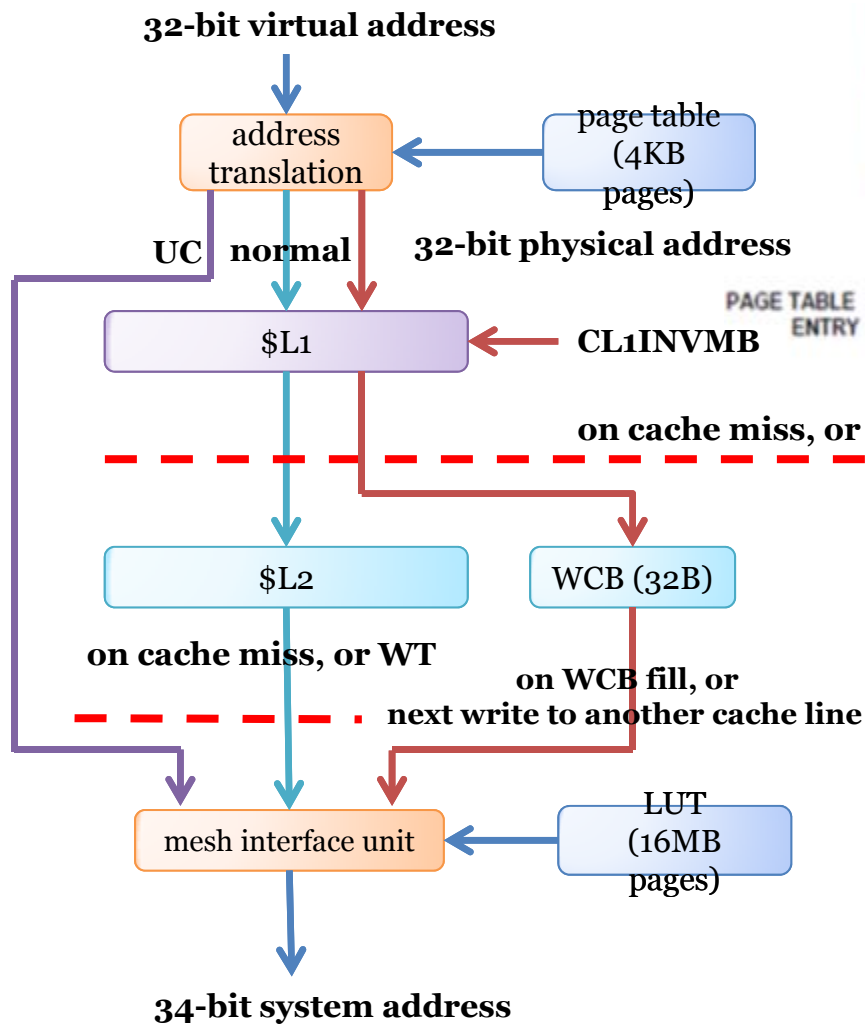
SCC Cache Behavior - UnCached



PCD	PWT	PMB	Memory Type
0	0	0	WB
0	1	0	WT
1	0	0	UC
1	1	0	UC
x	x	1	MPBT

Table 12. Memory Type Determined by PCD, PWT, and PMB

SCC Cache Behavior - MPBT



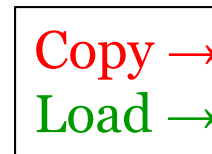
PCD	PWT	PMB	Memory Type
0	0	0	WB
0	1	0	WT
1	0	0	UC
1	1	0	UC
x	x	1	MPBT

