

# System Software for Big Data Computing

Cho-Li Wang

The University of Hong Kong



# 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

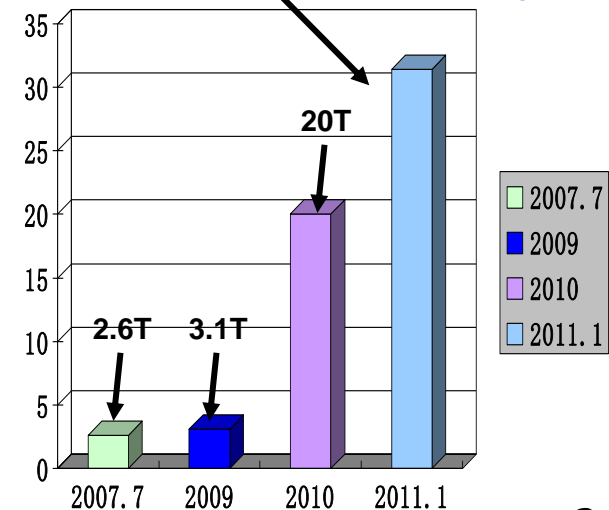


CS Gideon-II & CC MDRP Clusters



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


**31.45TFlops (X12 in 3.5 years)**




# 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)




  
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.

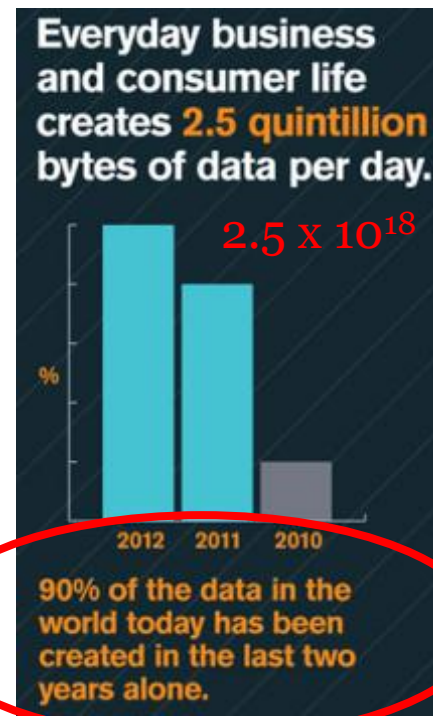
  
Zynga processes 1 petabyte of content for players every day; a volume of data that is unmatched in the social game industry.

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.**



# Our Research

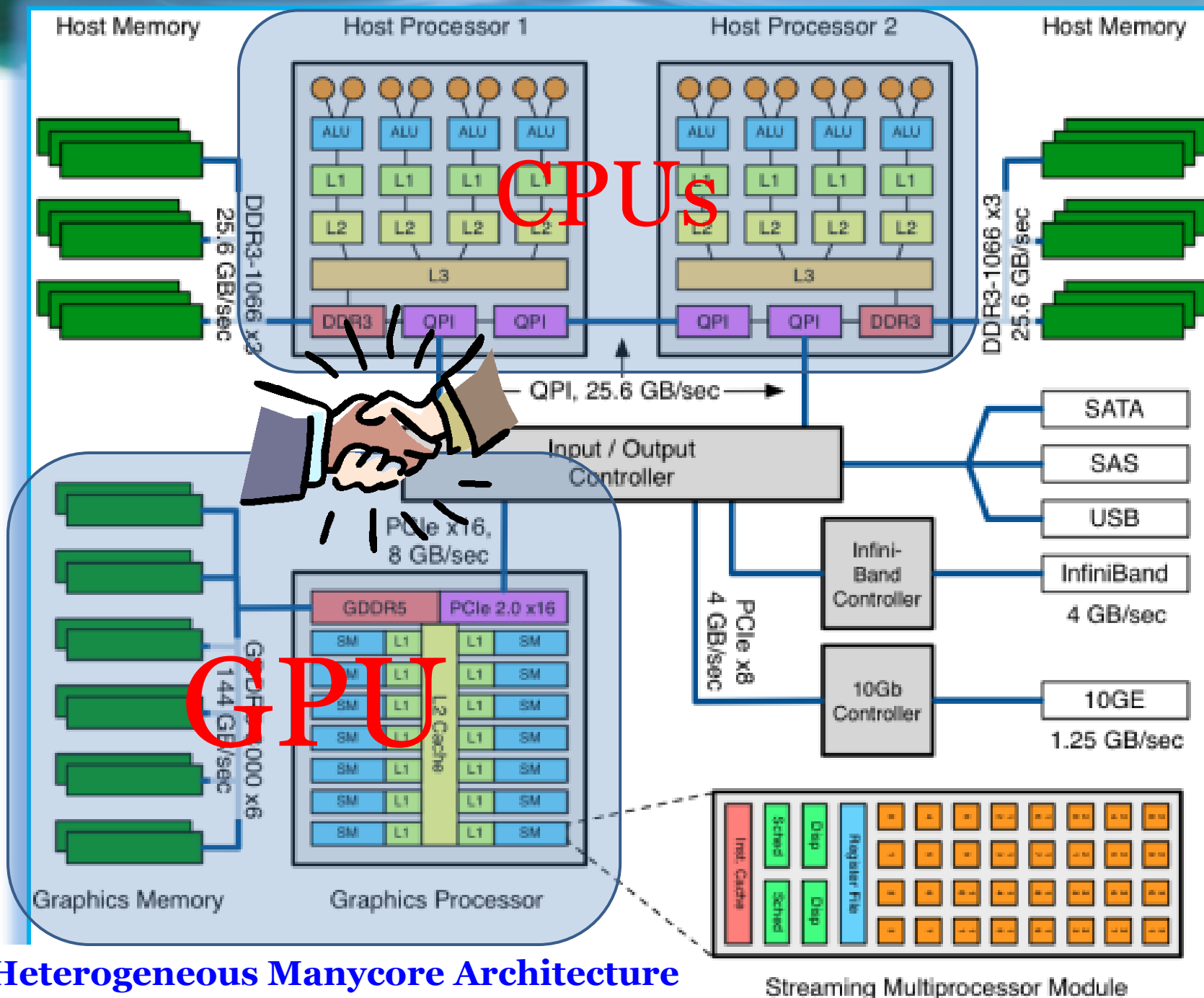
- **Heterogeneous Manycore Computing (CPUs+ GUPs)**
- **Big Data Computing on Future Manycore Chips**
- ~~**Multi-granularity Computation Migration**~~





## **(1) Heterogeneous Manycore Computing (CPUs+ GUPs)**

**JAPONICA : Java with Auto-  
Parallelization ON GraphIcs  
Coprocessing Architecture**



## Heterogeneous Manycore Architecture

# New GPU & Coprocessors

Vendor	Model	Launch Date	Fab. (nm)	#Accelerator Cores (Max.)	GPU Clock (MHz)	TDP (watts)	Memory	Bandwidth (GB/s)	Programming Model	Remarks
Intel	<b>Sandy Bridge</b>	2011Q1	32	12 HD graphics <b>3000 EUs</b> (8 threads/EU)	850 – 1350	95	L3: 8MB Sys mem (DDR3)	21	OpenCL	Bandwidth is system DDR3 memory bandwidth
	<b>Ivy Bridge</b>	2012Q2	22	16 HD graphics <b>4000 EUs</b> (8 threads/EU)	650 – 1150	77	L3: 8MB Sys mem (DDR3)	25.6		
	<b>Xeon Phi</b>	2012H2	22	<b>60</b> x86 cores (with a 512-bit vector unit)	600-1100	300	8GB GDDR5	<b>320</b>	OpenMP#, OpenCL*, OpenACC%	Less sensitive to branch divergent workloads
AMD	<b>Brazos 2.0</b>	2012Q2	40	80 Evergreen shader cores	488-680	18	L2: 1MB Sys mem (DDR3)	21	OpenCL, C++AMP	APU
	<b>Trinity</b>	2012Q2	32	128-384 Northern Islands cores	723-800	17-100	L2: 4MB Sys mem (DDR3)	25		
Nvidia	<b>Fermi</b>	2010Q1	40	512 Cuda cores (16 SMs)	1300	238	L1: 48KB L2: 768KB 6GB	<b>148</b>	CUDA, OpenCL, OpenACC	3X Perf/Watt, Dynamic Parallelism, HyperQ
	<b>Kepler (GK110)</b>	2012Q4	28	<b>2880</b> Cuda cores	836/876	300	6GB GDDR5	<b>288.5</b>		

# #1 in Top500 (11/2012): **Titan** @ Oak Ridge National Lab.

- **18,688** AMD Opteron 6274 16-core CPUs (32GB DDR3) .
- **18,688** Nvidia Tesla **K20X** GPUs
- Total RAM size: over 710 TB
- Total Storage: 10 PB.
- **Peak Performance: 27 Petaflop/s**
  - GPU: CPU = 1.311 TF/s: 0.141 TF/s = **9.3 : 1**
- **Linpack: 17.59 Petaflop/s**
- **Power Consumption: 8.2 MW**



Titan compute board: 4 AMD Opteron  
+ 4 NVIDIA Tesla K20X GPUs



NVIDIA Tesla K20X (Kepler GK110)  
GPU: **2688** CUDA cores

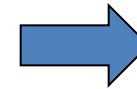




# Design Challenge: GPU Can't Handle Dynamic Loops

**GPU = SIMD/Vector**

Data Dependency Issues (RAW, WAW)



Solutions?

