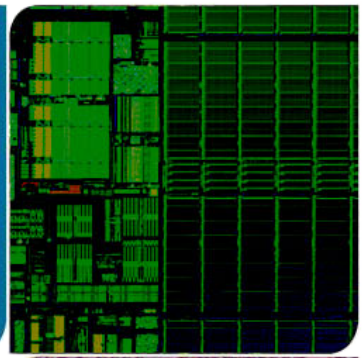
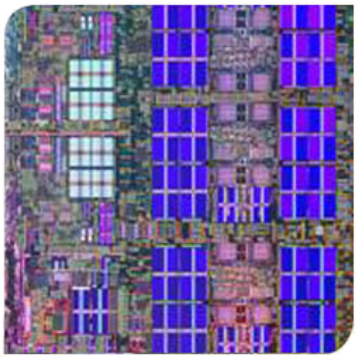




GraphBIG: Understanding Graph Computing in the Context Of Industrial Solutions

SC15 Austin, TX | hpc transforms.

Lifeng Nai, Hyesoon Kim (Georgia Tech)

Yinglong Xia, Ilie Tanase, Ching-Yung Lin (IBM Research)

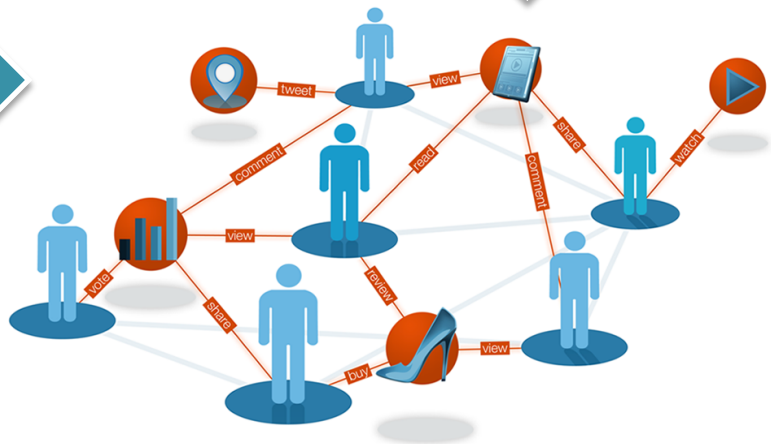
IBM System G

Georgia Tech

comparch

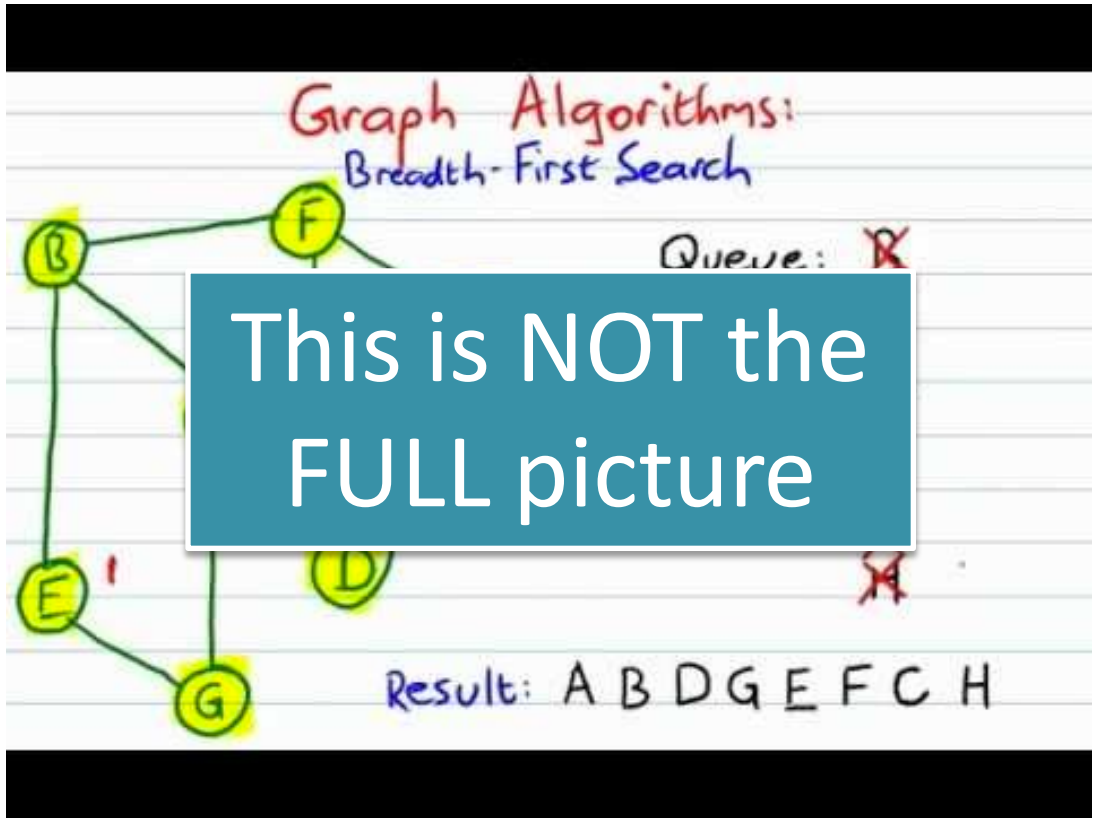
BIG DATA

- This is the Big Data era
- Big Data are linked



WHAT IS GRAPH COMPUTING

- Graph traversal?



This is NOT the FULL picture

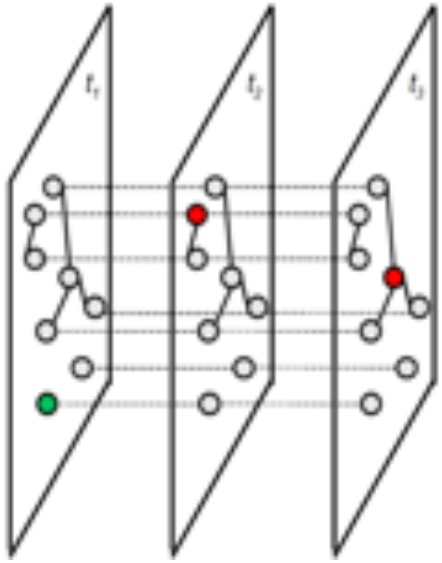
GRAPH COMPUTING

The GRAPH can be

- Big or Small

GRAPH COMPUTING

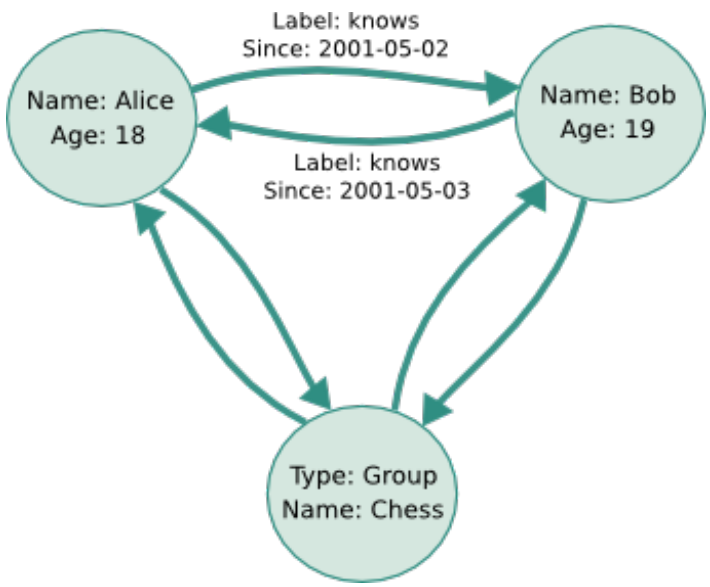
- The GRAPH can be
  - Static or Dynamic