Table 12. Memory Type Determined by PCD, PWT, and PMB

Alleviate the Memory Wall Problem

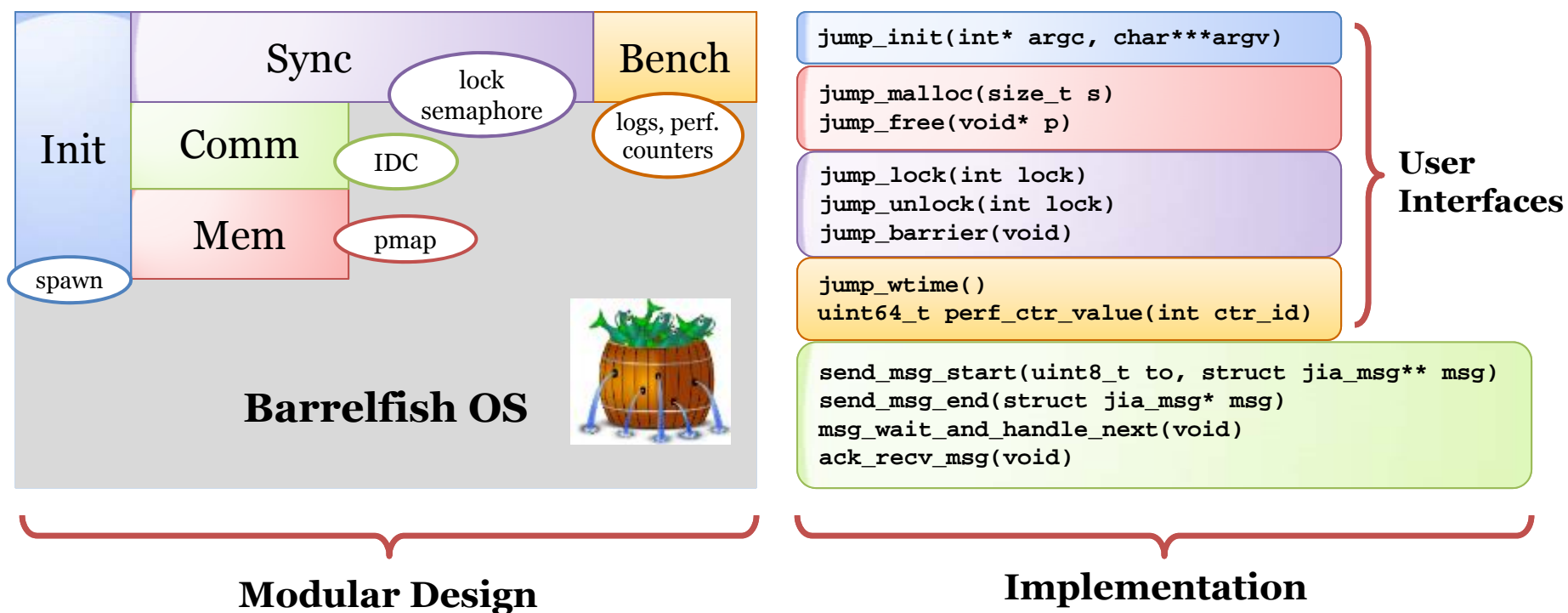
- Minimize DRAM access by exploiting the MPB space to store data (programmer-hinted or profiler-guided)
- Now we have two datapaths:
 1. DRAM \rightarrow L2 \rightarrow L1
 2. DRAM \rightarrow MPB \rightarrow L1
- Example uses of MPB:
 - Used to reduce **cache pollution**
 - For sequential data access (data without reuse), manually allocate buffer in MPB and copy the data from off-die DRAM to MPB; then L2 cache won't evict any cache lines and keep the hottest data set.
 - Used to cache data of “warm temperature”
 - **Warm data** (long reuse distance) is secondary to **hot data** (short one);
 - **Data of reuse distance > L2 capacity** can still be read within on-chip speed if read from MPB rather than from DRAM.

Key:

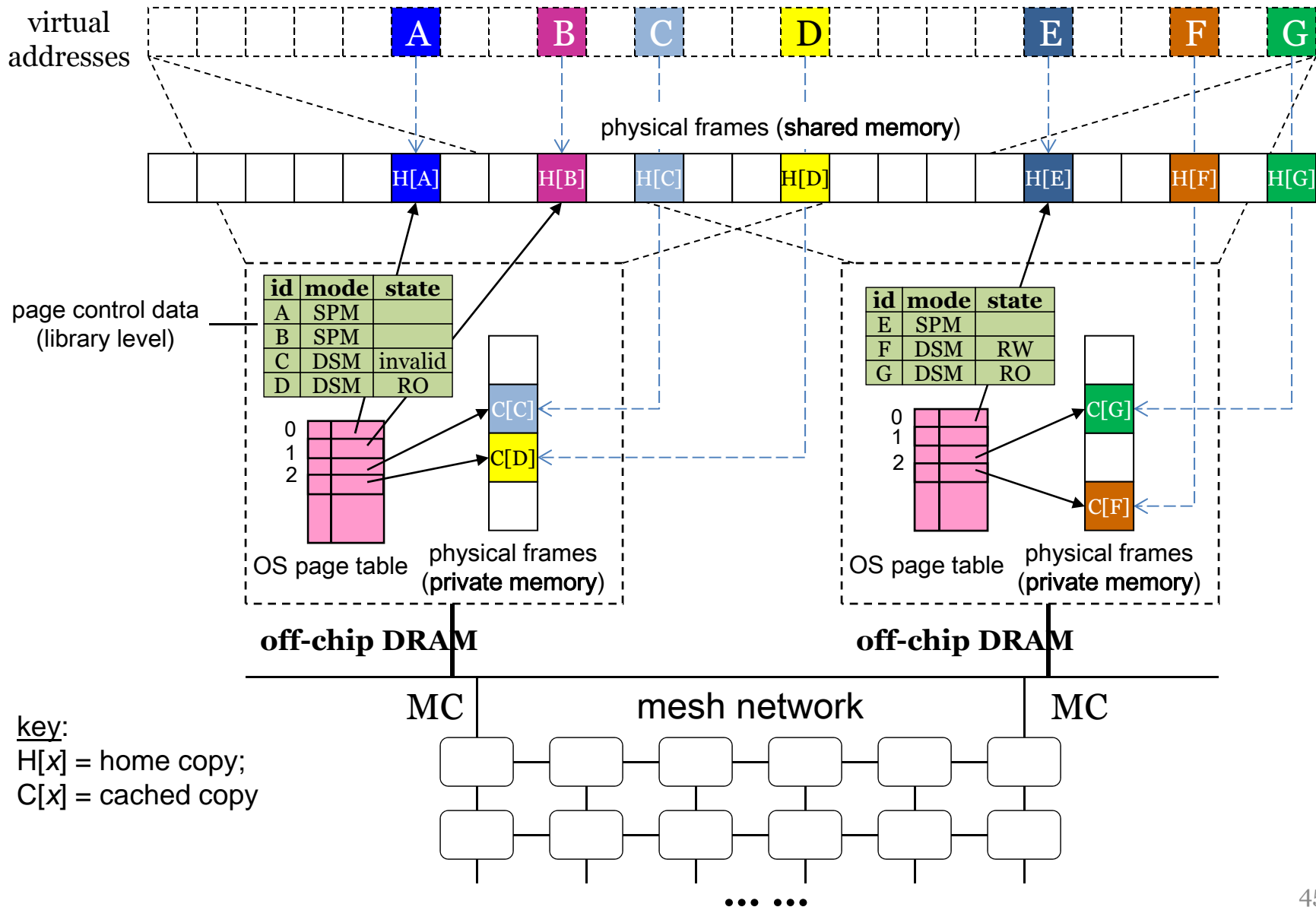


JumpSCC: System Design

- Built on top of Barrelfish OS as a user library
- Resemble traditional shared-memory programming
- Just a different set of malloc, lock, unlock (and barrier)



Global virtual address space (programmer's view)





Page Remapping for Data Locality

- **Programmer hint API:**

- JIA_ATTR_FREQ_ACCESS: frequently accessed (read-write)
- JIA_ATTR_READONLY: frequently accessed (read-only)
- JIA_ATTR_SINGLE_WRITER: single writer
- JIA_ATTR_NOCACHE: non-cacheable (avoid cache pollution)

Exploit L2 cache
(our protocol
ensures no
consistency issues)

- **System handling:**

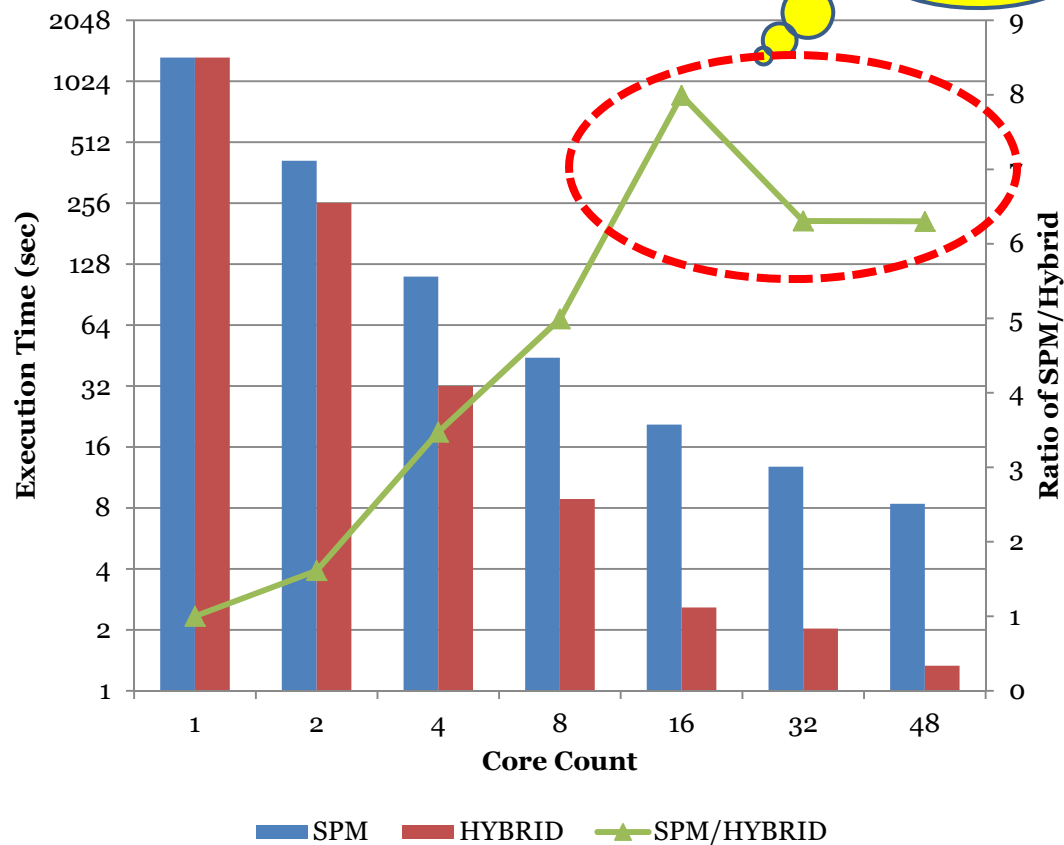
- **JIA_ATTR_FREQ_ACCESS**: copy golden page to **private DRAM**
- **JIA_ATTR_READONLY**: set to **non-MPBT (make use of large L2 \$)**
- **JIA_ATTR_SINGLE_WRITER**:
 - The writer sets to non-MPBT R/W; readers set to MPBT R/O.
 - At sync pts, the writer flushes L2 cache by reading 256 KB data.
- **JIA_ATTR_NOCACHE**: set PTE to non-cacheable



Performance Benchmarking

- **MalStone Benchmark**
 - Analysis of “drive-by exploits” in web site log files
- **Graph 500 Benchmark**
 - Generation, compression and breadth-first search of large graphs
- **Sorting (bucket-sort kernel)**
- **Miscellaneous compute kernels (skipped)**