## Static loops

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

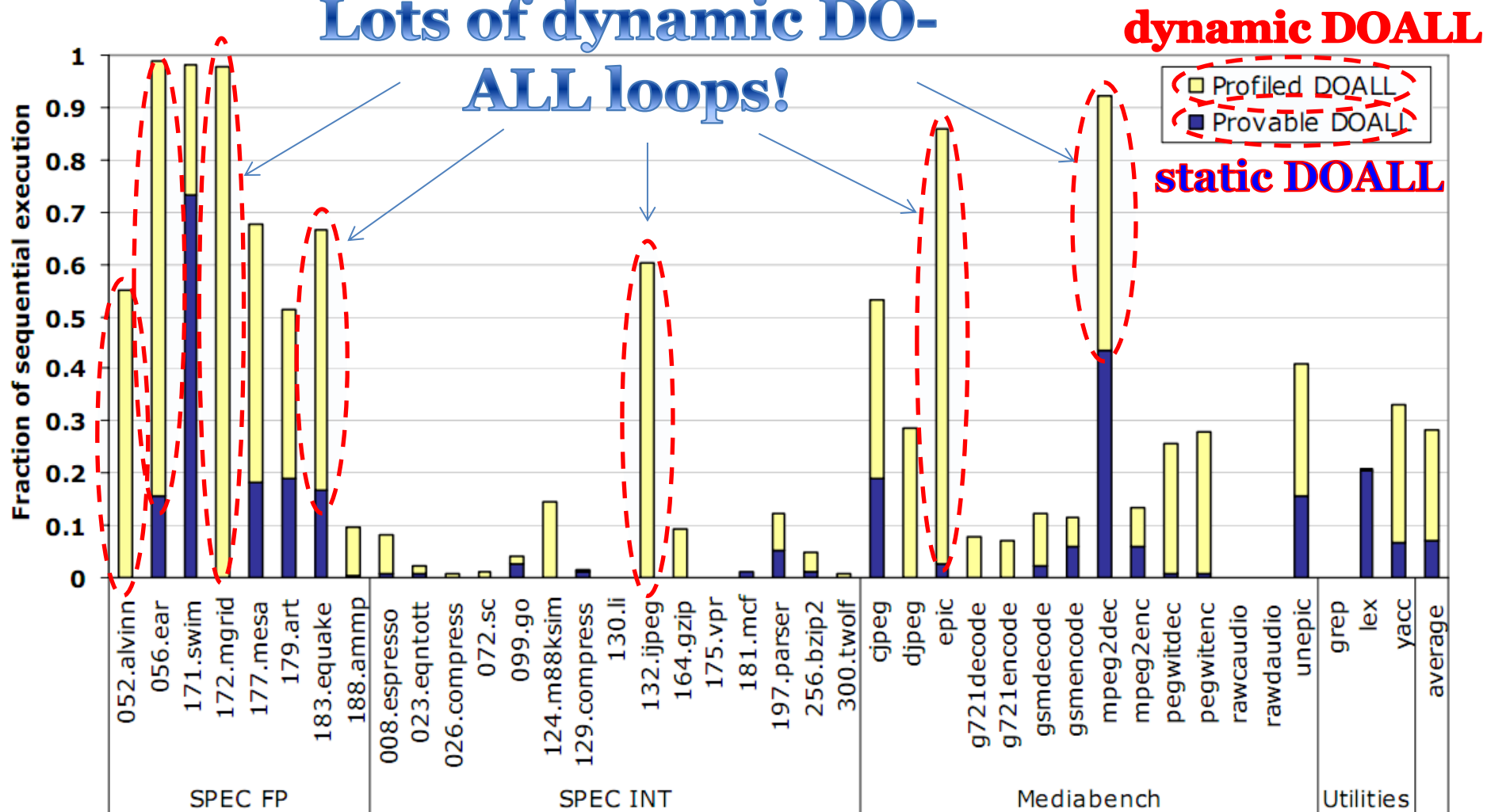
## Dynamic loops

```
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.

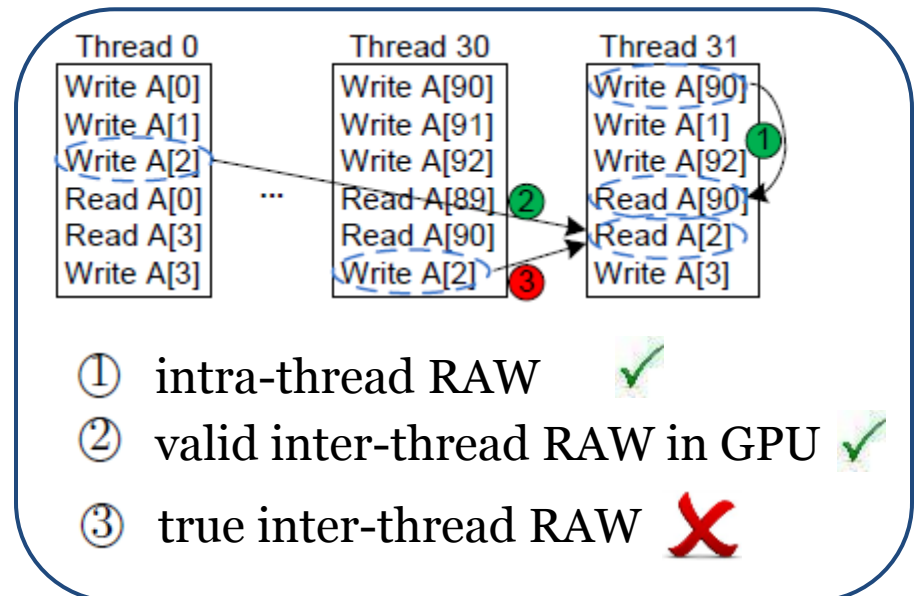
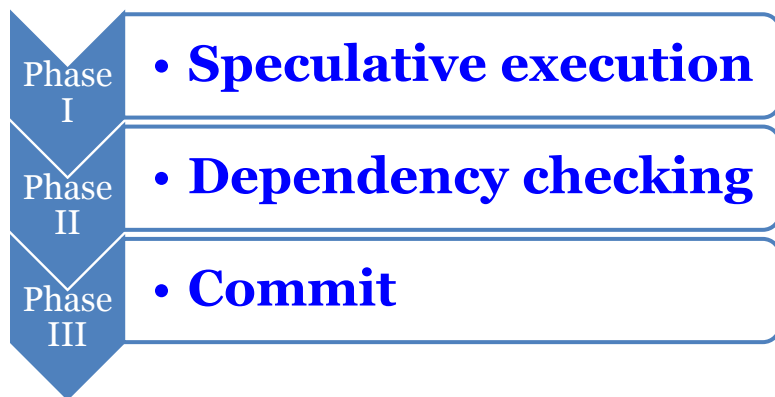
# Dynamic loops are common in scientific and engineering applications

**Lots of dynamic DO-ALL loops!**



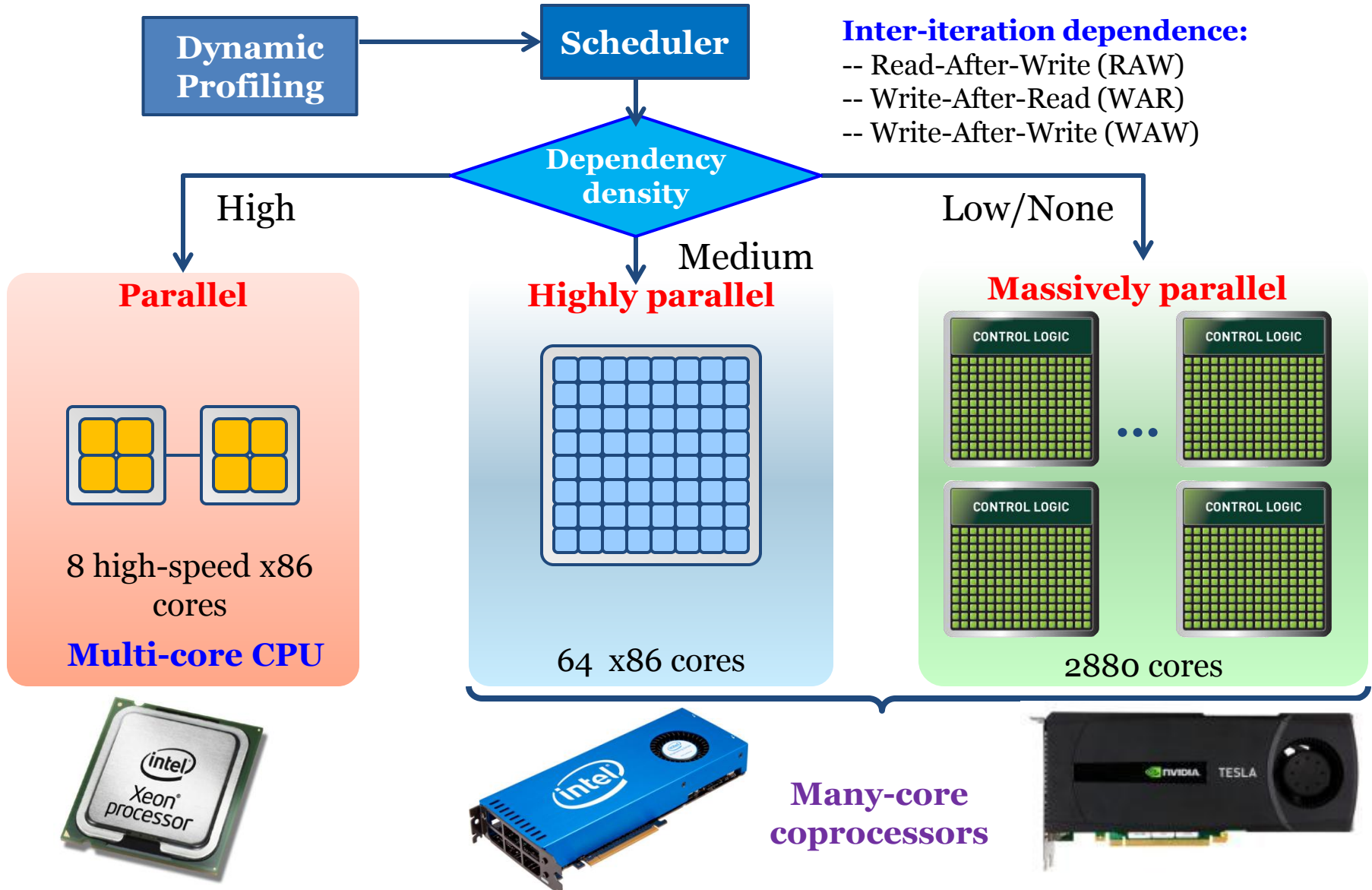
# GPU-TLS : Thread-level Speculation on GPU

- **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**



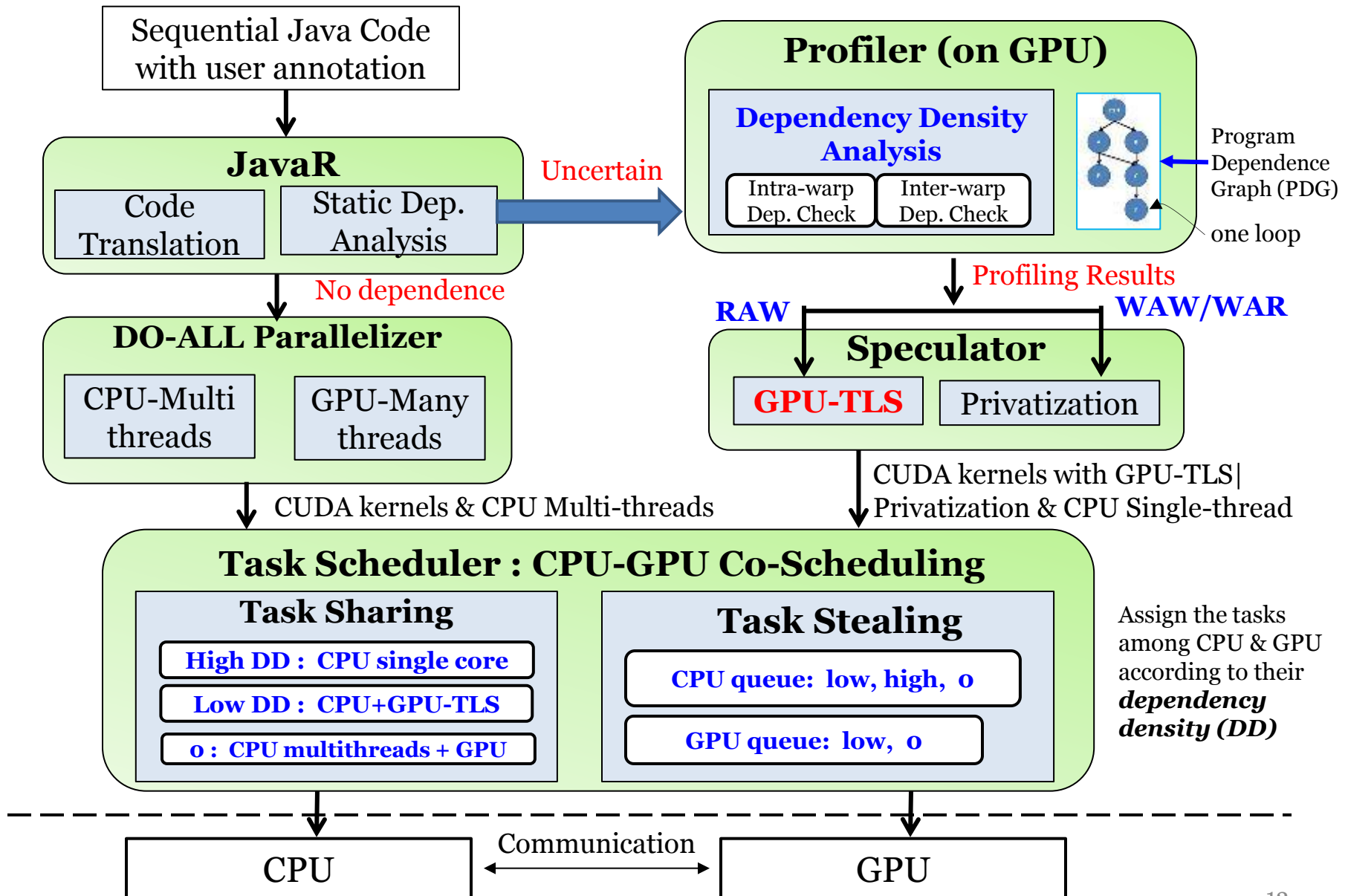
GPU: **lock-step execution** in the same warp (32 threads per warp).

# JAPONICA : Profile-Guided Work Dispatching



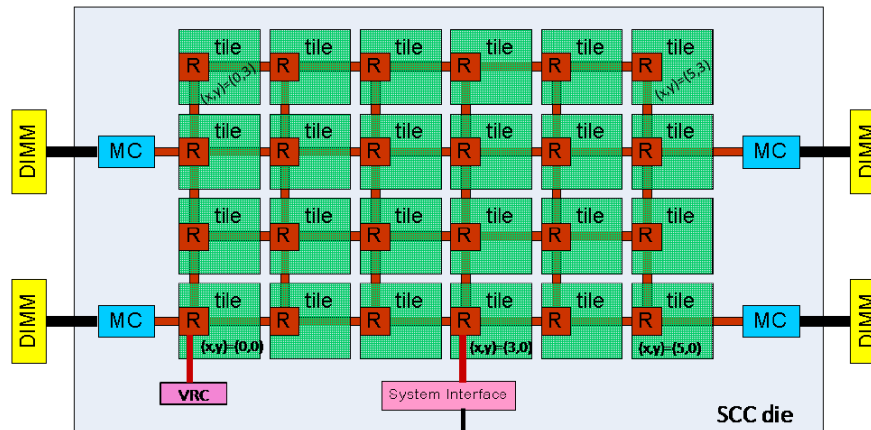


# JAPONICA : System Architecture





## (2) Crocodiles: Cloud Runtime with Object Coherence On Dynamic **tILES**”



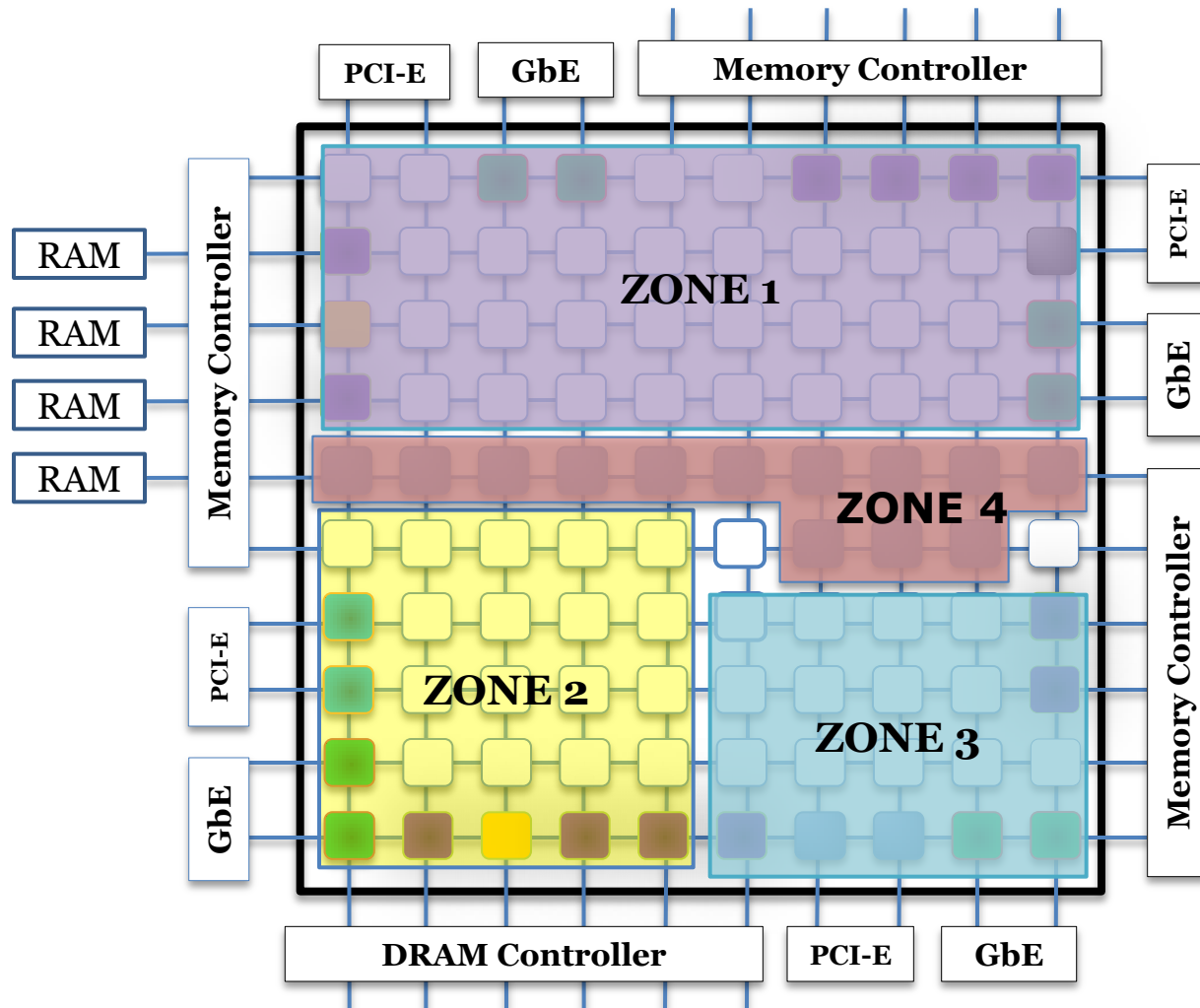
# “General Purpose” Manycore

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

## 鳄鱼 @ HKU (01/2013-12/2015)

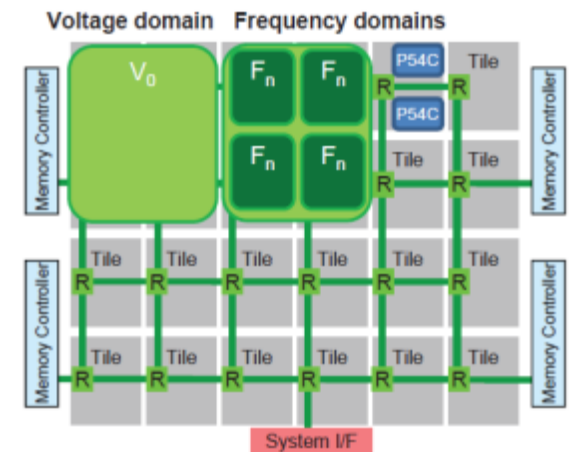
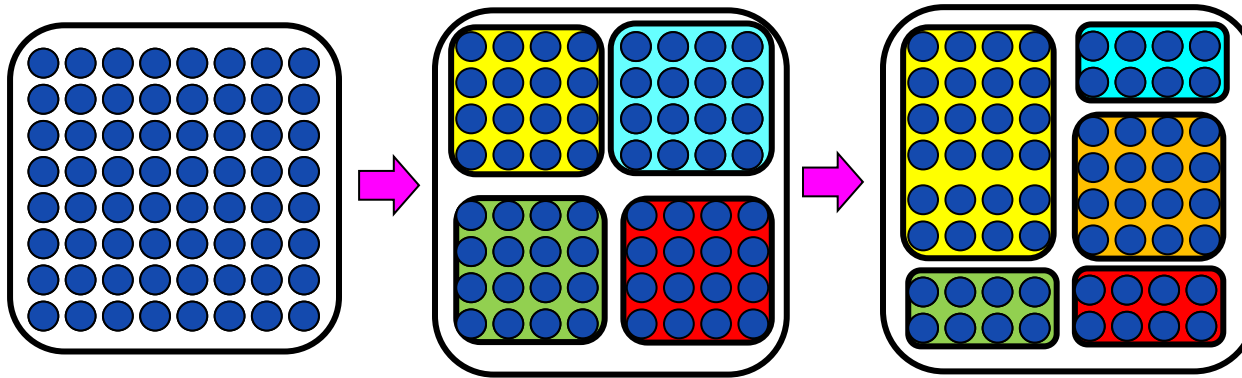
- **Crocodiles**: Cloud Runtime with Object Coherence On Dynamic **tILES** for future 1000-core tiled processors”