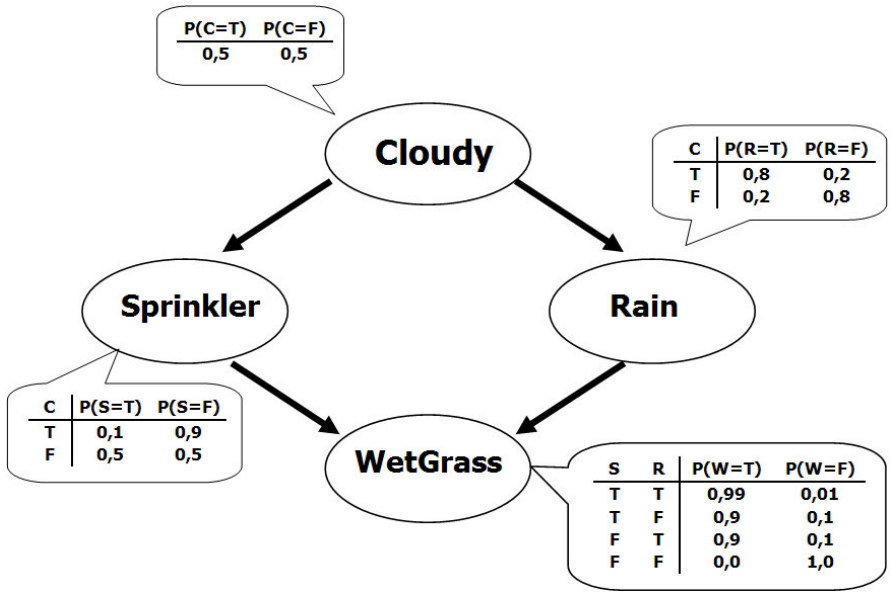


GRAPH COMPUTING

The GRAPH can be

- Property or Bayesian





P(C=T)	P(C=F)
0,5	0,5

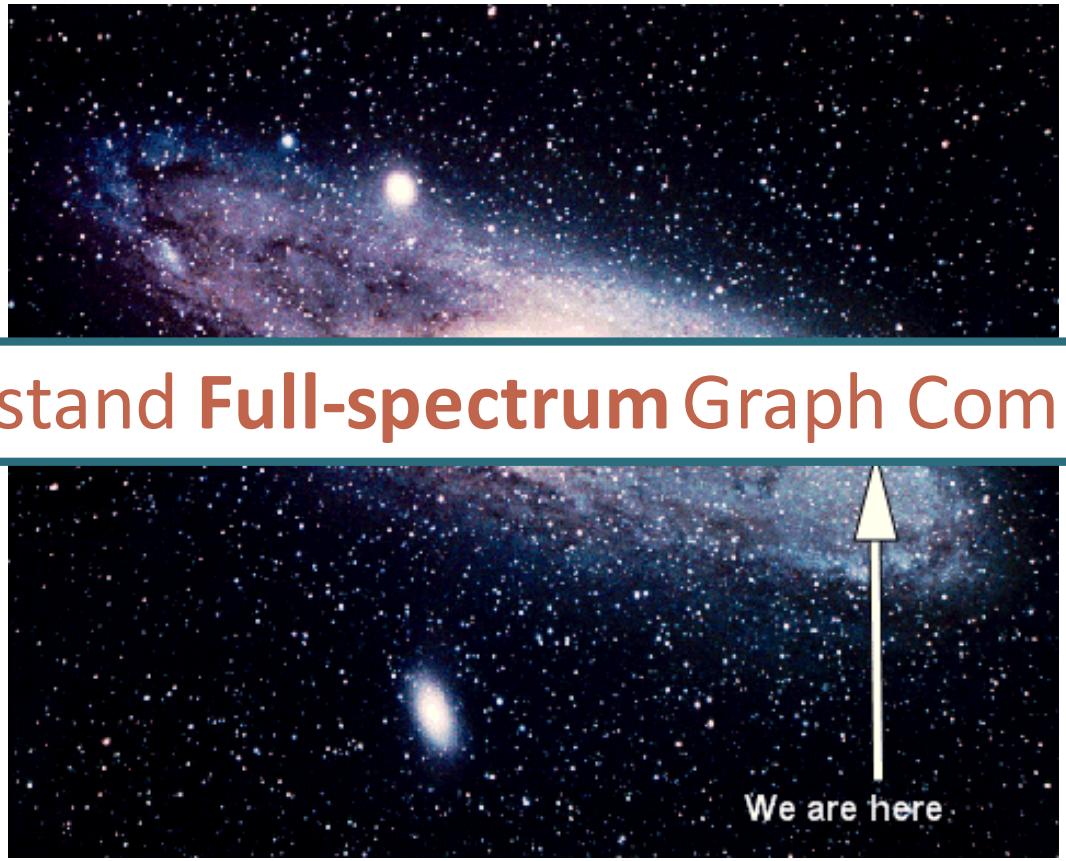
C	P(R=T)	P(R=F)
T	0,8	0,2
F	0,2	0,8

C	P(S=T)	P(S=F)
T	0,1	0,9
F	0,5	0,5

S	R	P(W=T)	P(W=F)
T	T	0,99	0,01
T	F	0,9	0,1
F	T	0,9	0,1
F	F	0,0	1,0

GRAPH COMPUTING

- Graph computing contains a BIG scope
  - $Traversal \neq Graph\ Computing$



Understand **Full-spectrum** Graph Computing

GRAPHBIG

- Understand **full-spectrum** graph computing
  - Diverse workloads + Framework
- Propose an open-source benchmark suite: **GraphBIG**
  - Workloads from real-world use cases
  - Cover major graph computing types and data types
  - Both CPU and GPU implementations
- An open-source graph framework: **OpenG**
  - Designed from scratch
  - Similar design methodology as **IBM System G** commercial toolkits

OUTLINE

- Motivation
- GraphBIG: Key factors
- Characterizations
- Conclusion

GRAPHBIG

OpenG Framework

Vertex-centric Data Representation

Representative Graph Workloads

Graph Datasets

GRAPHBIG

OpenG Framework

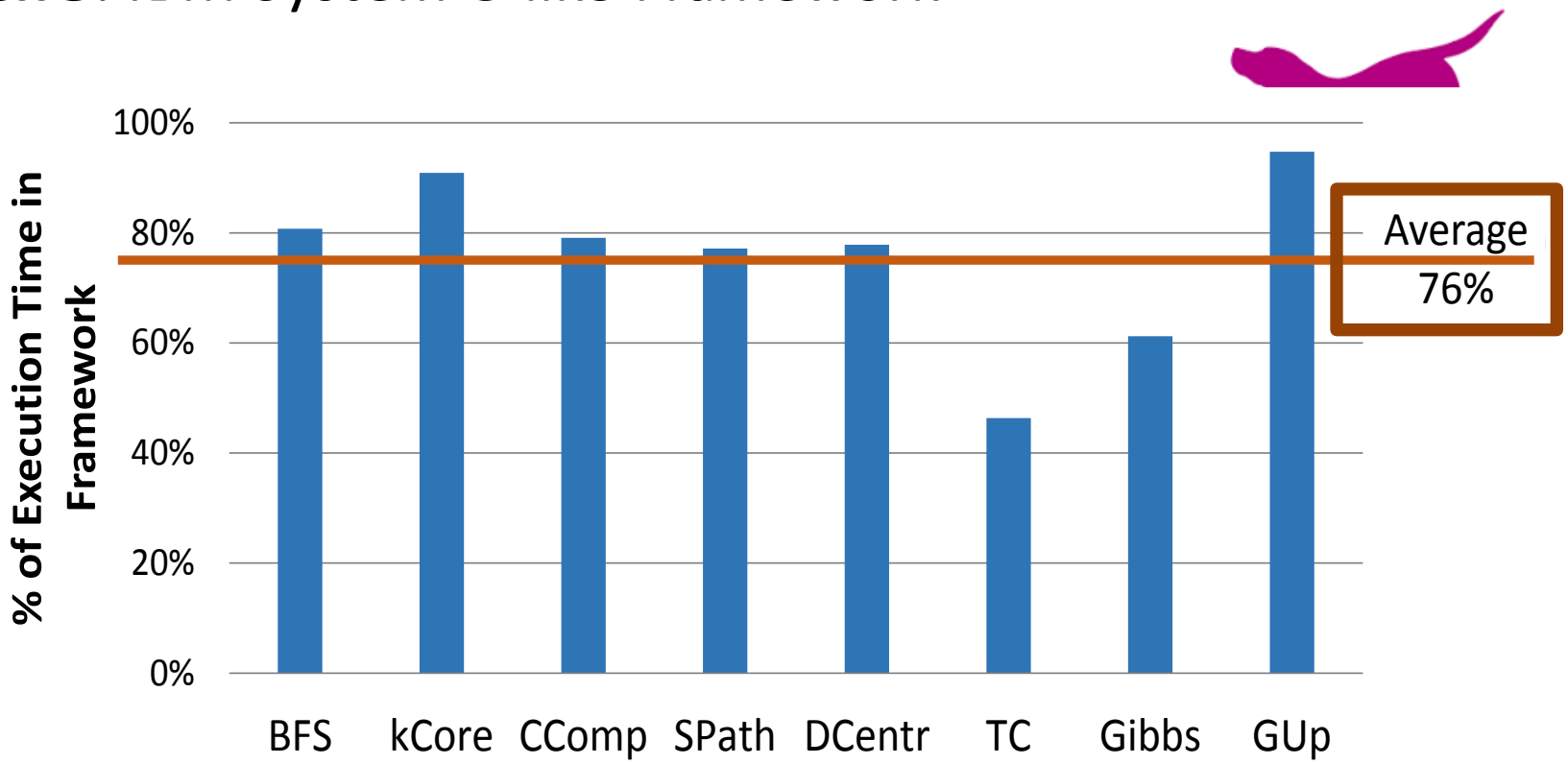
Vertex-centric Data Representation

Representative Graph Workloads

Graph Datasets

GRAPHBIG: FRAMEWORK

- Graph applications ← Framework primitives
- **OpenG**: IBM System G-like Framework