Malstone

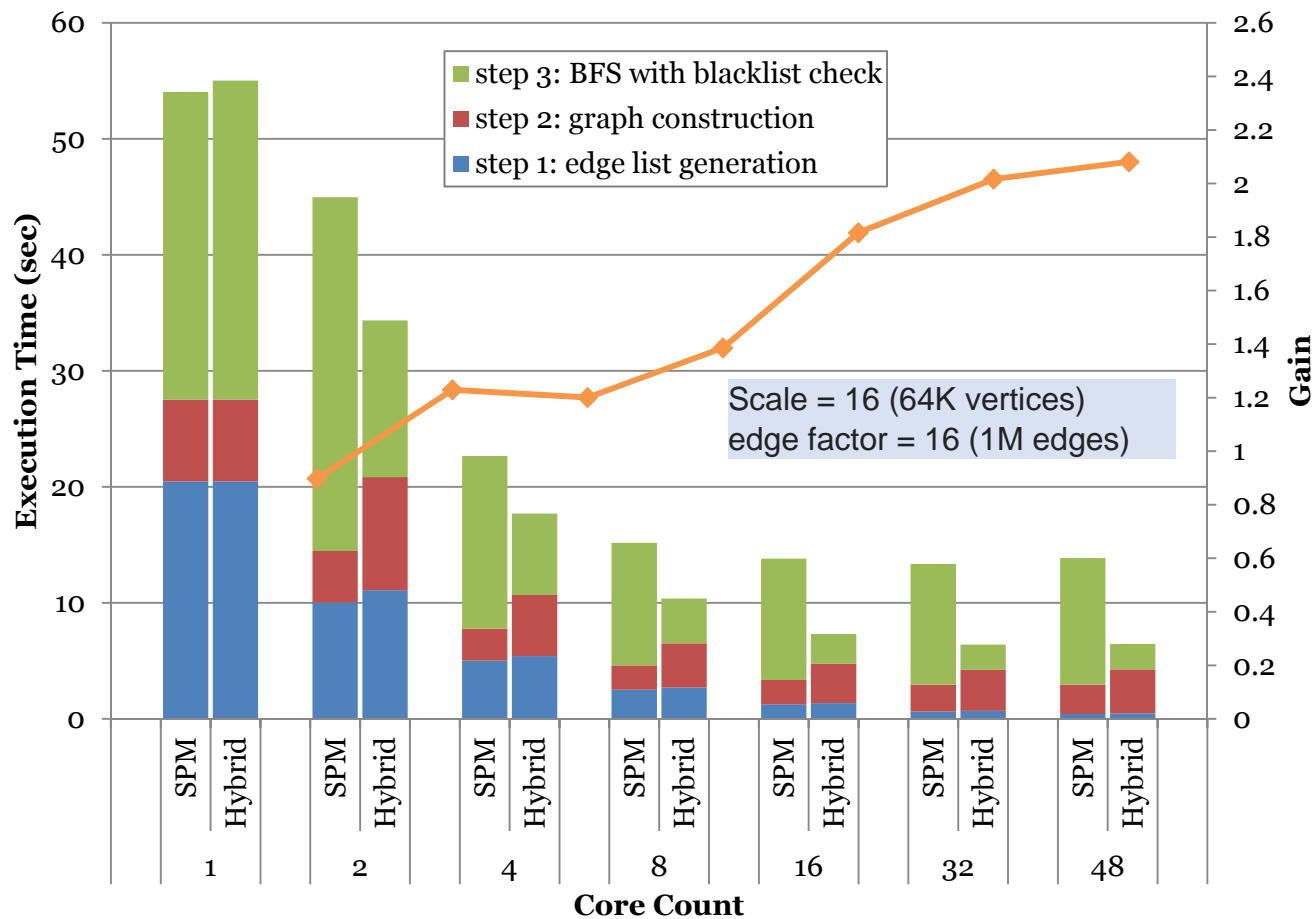


Hybrid mode gives 6x to 8x better performance than SPM mode.

- With the hybrid memory model, the pages mapped as private page mode (**MPBT bit off**) can exploit L2 cache and hence have augmented the cache effect enormously.