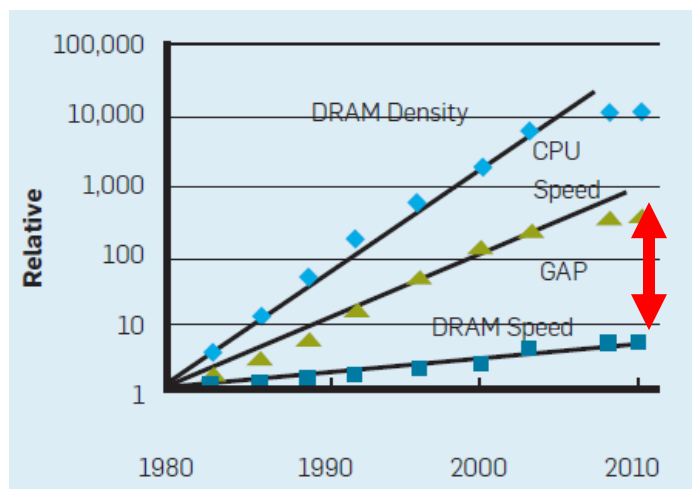
## • Dynamic Zoning

- Multi-tenant Cloud Architecture → Partition varies over time, mimic “Data center on a Chip”.
- Performance isolation
- On-demand scaling.
- Power efficiency (high flops/watt).

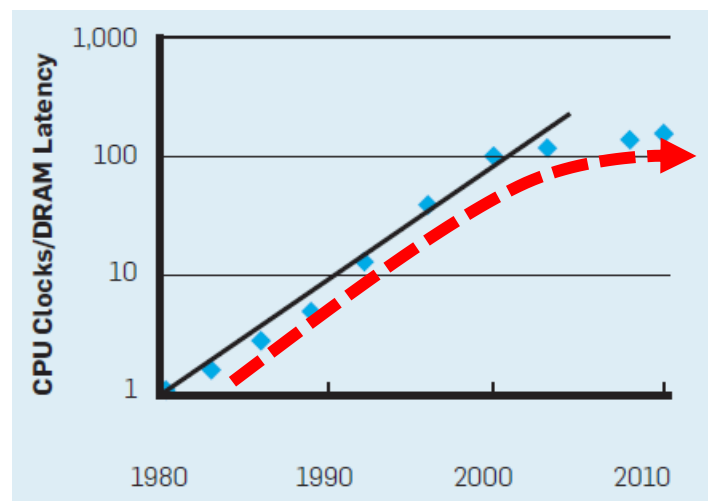


# Design Challenge: “Off-chip Memory Wall” Problem

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



(a) Gap of DRAM Density & Speed

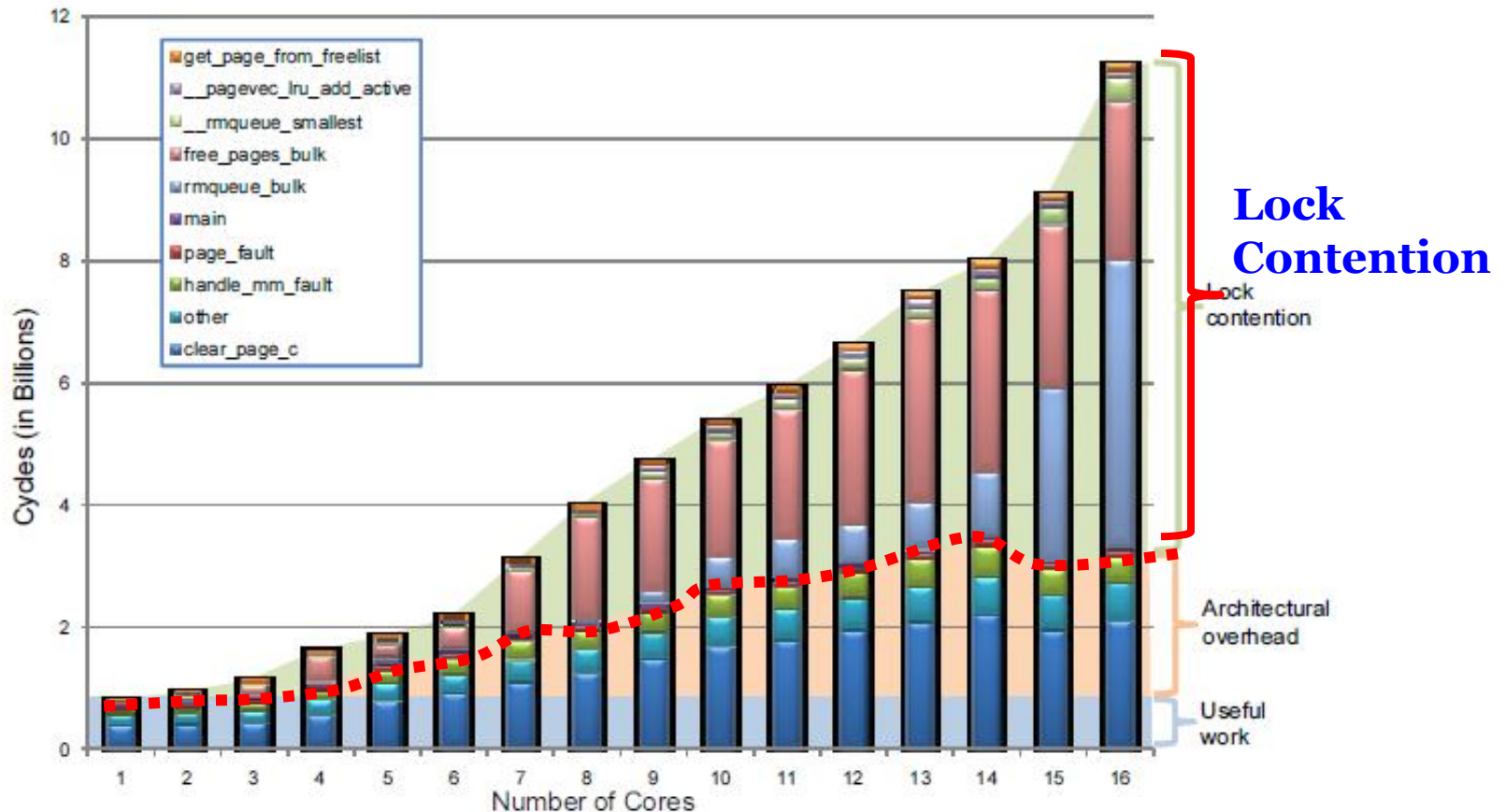


(b) DRAM Latency Not Improved

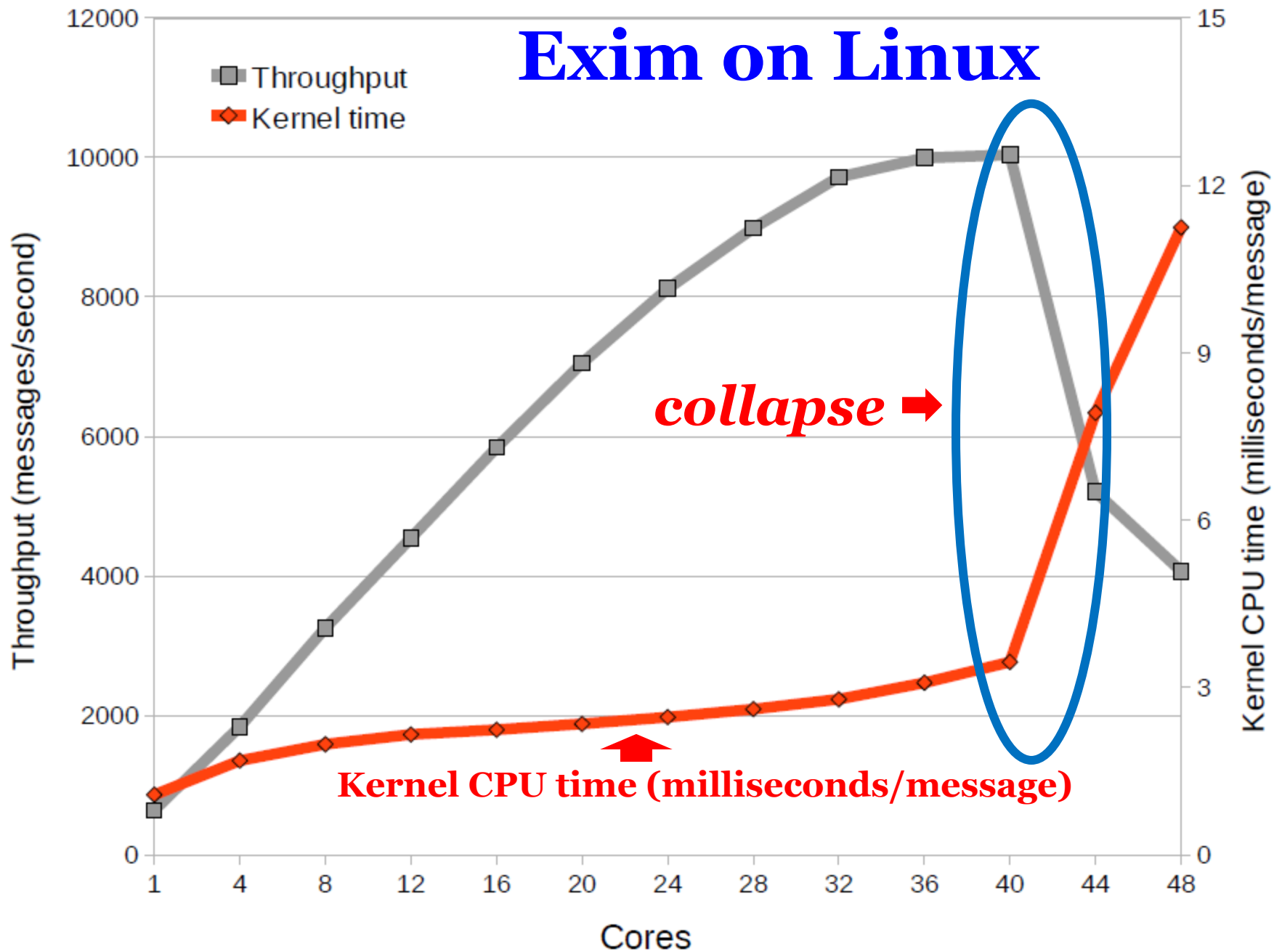
**Memory density has doubled nearly every two years, while performance has improved slowly (e.g. still 100+ of core clock cycles per memory access)**

# 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.



# Exim on Linux

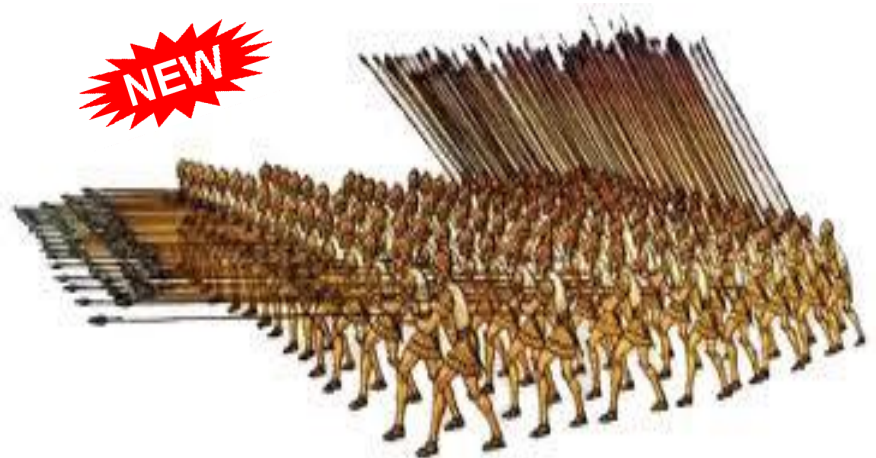




# Challenges and Potential Solutions

- **Cache-aware design**
  - Data Locality/Working Set getting critical!
  - Compiler or runtime techniques to improve data reuse
- **Stop multitasking**
  - Context switching breaks data locality
  - **Time Sharing → Space Sharing**

马其顿方阵众核操作系统：  