GRAPHBIG

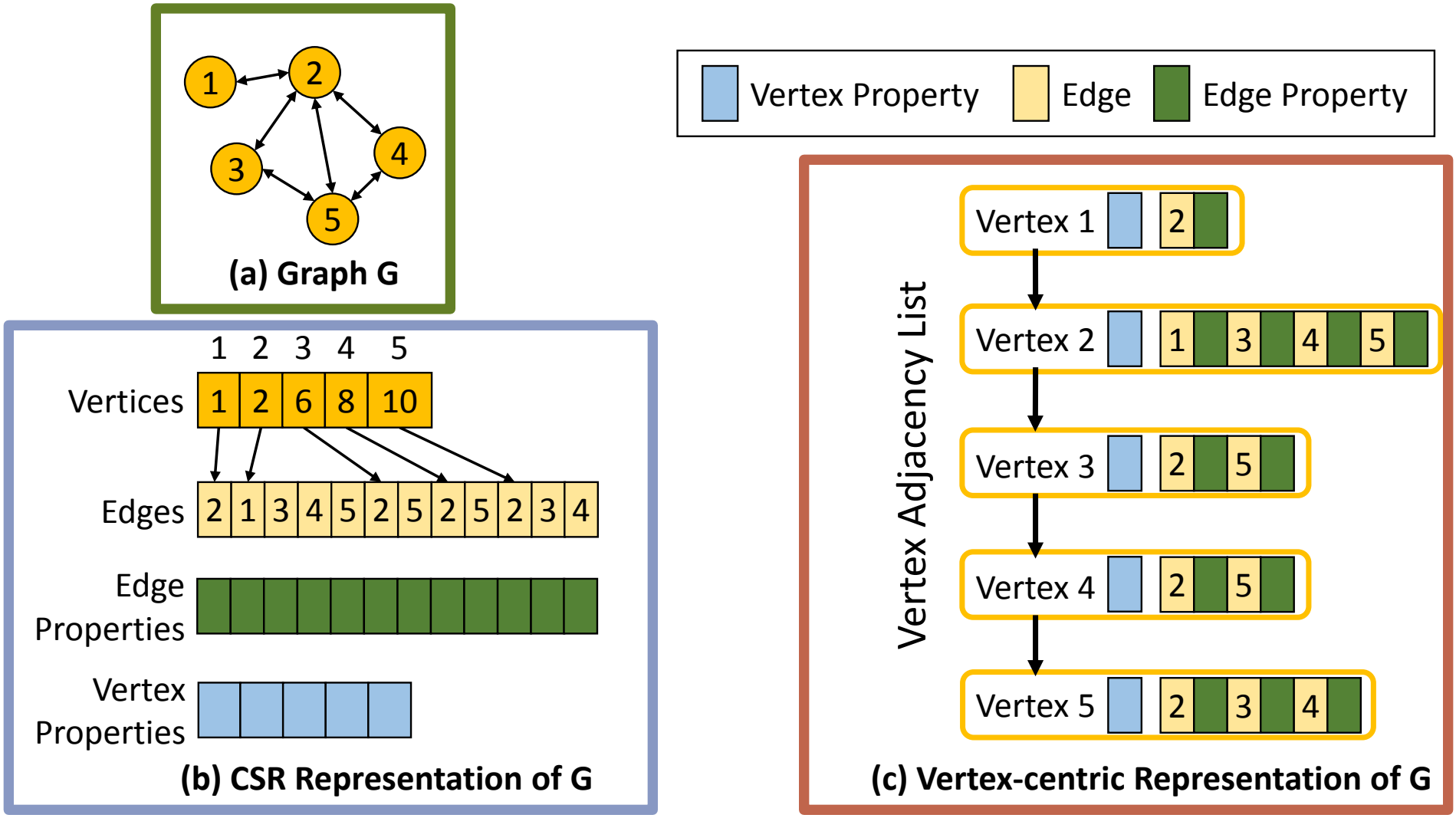
OpenG Framework

Vertex-centric Data Representation

Representative Graph Workloads

Graph Datasets

GRAPHBIG: DATA REPRESENTATION



GRAPHBIG

OpenG Framework

Vertex-centric Data Representation

Representative Graph Workloads

Graph Datasets

GRAPHBIG: WORKLOAD SELECTION

Coverage

- Workloads cover all computation types

Representativeness

- Workloads are selected from real-world use cases

GRAPHBIG: COMPUTATION TYPES

Computation on graph structure (CompStruct)

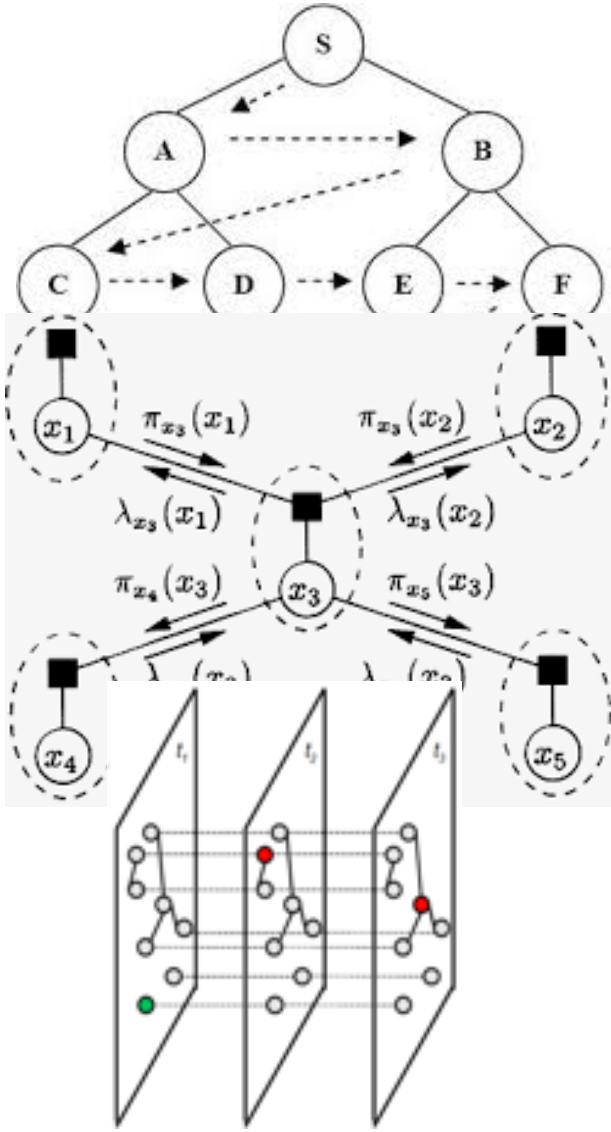
- Example: Breadth-first search
- Irregular access pattern, heavy read access

Computation on graph property (CompProp)

- Example: Belief propagation
- Heavy numeric operations on graph property

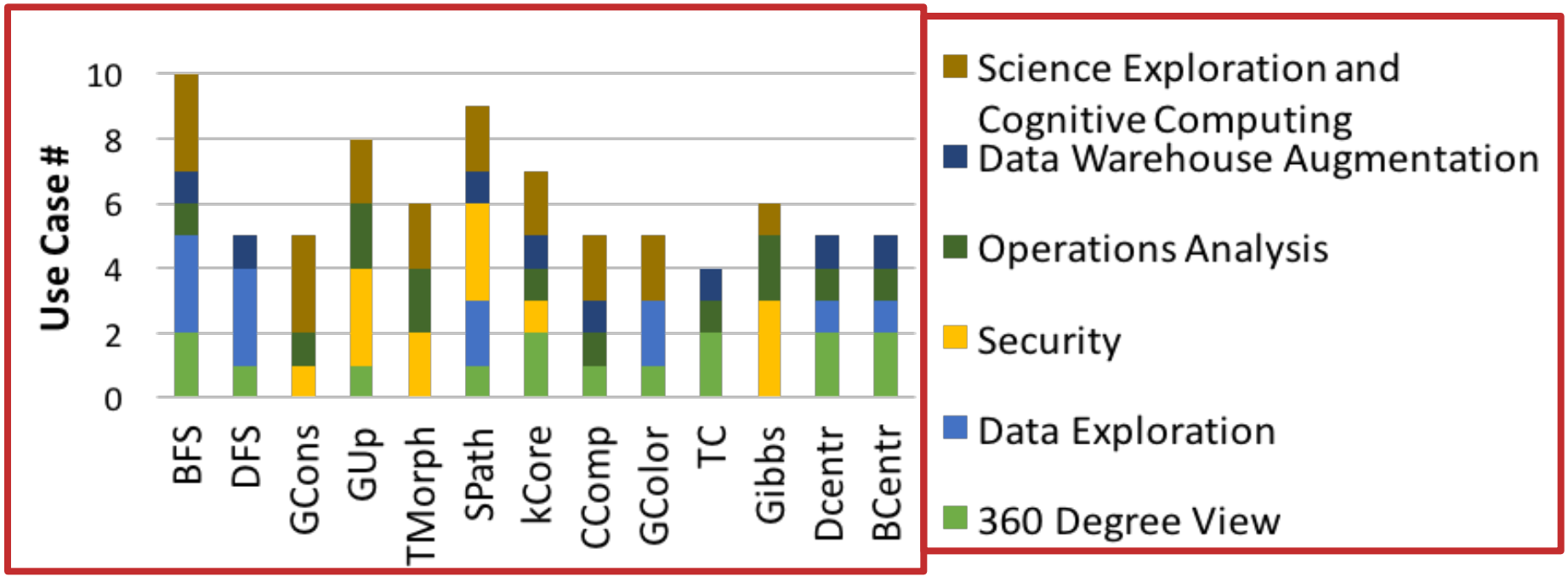
Computation on dynamic graph (CompDyn)

- Example: Streaming Graph
- Dynamic graph structure, dynamic memory usage



GRAPHBIG: WORKLOAD SELECTION

- Selected from **21 real-world use cases** of IBM System G



(A) # of Use Cases with Each Workload

(B) Distribution of Selected Use Cases in 6 Categories

GRAPHBIG: WORKLOADS

Category	Workload	Computation Type	CPU	GPU
Graph traversal	BFS	CompStruct	✔	✔
	DFS	CompStruct	✔	
Graph update	Graph construction (GCons)	CompDyn	✔	
	Graph update (GUp)	CompDyn	✔	
	Topology morphing (TMorph)	CompDyn	✔	
Graph analytics	Shortest path (SPath)	CompStruct	✔	✔
	kCore	CompStruct	✔	✔
	Connected component (CComp)	CompStruct	✔	✔
	Graph coloring (GColor)	CompStruct		✔
	Triangle counting (TC)	CompProp	✔	✔
	Gibbs Inference (GI)	CompProp	✔	
Social analytics	Betweenness Centrality (BCentr)	CompStruct	✔	✔
	Degree Centrality (DCentr)	CompStruct	✔	✔