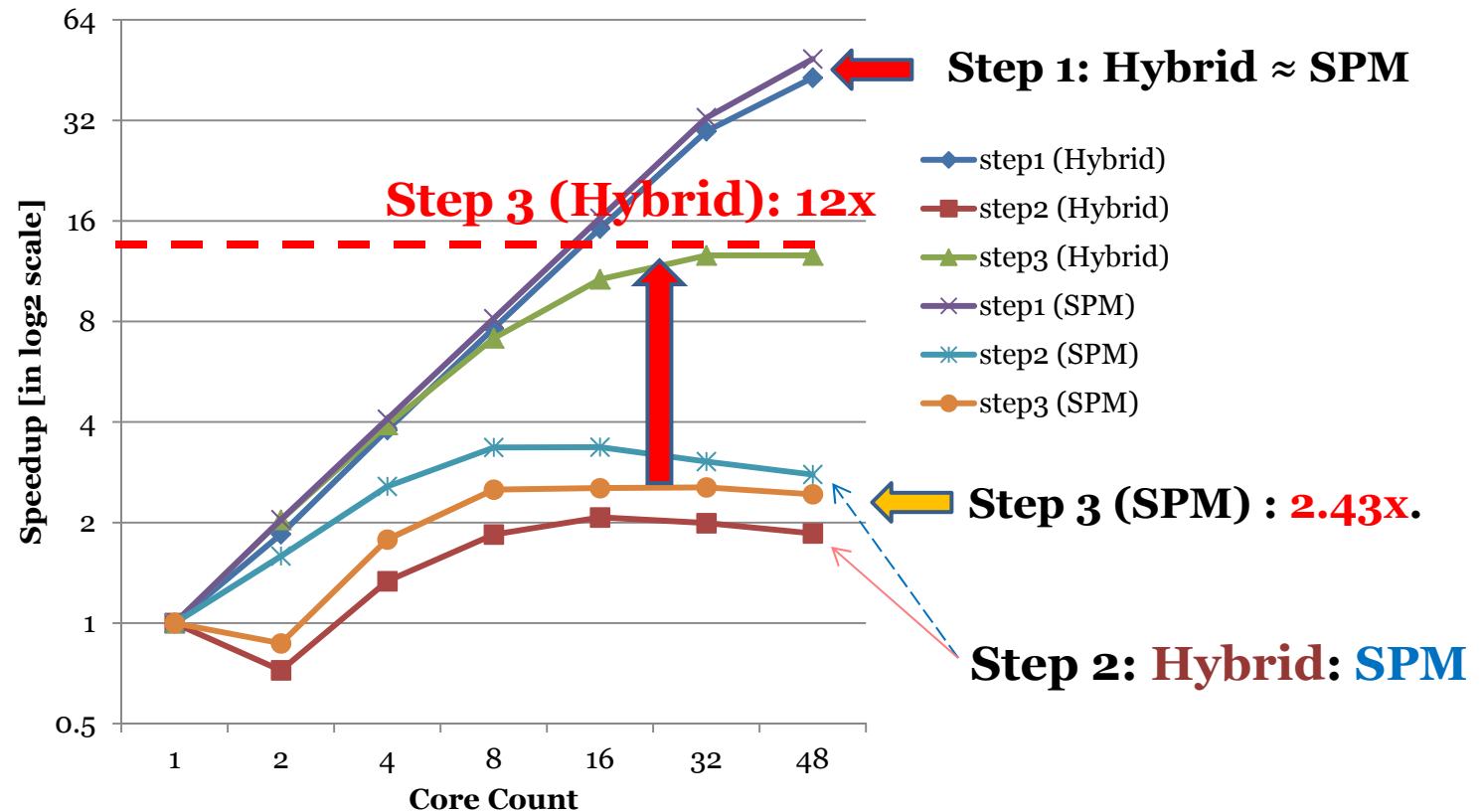
Graph 500

- On 48 cores, can reach **2.1x** gain in hybrid mode over SPM;
 - the BFS loop with certain data reuse (blacklist checks) contributes that.