Next-generation Operating  
System for 1000-core  
processor



# Thanks!

**For more information:**

C.L. Wang's webpage:

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



We have a few PhD (or 4-year 直博生) positions open for self-motivated and academically strong students this year. If you are interested in one of the projects, please contact me at [clwang@cs.hku.hk](mailto:clwang@cs.hku.hk). Interview will be arranged for qualified students.

- 1 **Crocodiles : Scalable Cloud-on-Chip Runtime Support with Software Coherence for Future 1000-Core Tiled Architectures**, HKU 716712E, 9/2012-8/2015, supported by HK RGC.

Moving up to a parallelism with 1,000 cores requires a fairly radical rethinking of how to design system software. With a growing number of cores, providing hardware-level cache coherence gets increasingly complicated and costly, leading researchers to promote abandoning it if future many-core architectures are to stay inherently scalable. That means software now has to take on the role in ensuring data coherence among cores, yet exposing the low-level core-to-core message passing interfaces to programmers for managing coherence hampers programmability considerably. In this research, we address the above issues and propose novel methodologies to build a scalable CoC runtime platform, dubbed **Crocodiles** (*Cloud Runtime with Object Coherence On Dynamic tILES*), for future 1000-core tiled processors. **Crocodiles** involves the development of two important software subsystems: (1) Cache coherency protocol (2) **DVFS**-based power management.

➤ **2 Ph.D students (highest priority)**: strong background in OS kernel, full knowledge in memory subsystem (cache/DRAM, paging), cache coherent protocols.

➤ **1-2 RAs**: Require strong background in [software distributed shared memory systems](#) (e.g., TreadMarks, JiaJia, [JUMP](#)), programming experiences in multicore power management systems. Starting date: **ASAP**.

- 2 ➤ **Japonica: Transparent Runtime and Memory Coherence Support for GPU Based Heterogeneous Many-Core Architecture**, 11/2011-10/2013, supported by HK RGC.

In this project, we propose a new runtime platform, called **Japonica** (Java with Auto-Parallelization ON GraphIcs Co-processing Architecture), which enables a multithreaded Java program to scale transparently