GRAPHBIG

OpenG Framework

Vertex-centric Data Representation

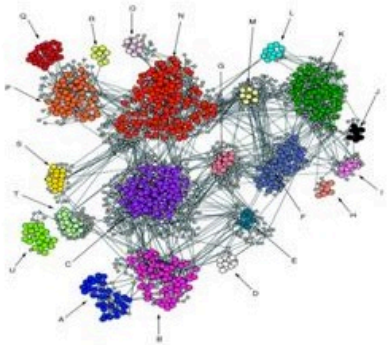
Representative Graph Workloads

Graph Datasets

GRAPHBIG: DATA TYPES

Big Data includes all sorts of Networks

Type 1

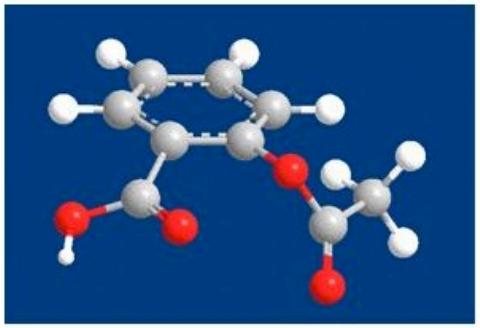


Social/Economic/Political Network

Type 2

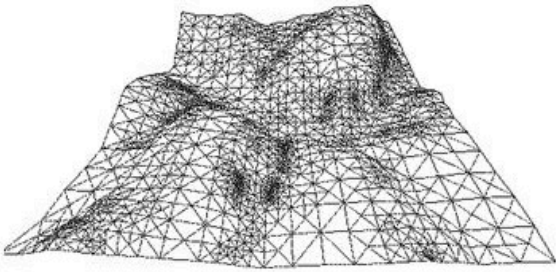
Information/Knowledge Network

Type 3



Nature/Bio/Cognitive Network

Type 4



Man-Made Technology Network

GRAPHBIG: DATASETS

Data set	Type	Vertex #	Edge #
Twitter Graph	Type 1	120M	1.9B
IBM Knowledge Repo	Type 2	154K	1.72M
IBM Watson Gene Graph	Type 3	2M	12.2M
CA Road Network	Type 4	1.9M	2.8M
LDBC Graph	Synthetic	Any	Any

CHARACTERIZATION

Methodology

- Real machine + hardware performance counters

- CPU: tool integrated within benchmarks
- GPU: CUDA nvprof

CHARACTERIZATION

Processor	Type	**Xeon E5-2670**
	Frequency	2.6 GHz
	Core #	**2 sockets x 8 cores x 2 threads**
	Cache	32KB L1, 256KB L2, 20MB L3
	Memory BW	51.2 GB/s (DDR3)
GPU	Type	**Nvidia Tesla K40**
	CUDA Core	2880
	Memory	12 GB
	Memory BW	288 GB/s
	Frequency	Core-745 MHz, mem-3 GHz
System	Memory	192 GB
	Disk	2 TB HDD
	OS	RHEL 6

CHARACTERIZATION

Showcase (Data: LDBC-graph 1M vertices)

- CPU execution time breakdown
- CPU core analysis
- CPU cache performance

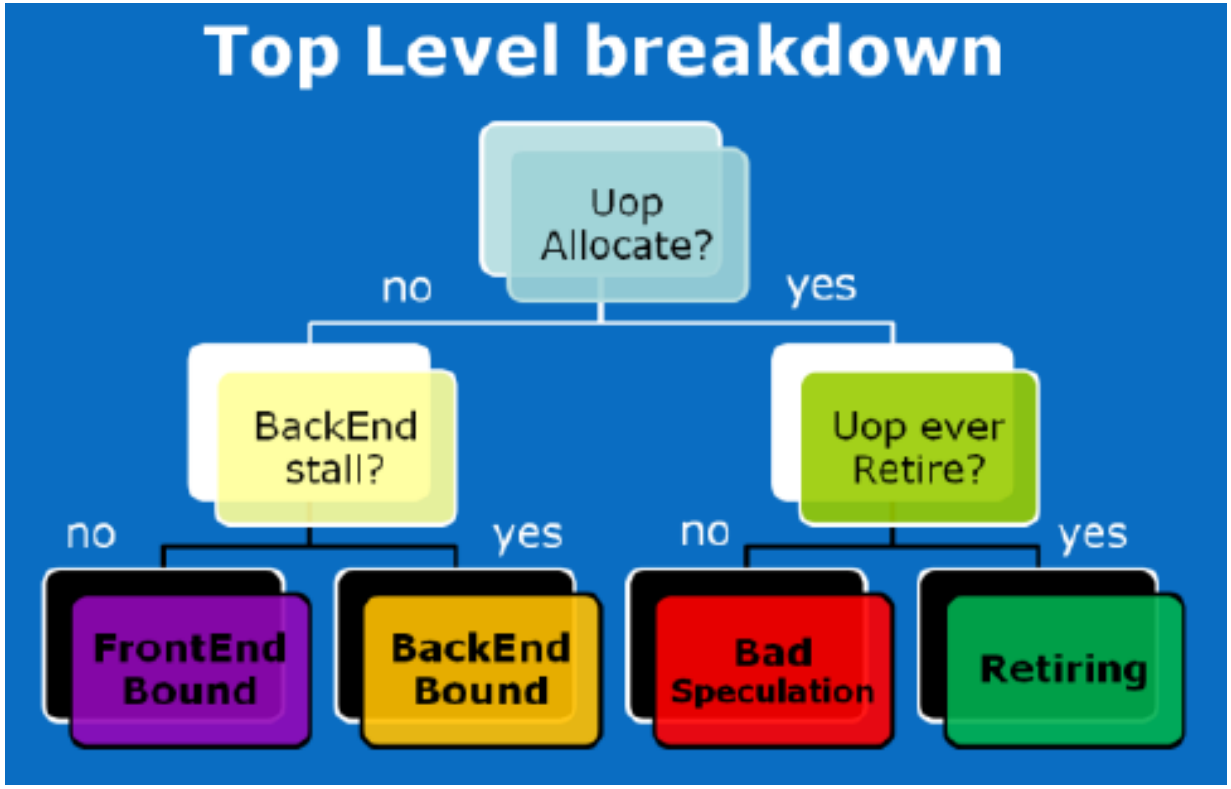
- GPU divergence
- GPU speedup

More experiment results can be found in the paper

- More analysis (memory bandwidth, IPC, etc.)
- Input data sensitivity (all data sets are evaluated)