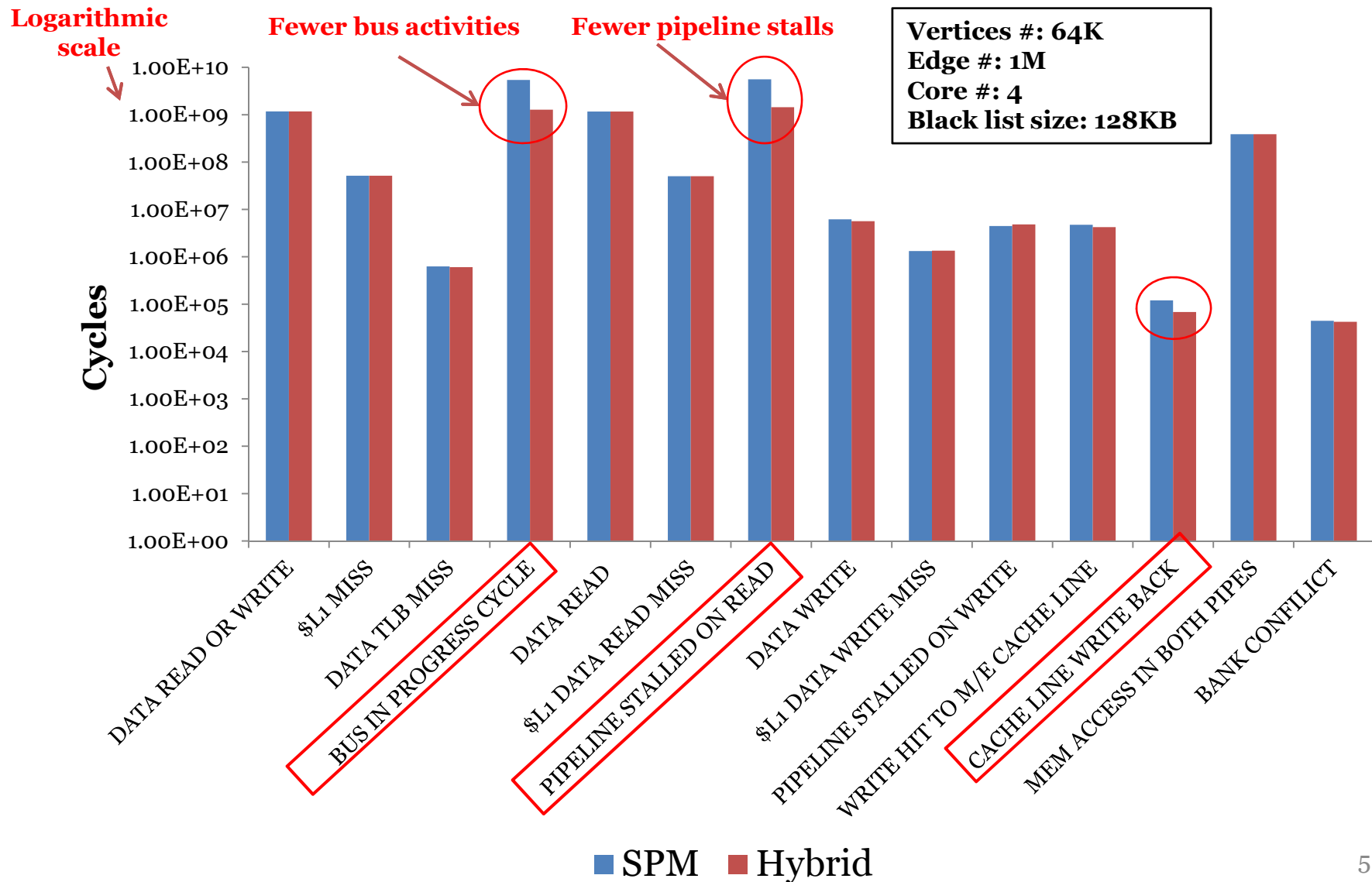


Graph 500: Scalability Analysis

- Step 3 (BFS) @ hybrid mode achieved **12x speedup**.
- Step 3 @ SPM only **2.43x**.

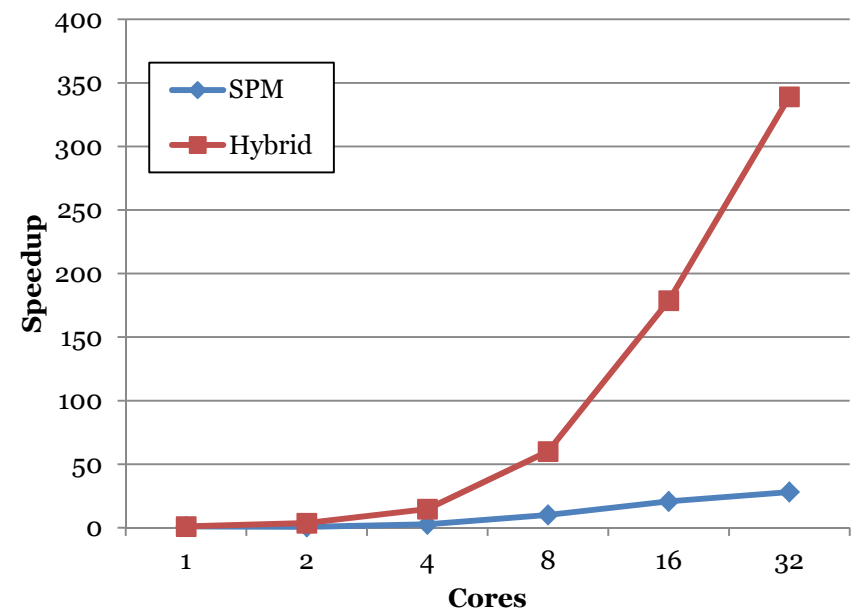
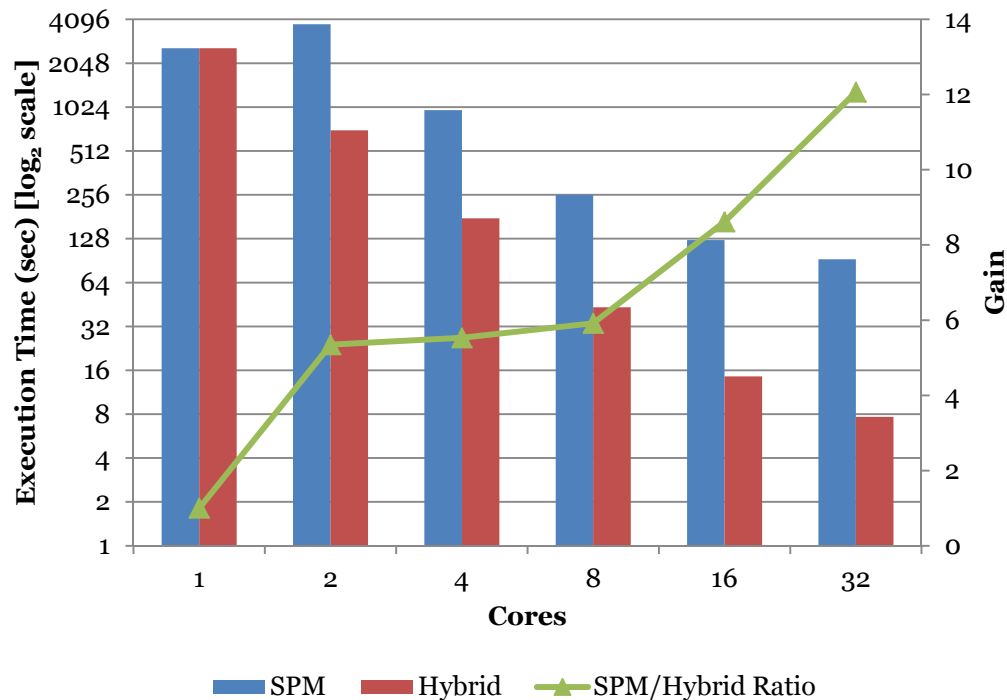