on a GPU-based heterogeneous system. With the transparent runtime support, application developers can utilize both CPU and GPU resources seamlessly with an idiomatic Java programming model. Japonica possesses several unique features: **(1)** automatic translation from Java bytecode to OpenCL, **(2)** auto-parallelization of loops with non-deterministic data dependencies, **(3)** dynamic load scheduling and rebalancing via task migration between CPU and GPU, **(4)** virtual shared memory between host and device, and **(5)** speculative coherency protocol for threads running on both CPU and GPU cores. The proposed work explores GPU-friendly ways to support a partial Java heap and STM-based synchronization of shared objects and arrays mirrored in GPU.

**1 Ph.D student**: interested in Java Virtual Machine, compiler (e.g., loop parallelization), software transactional memory. Experiences in GPU programming (CUDA or OpenCL) is required.

**RAs**: We are now building **Japonica** on a multicore GPU Cluster. We need 1 or 2 RAs who are good in OpenCL or CUDA programming.

More Information about Distributed Java Virtual Machines: <http://www.cs.hku.hk/~clwang/projects/JESSICA2.html>

- 3 ➤ **Self-Organizing Desktop Cloud (SODC)**

We are currently building a P2P Personal Cloud system which heavily adopts the Xen virtual machines. The research focus will be on (1) **VM I/O performance isolation**, (2) **live VM migration over WAN**. Read [WAVNet](#) webpage for more details.

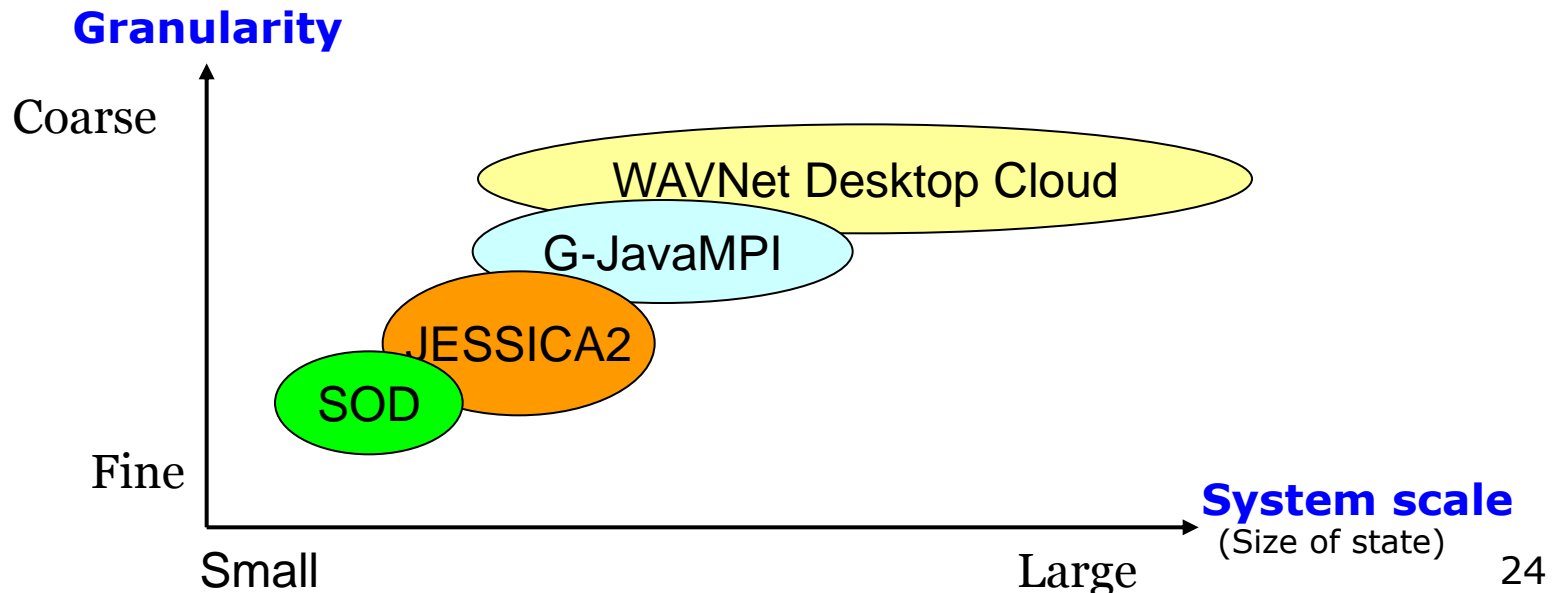
**1 Ph.D student**: Look for a strong candidate with good knowledge in virtual machines internal design (e.g., KVM, Xen), solid background in Linux kernel, and good knowledge in NAT/firewall tunneling solutions.

- 4 ➤ **OS-1K: New Operating System for Manycore Systems (part of Crocodiles project)**

Traditional operating systems are based on the sequential execution model developed in the 1960s. Such operating systems cannot address new many-core parallel hardware architecture without major redevelopment. For instance, how can you harness the power a next-generation manycore processor with >1,000 cores? We will investigate various perspectives on the future OS design towards the goal.

# 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)



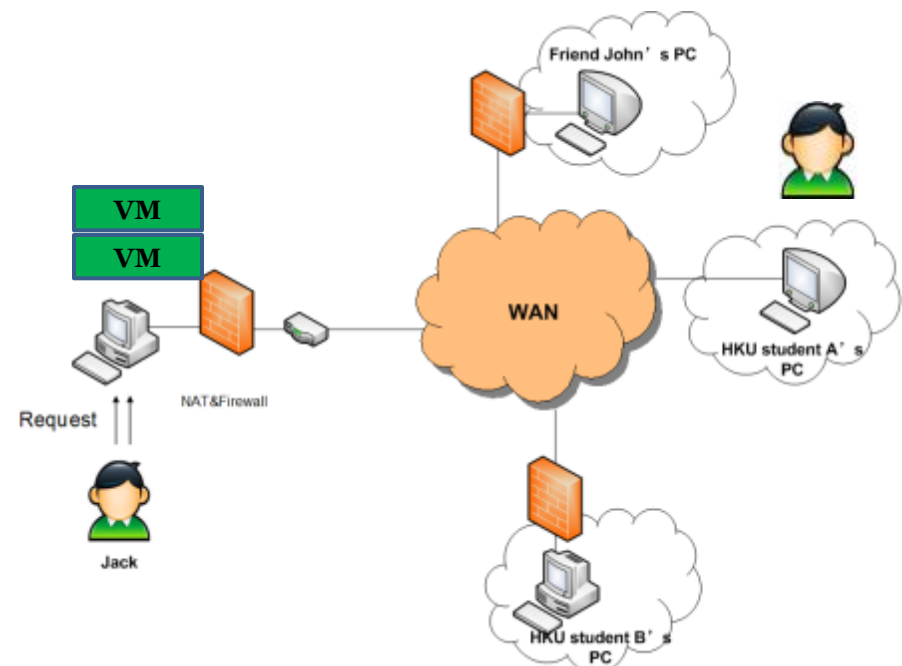