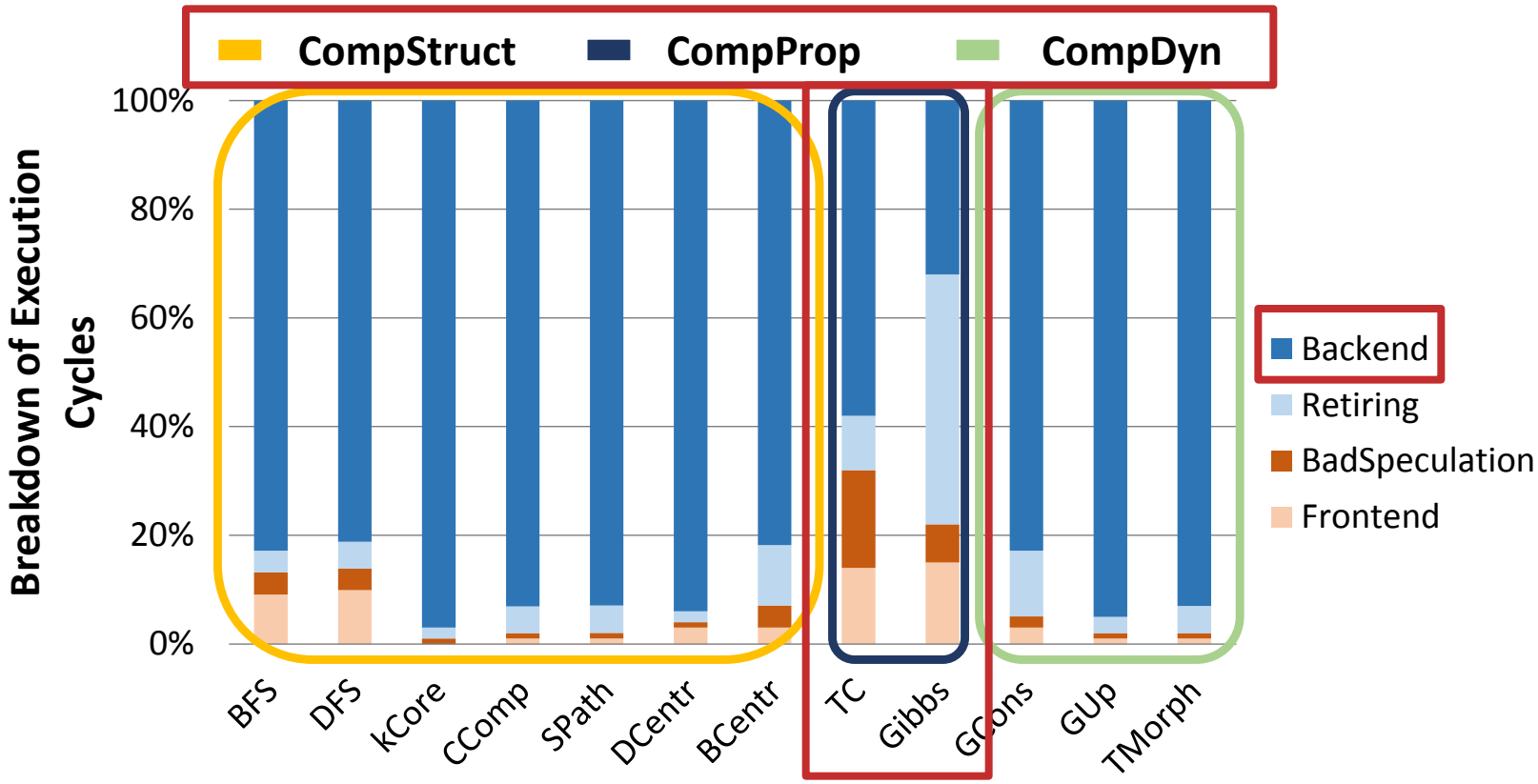
CPU: EXECUTION TIME BREAKDOWN

- Four categories:
  - Frontend, Backend, Bad Speculation, and **Retiring**



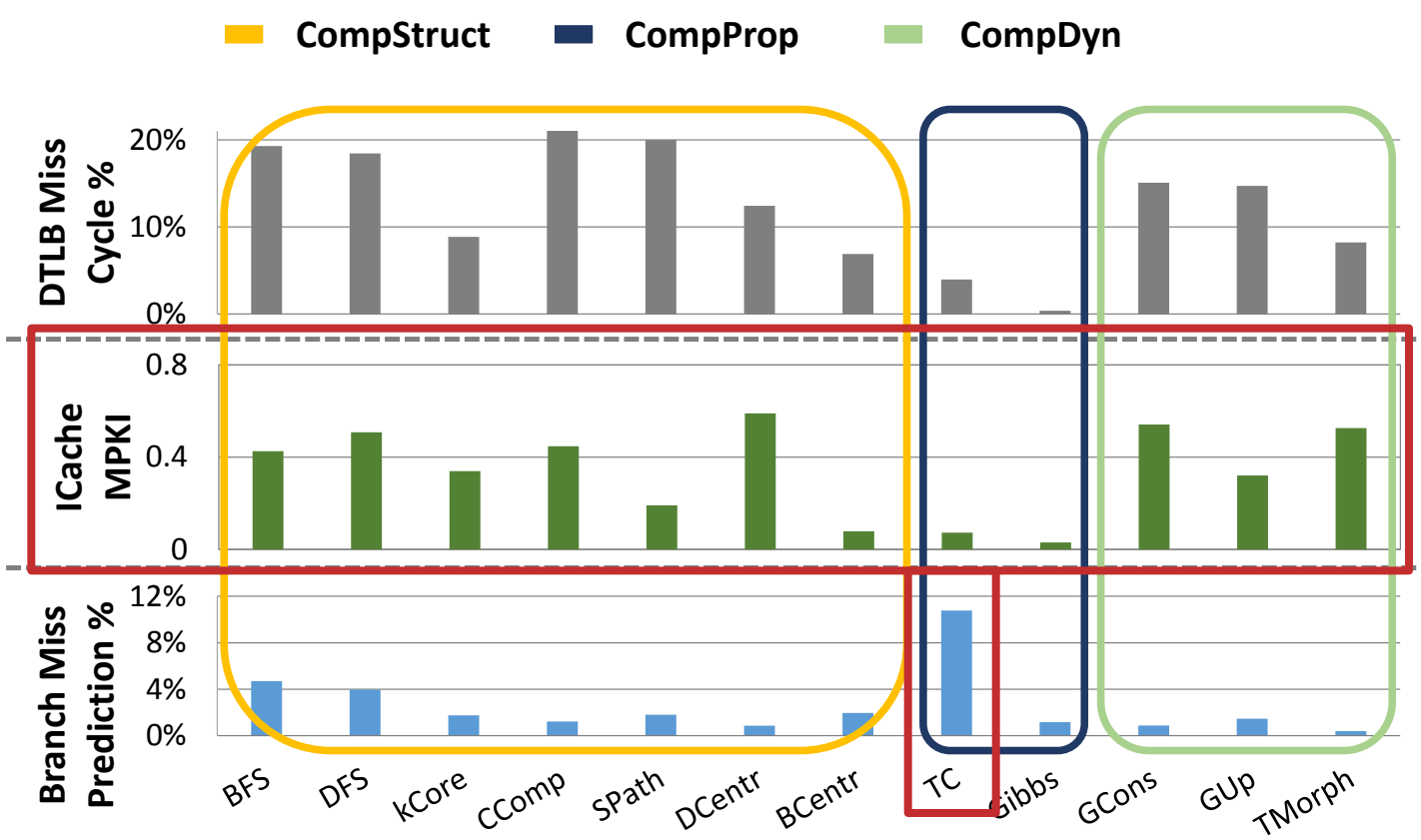
CPU: EXECUTION TIME BREAKDOWN

- Backend is the bottleneck: memory sub-system issue
  - CompProp is different: TC-triangle counting Gibss-gibbs inference



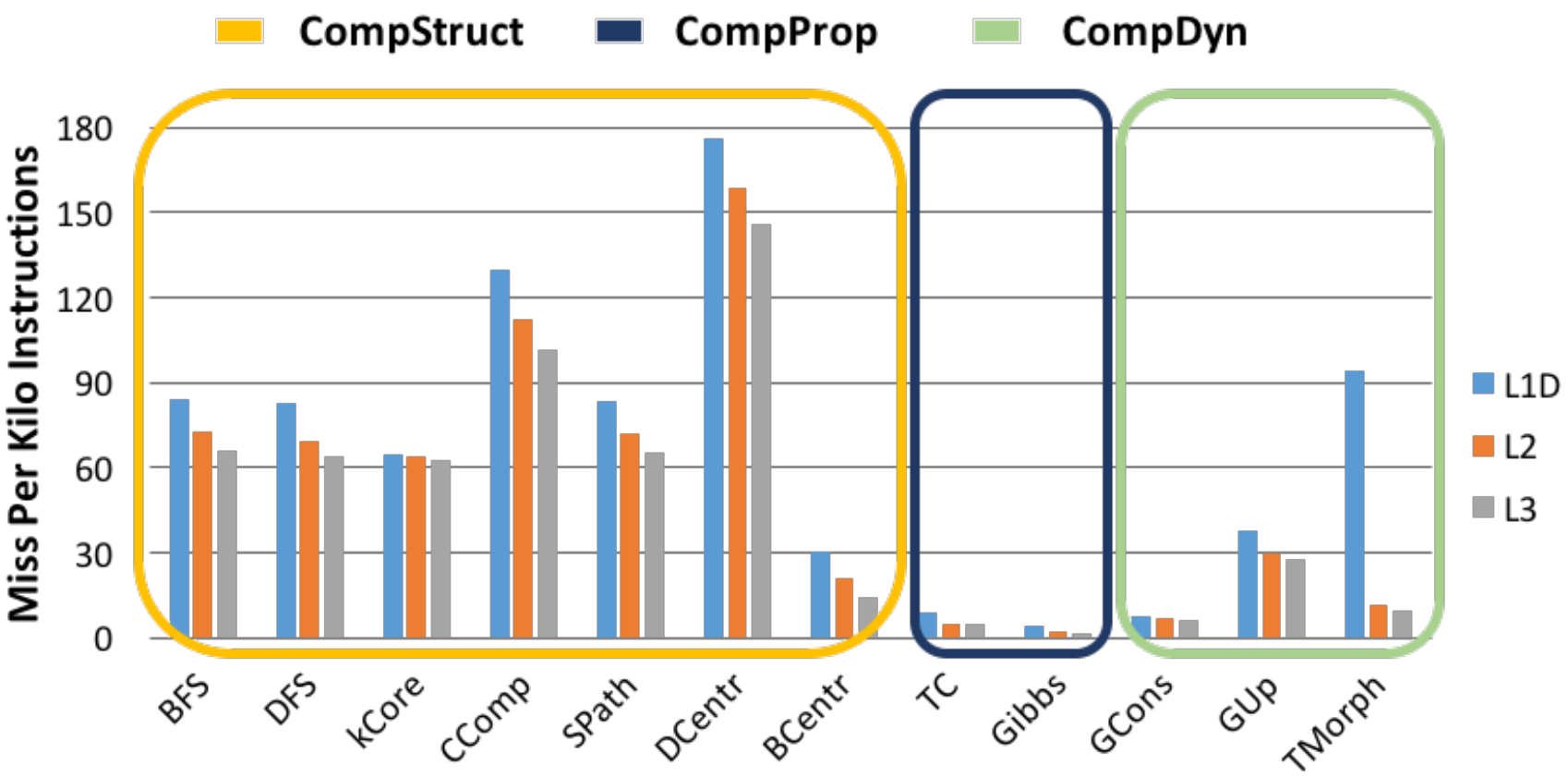
CPU: CORE ANALYSIS

- Significantly high DTLB penalty
- ICache and Branch prediction: not a major bottleneck