Performance Counter Analysis

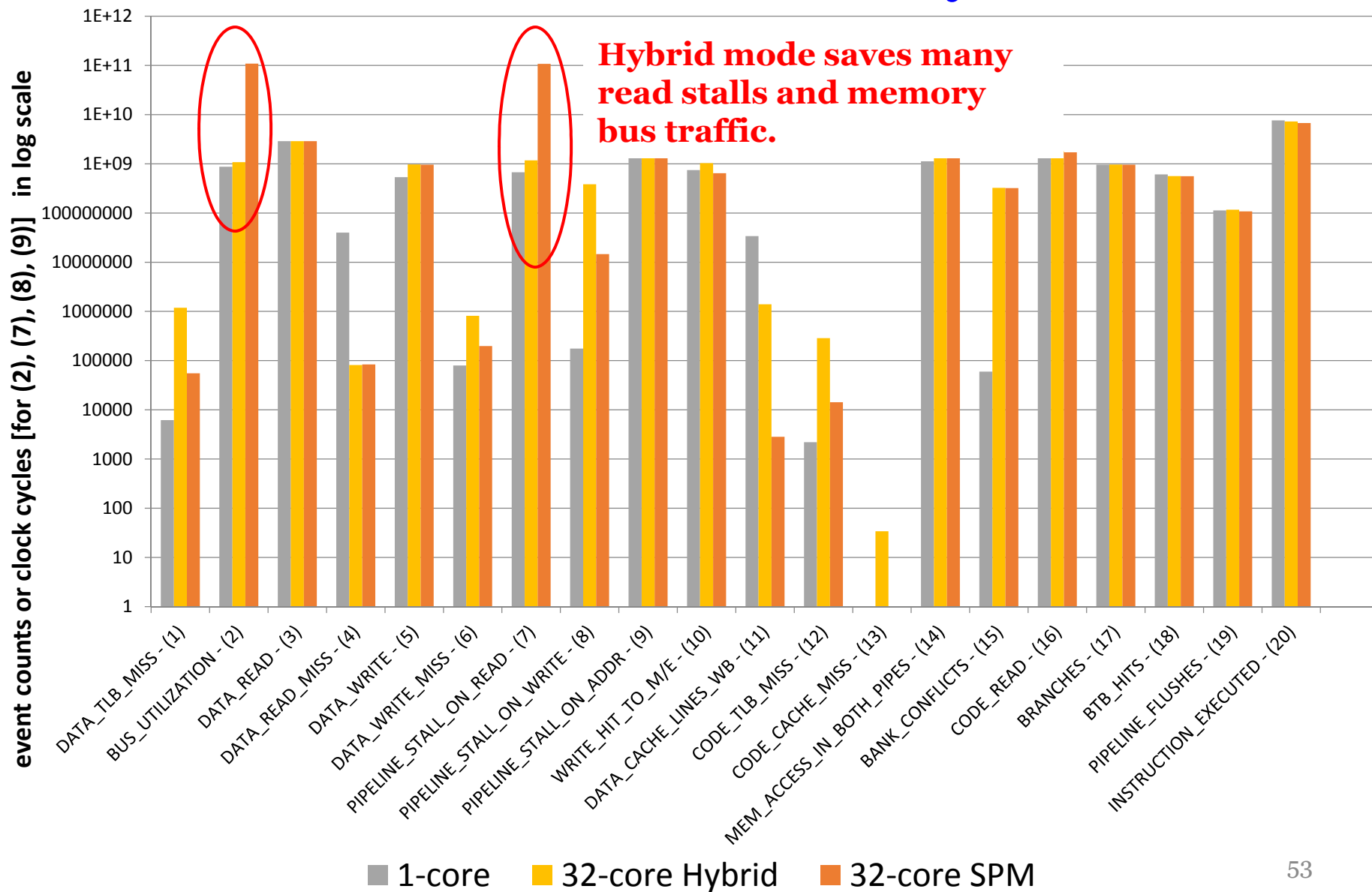


Bucket Sort

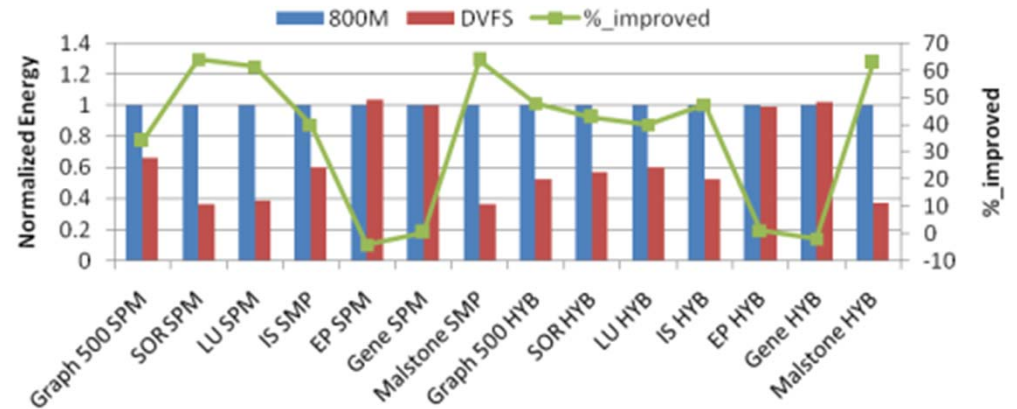
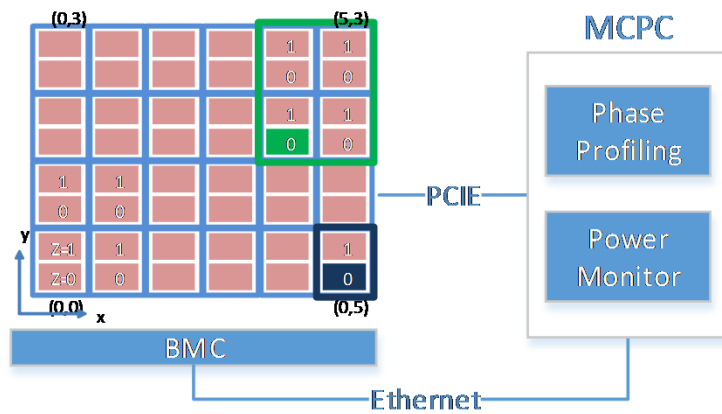
- **12x** better in hybrid mode than SPM alone
- Superlinear speedup observed in hybrid mode
 - Augmented cache effect since L2\$ not bypassed



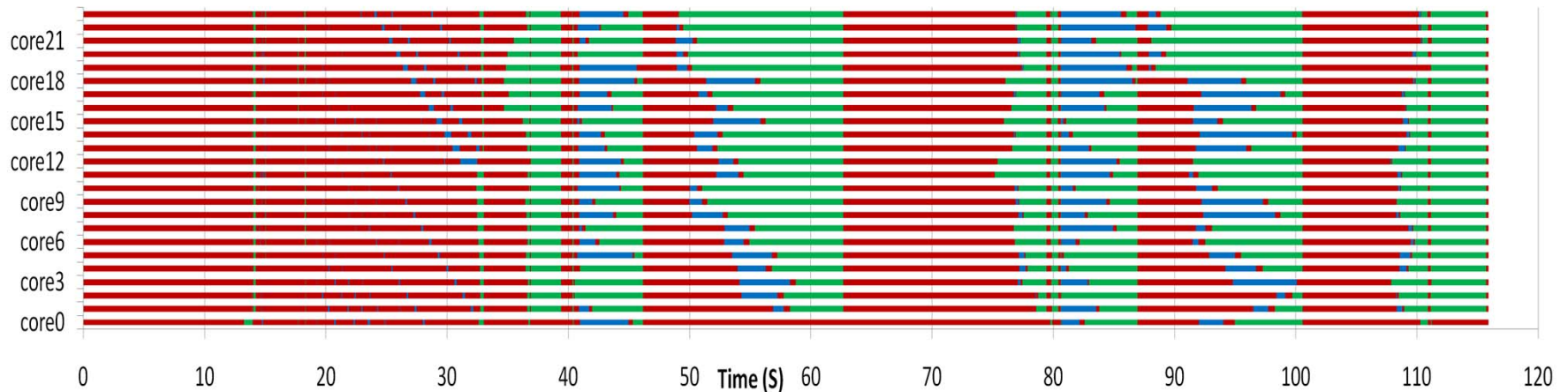
Performance Counter Analysis



Zone-based DVFS Method for Power Saving



Energy normalized to values of 800M
 The average energy saving is **35.68%**, and the average EDP reduction is **21.42%**.



“Latency-aware Dynamic Voltage and Frequency Scaling on Many-core Architecture for Data-intensive Applications”, CLOUDCOM-ASIA 2014



Significance of JumpSCC

- **The first SVM for the Barrelfish OS**
- **Novel software CC system design:**
 - **Exploit both private memory and shared memory efficiently (selectively)**
 - **Support two “coherence modes” (or memory models) concurrently on a per-page basis**
 - **Harness non-coherent L2 caches while others can’t**
- **Performance is 12% to 12 times better than Intel SMC.**
- **Three patents claimed:**
 1. A hybrid shared virtual memory system with adaptive page remapping for non-cache-coherent many-core processors
 2. A proactive scope consistency protocol tailored to many-core tiled processors with programmable on-chip buffers
 3. A location-aware k-way hash-based distributed on-chip page directory for tiled CMPs



Conclusion

- **GHz game is over → Go for Manycore**
 - Processors parallelism is primary method of performance improvement
- **Coherency-, memory- and power-wall challenges in the 1000-core era are discussed.**
- **Software-managed Cache Coherence Support:**
 - Transfer the burden of cache coherence from hardware to software, while preserving hardware support for remote cache accesses.
 - On-chip programmable memory like MPB enable customizable or programmable on-chip activities.
- **Power efficiency is the key challenge (flops/watt) →**
“DON’T MOVE THE DATA!”



Thanks!

For more information:

C.L. Wang's webpage:

<http://www.cs.hku.hk/~clwang/>