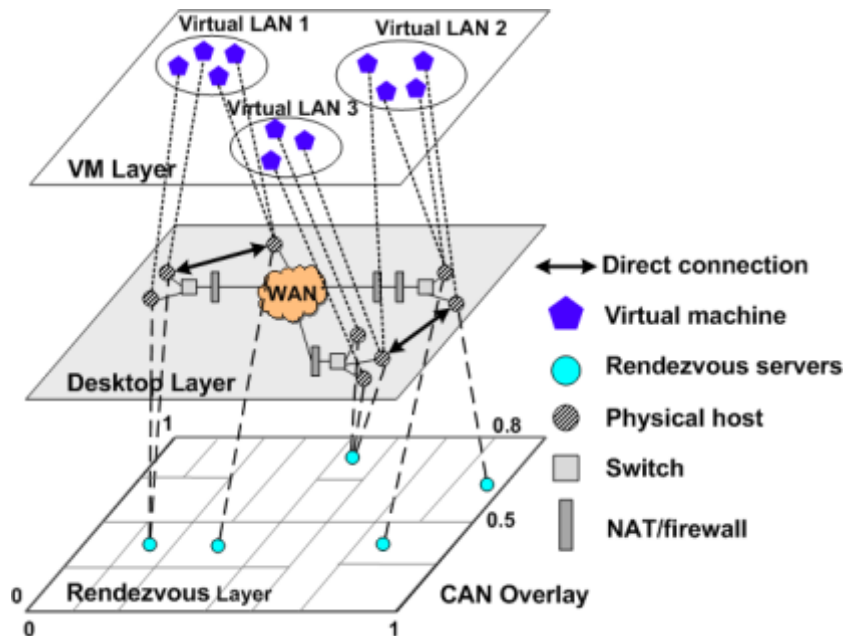


# WAVNet: Live VM Migration over WAN

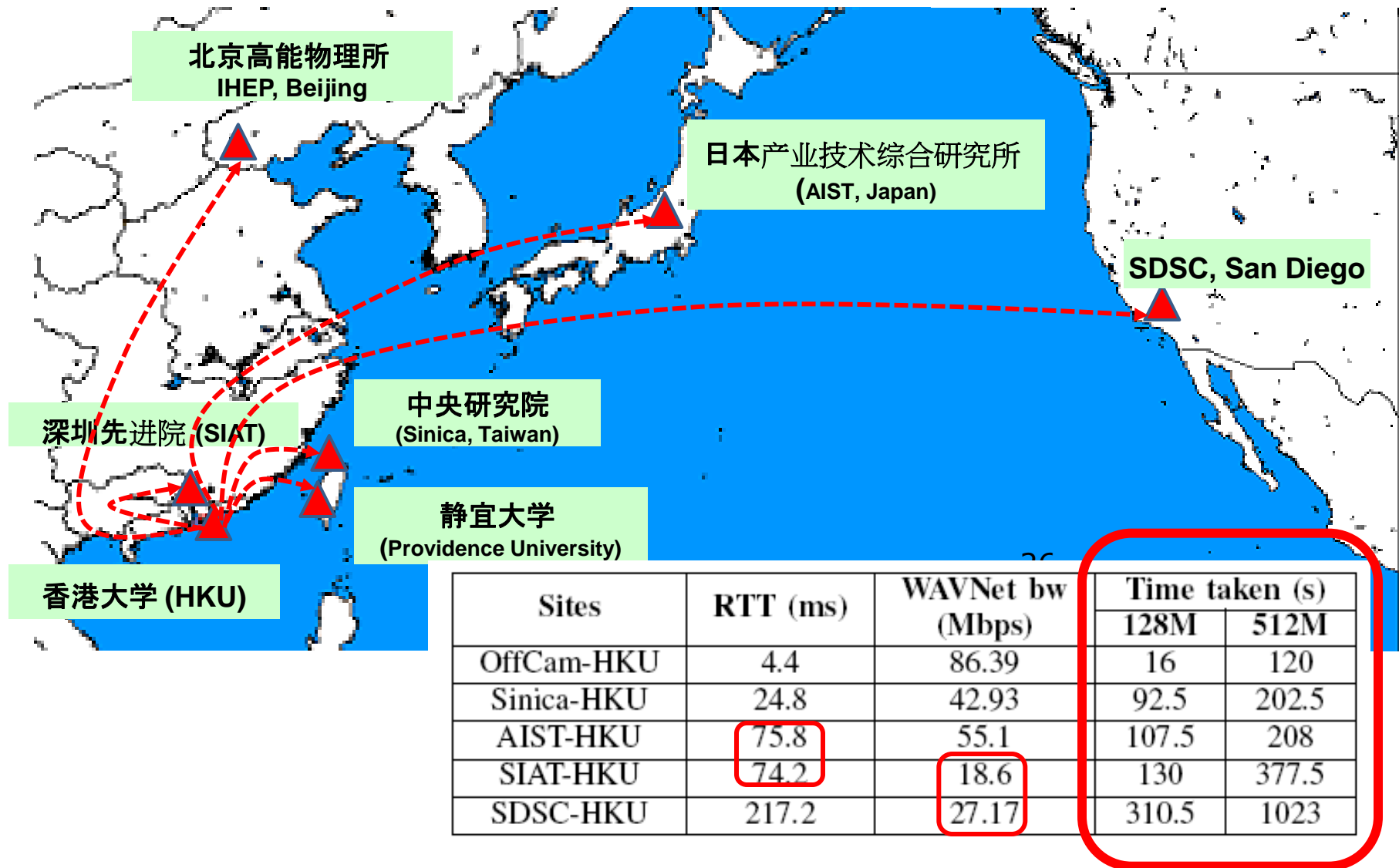
- A P2P Cloud with Live VM Migration over WAN
  - “Virtualized LAN” over the Internet”
- High penetration via NAT hole punching
  - Establish direct host-to-host connection
  - Free from proxies, able to traverse most NATs



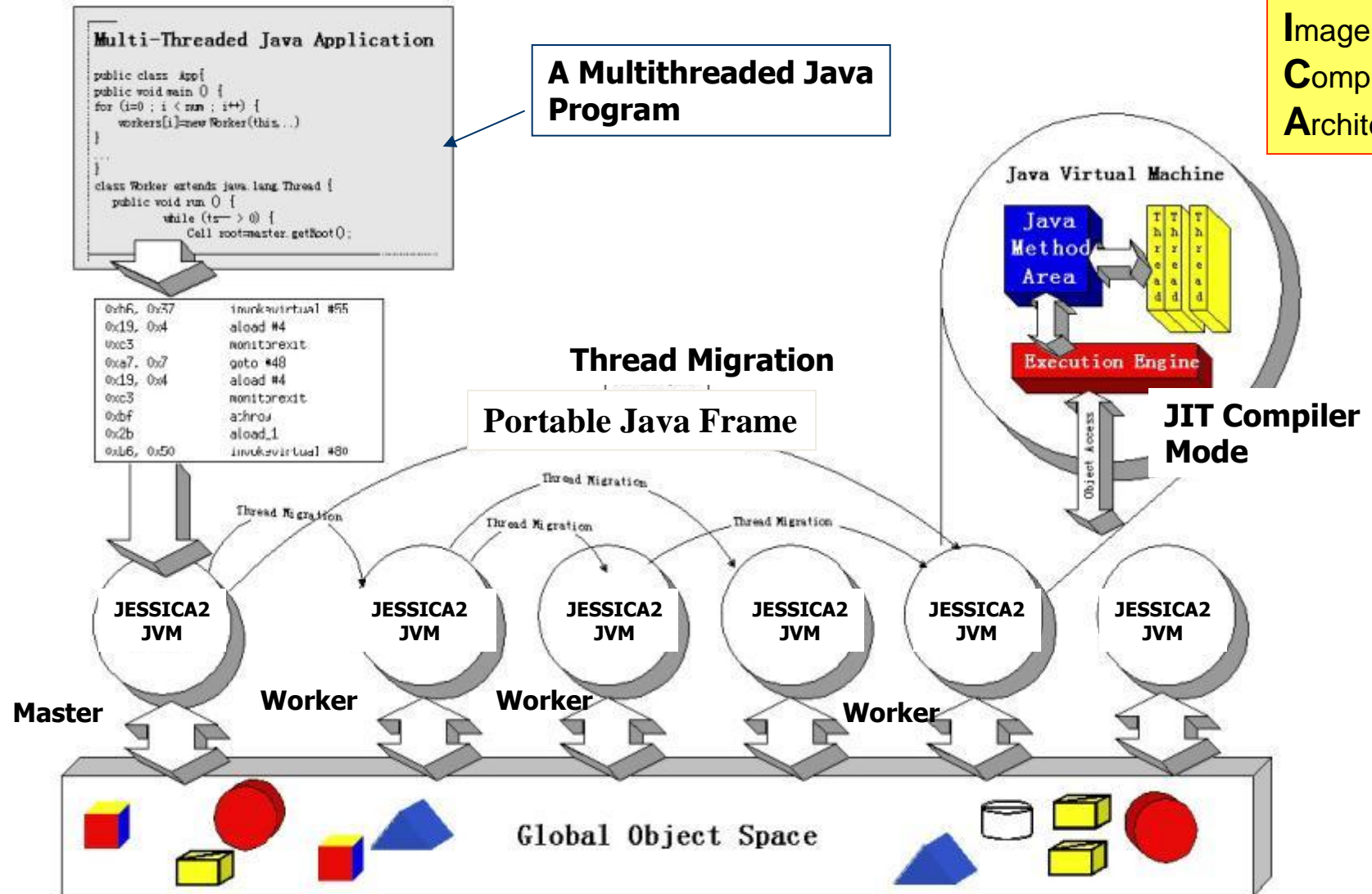
## Key Members



# WAVNet: Experiments at Pacific Rim Areas



# JESSICA2: Distributed Java Virtual Machine



# 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
  - **TrC-DC** : a software transactional memory system on cluster with distributed clocks (not discussed)



**Past Members**



King Tin LAM,



Chenggang Zhang



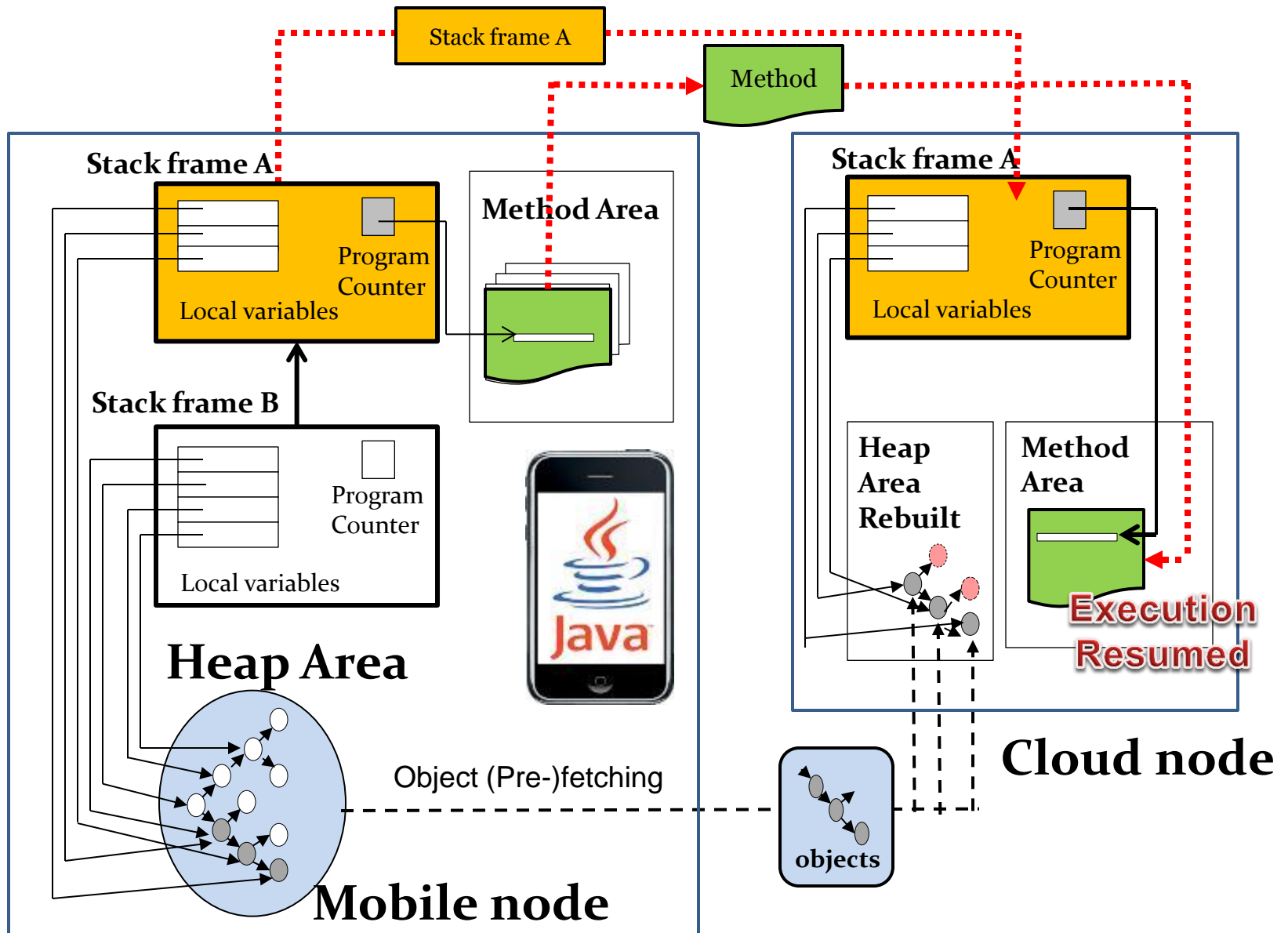
Kinson Chan



Ricky Ma

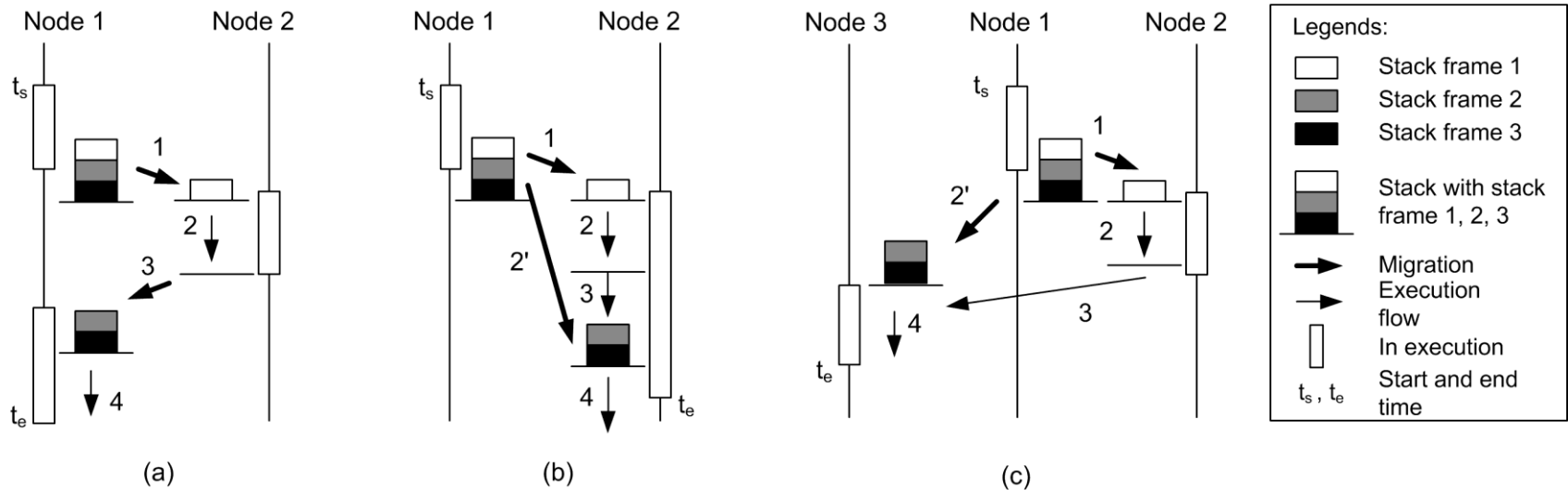
J1 and J2 received a total of **1107** source code downloads

# Stack-on-Demand (SOD)





# Elastic Execution Model via SOD



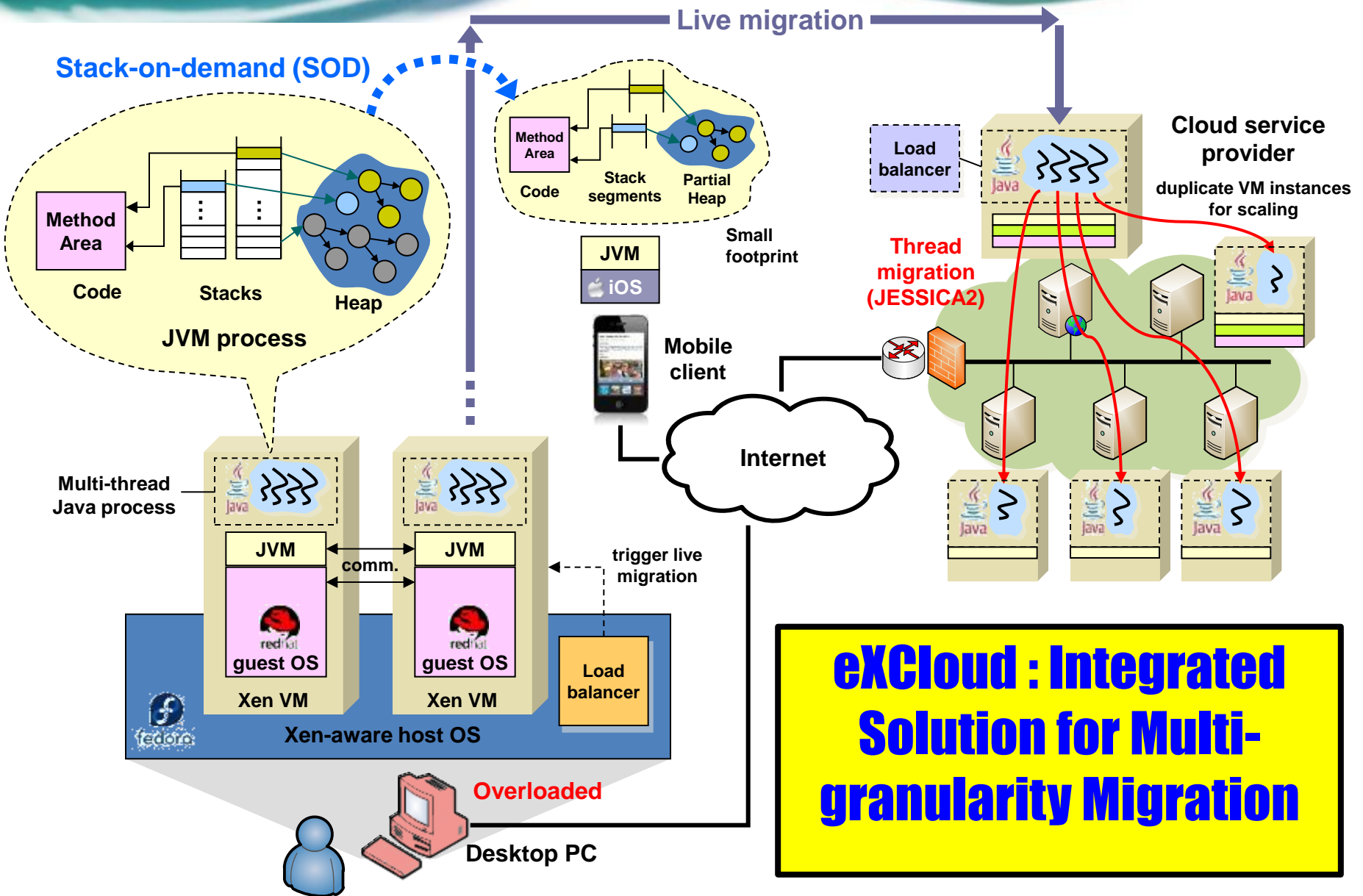
(a) "Remote Method Call"

(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**



# eXCloud : Integrated Solution for Multi-granularity Migration