CPU: CACHE PERFORMANCE

- High cache MPKI because of irregular access pattern



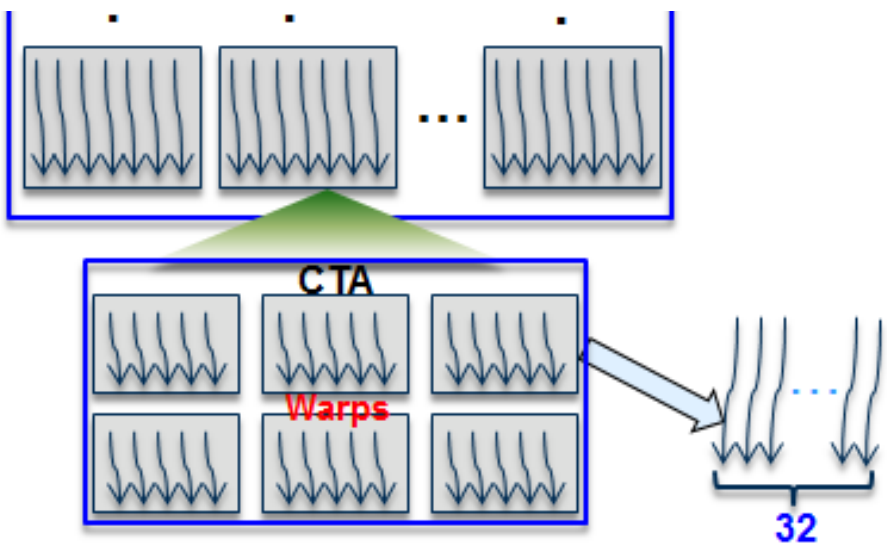
GPU DIVERGENCE

Branch divergence

- Branch divergence rate = inactive threads per warp/warp size

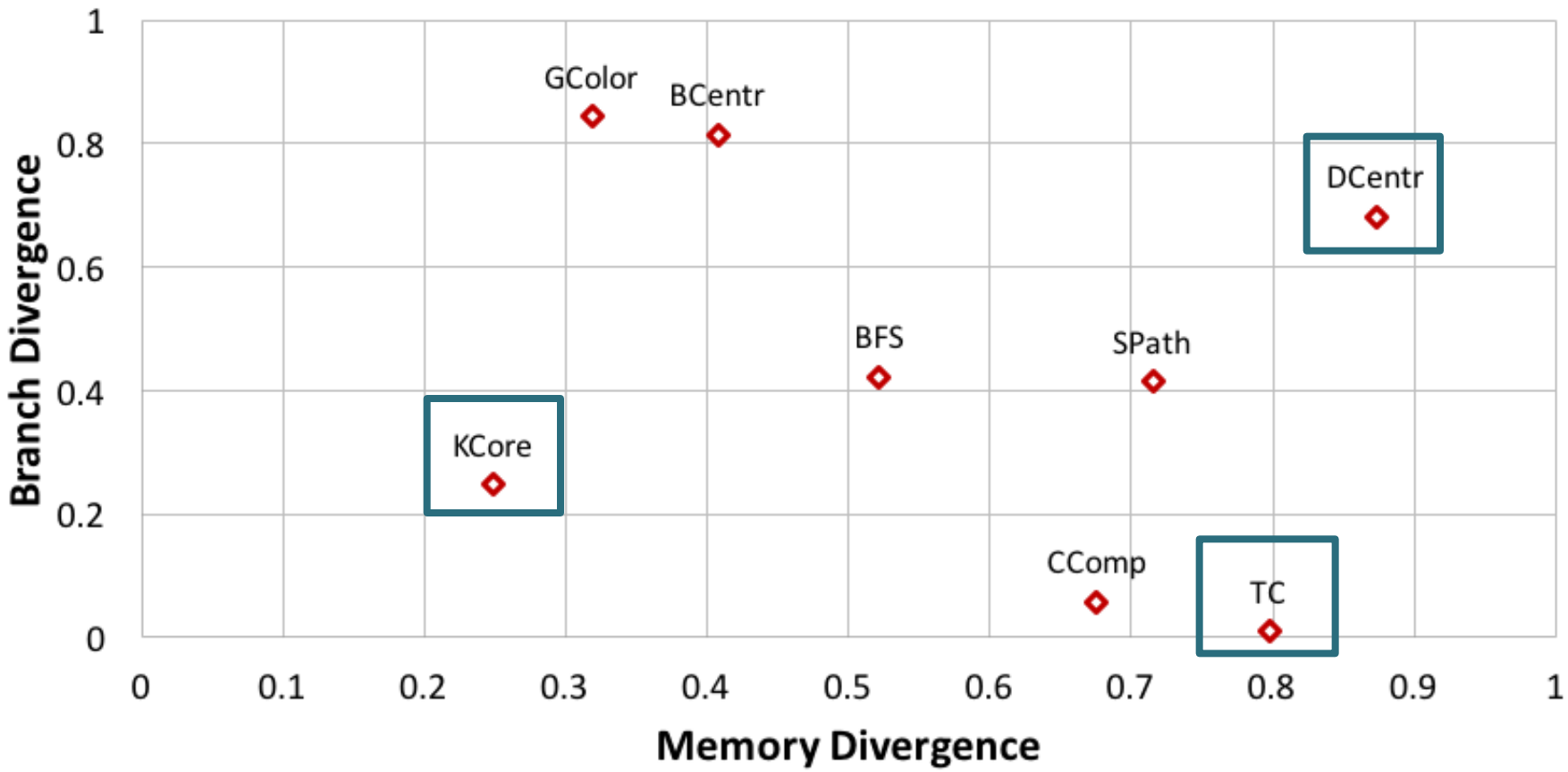
Memory divergence

- Memory divergence rate = replayed instructions/issued instructions



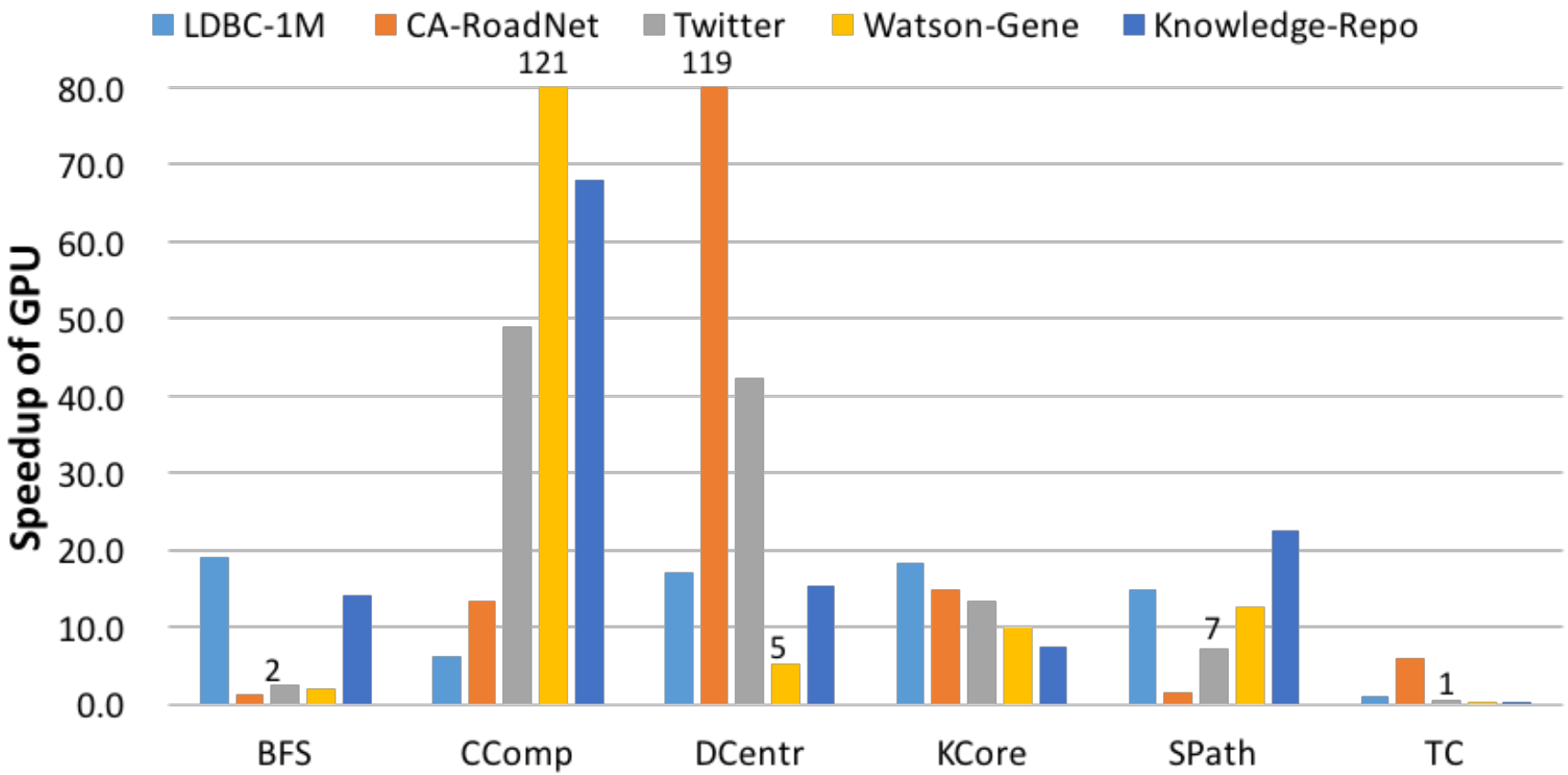
GPU DIVERGENCE

- High branch & memory divergence
- Diverse behaviors across workloads



GPU SPEEDUP

- Significant speedup over 16-core CPU



GRAPHBIG: TAKE AWAY

- Graph computing has a wide scope, not just BFS

- Multiple factors influence graph computing significantly, not only workload algorithms.
  - Framework, data representation, datasets

- Characterization
  - CPU: irregular access pattern -> poor cache performance
  - CPU: properly design code hierarchy can avoid ICache issue
  - GPU: memory and branch divergence issue
  - Diversity across workloads: both CPU and GPU sides

CONCLUSION

- Graph Computing has a wide scope. To understand it, we have to consider multiple key factors in a holistic way

- We proposed *GraphBIG,* a suite of CPU/GPU graph benchmarks based on real-world industrial practices, and characterized it on real machines comprehensively

- GraphBIG is open sourced (BSD license)
  - Check: https://github.com/graphbig
  - Workloads, datasets, and documents

THANK YOU!

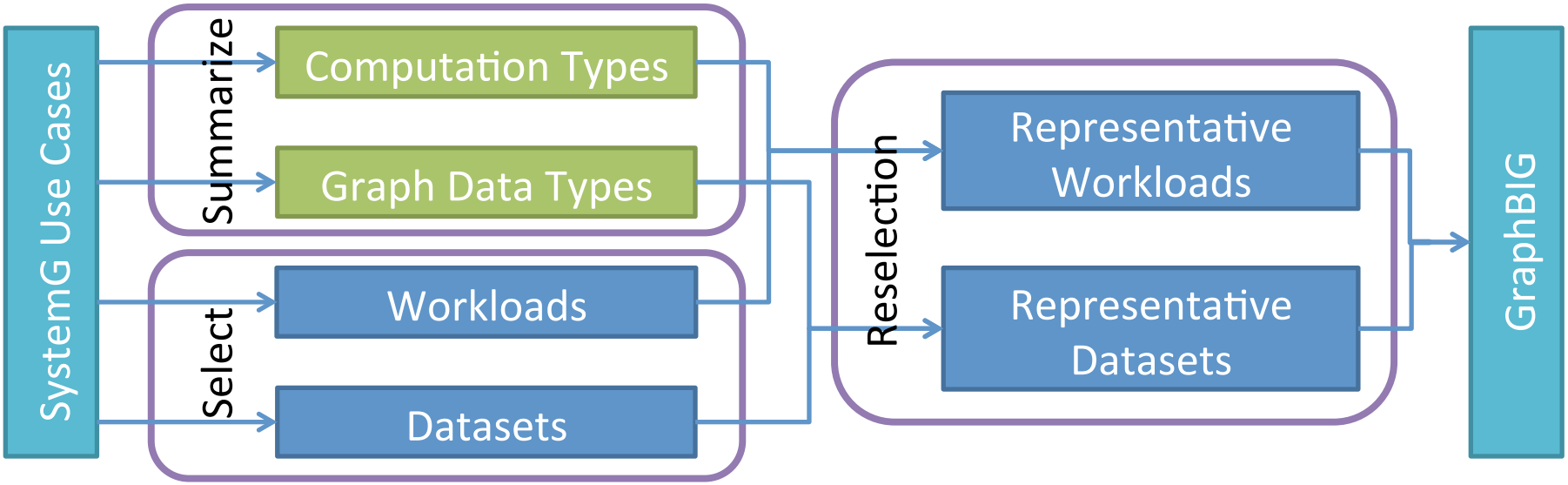
GraphBIG http://github.com/graphbig

IBM System G http://systemg.research.ibm.com/

HPArch Lab http://comparch.gatech.edu/hparch/

BACKUP SLIDES

WORKLOAD SELECTION



GRAPHBIG FEATURES

Design

- Framework: *property graph frame based on industrial practices*
- Representativeness: *workloads selected from real-world use cases*
- Coverage: *cover major computation types, much more than just traversal*
- CPU + GPU workloads

Code

- C++ code base: *requiring only c++0x*
- Standalone package: *no external package dependencies*
- Integrated profiling tool: *profiling via hardware performance counters*

GRAPHBIG HANDS-ON

Fetch Code

- Code: https://github.com/graphbig/graphBIG
- Doc: https://github.com/graphbig/GraphBIG-Doc

```
-bash:~$ git clone https://github.com/graphbig/graphBIG.git GraphBIG
Cloning into 'GraphBIG'...
remote: Counting objects: 497, done.
remote: Compressing objects: 100% (110/110), done.
remote: Total 497 (delta 57), reused 0 (delta 0), pack-reused 386
Receiving objects: 100% (497/497), 2.07 MiB | 0 bytes/s, done.
Resolving deltas: 100% (229/229), done.
Checking connectivity... done.
-bash:~$
```

GRAPHBIG HANDS-ON

Compile

- Require: gcc/g++ (>4.3), gnu make
- Just *"make all"*

```
-bash:~$ cd GraphBIG/
-bash:GraphBIG$ ls
benchmark  CHANGELOG.md  common  csr_bench  dataset  gpu_bench  LICENSE  openG  README.md  tools
-bash:GraphBIG$ cd benchmark/
-bash:benchmark$ ls
bench_betweennessCentr  bench_degreeCentr     bench_graphConstruct  bench_shortestPath   common.mk   ubench_add     ubench_traverse
bench_BFS               bench_DFS             bench_graphUpdate     bench_TopoMorph      Makefile    ubench_delete
bench_connectedComp     bench_gibbsInference  bench_kCore           bench_triangleCount  README.txt  ubench_find
-bash:benchmark$ make all
make -C ../tools all
make[1]: Entering directory `/home/lifeng/GraphBIG/tools'
rm -rf libpfm-4.5.0
tar xzvf libpfm-4.5.0.tar.gz
libpfm-4.5.0/./
libpfm-4.5.0/./COPYING
libpfm-4.5.0/./lib/
libpfm-4.5.0/./lib/pfmlib_ppc970.c
libpfm-4.5.0/./lib/pfmlib_powerpc.c
libpfm-4.5.0/./lib/pfmlib_sparc_priv.h
libpfm-4.5.0/./lib/pfmlib_powerpc_perf_event.c
```

GRAPHBIG HANDS-ON

Test Run

- Just *"make run"*
- Using default "small" dataset

```
-bash:benchmark$ cd bench_BFS/
-bash:bench_BFS$ make run
Running bfs, output in output.log
```

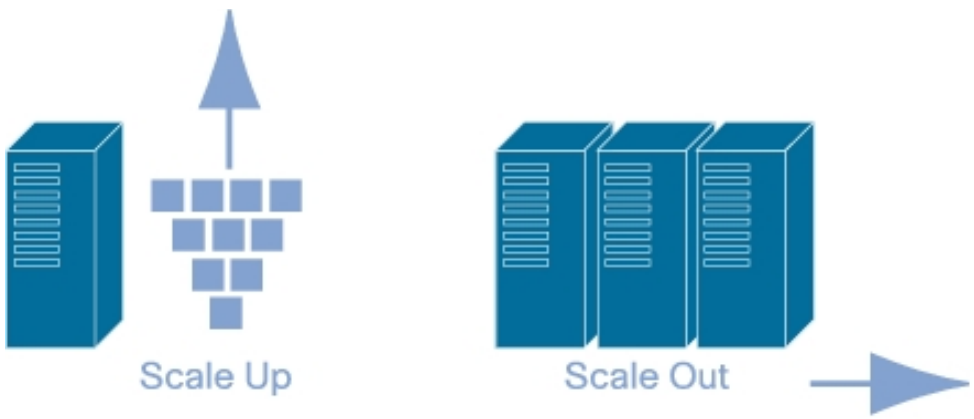
```
-bash:bench_BFS$ cat output.log
==================================================================
  _______                    .__   __________.___  _______
 /  ____/_________  _____ |  |__\_____   \  |/  ____/
/   \  __\_  __ \__  \ \___ \|  |  \|    |  _/  /   \  ___
\    \_\  \  | \// __ \|  |_> >   Y  \    |   \   \    \_\  \
 \_____  /__|  (____  /  __/|___|  /_____  /___|\_____  /
       \/           \/|__|       \/      \/           \/

==================================================================
Benchmark: BFS
loading data...
== 1000 vertices  29790 edges
== time: 0.0530188 sec

BFS root: 31
BFS finish:
== time: 0.00118506 sec
PERF_COUNT_HW_CPU_CYCLES        ==>     2581036
PERF_COUNT_HW_INSTRUCTIONS      ==>     774222
PERF_COUNT_HW_BRANCH_INSTRUCTIONS       ==>     200529
PERF_COUNT_HW_BRANCH_MISSES     ==>     10486
PERF_COUNT_HW_CACHE_L1D_READ_ACCESS     ==>     309284
PERF_COUNT_HW_CACHE_L1D_READ_MISS       ==>     99740
==================================================================
```

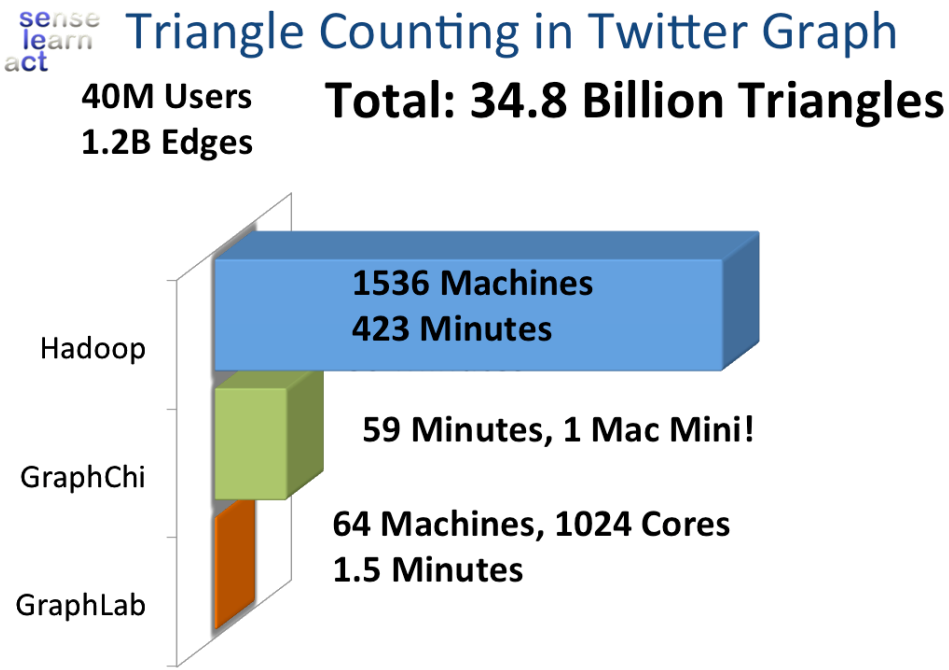
More Datasets

- Download: https://github.com/graphbig/graphBIG/wiki/GraphBIG-Dataset
- Untar and specify the correct path in benchmark argument *"--dataset"*
- Other 3rd party datasets (csv format) are also possible

SCALE UP VS. SCALE OUT

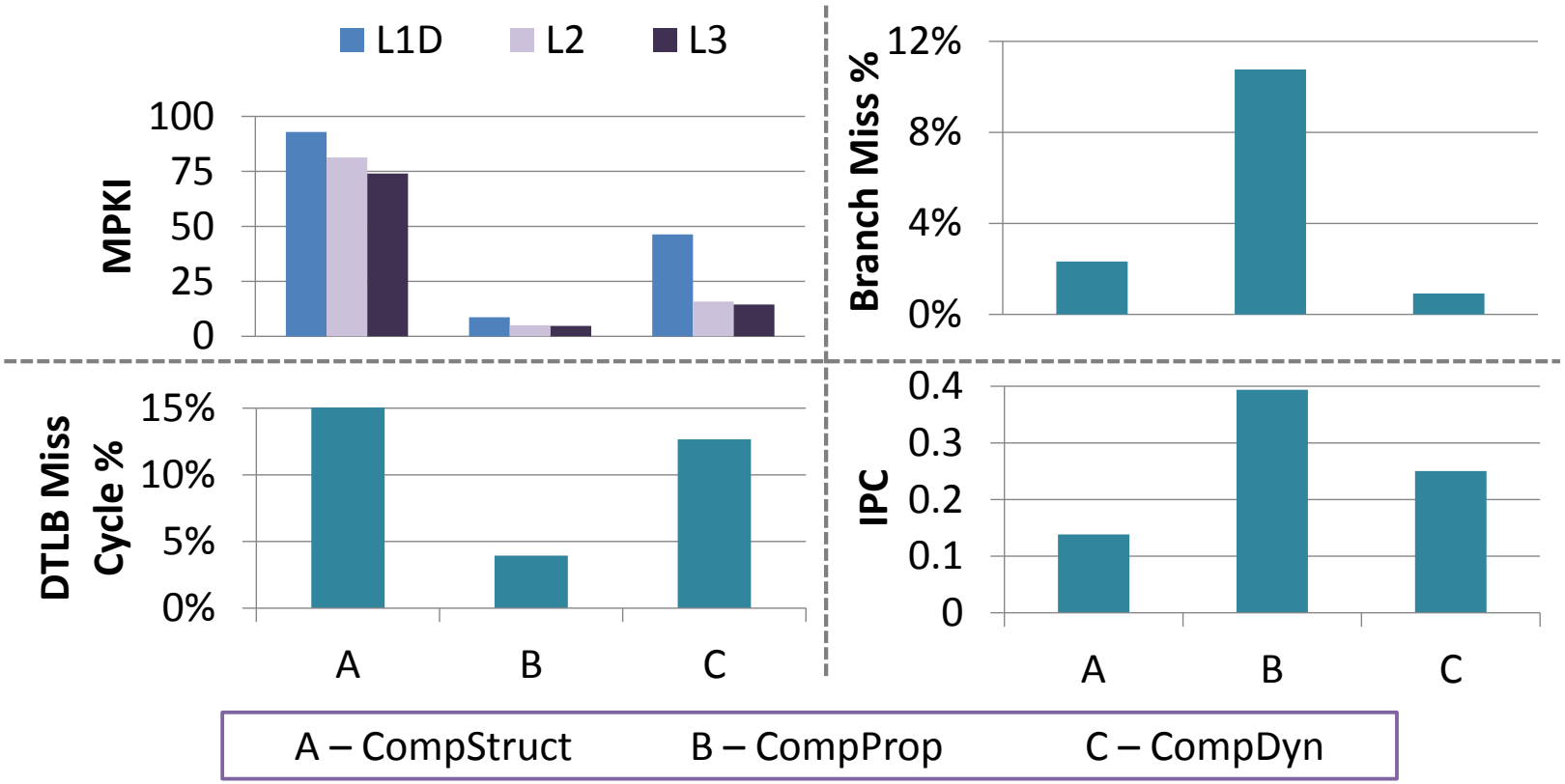
- Scale up before Scale out



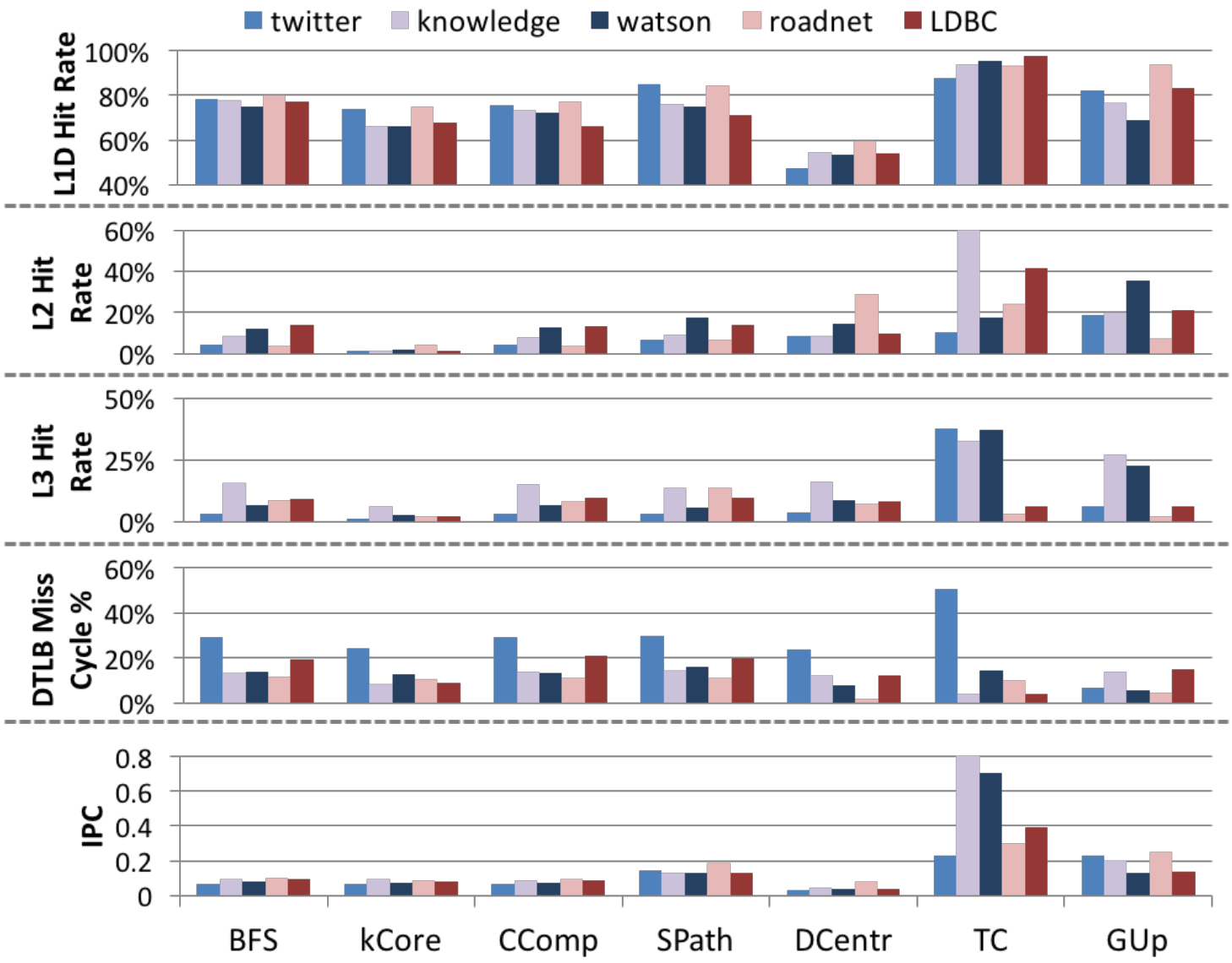


COMPUTATION TYPE BEHAVIOR

- Diverse behaviors across different computation types

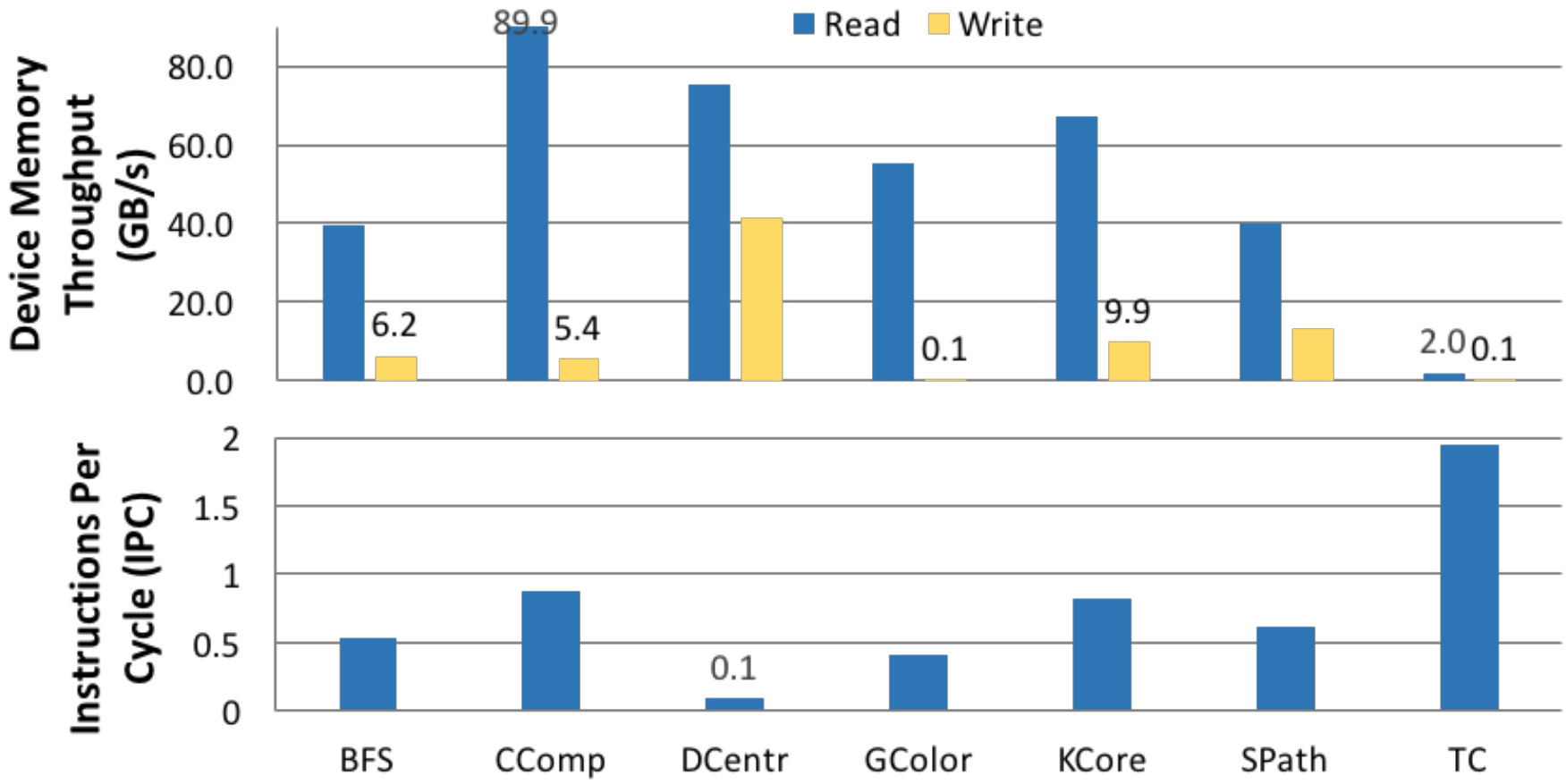


CACHE BEHAVIORS



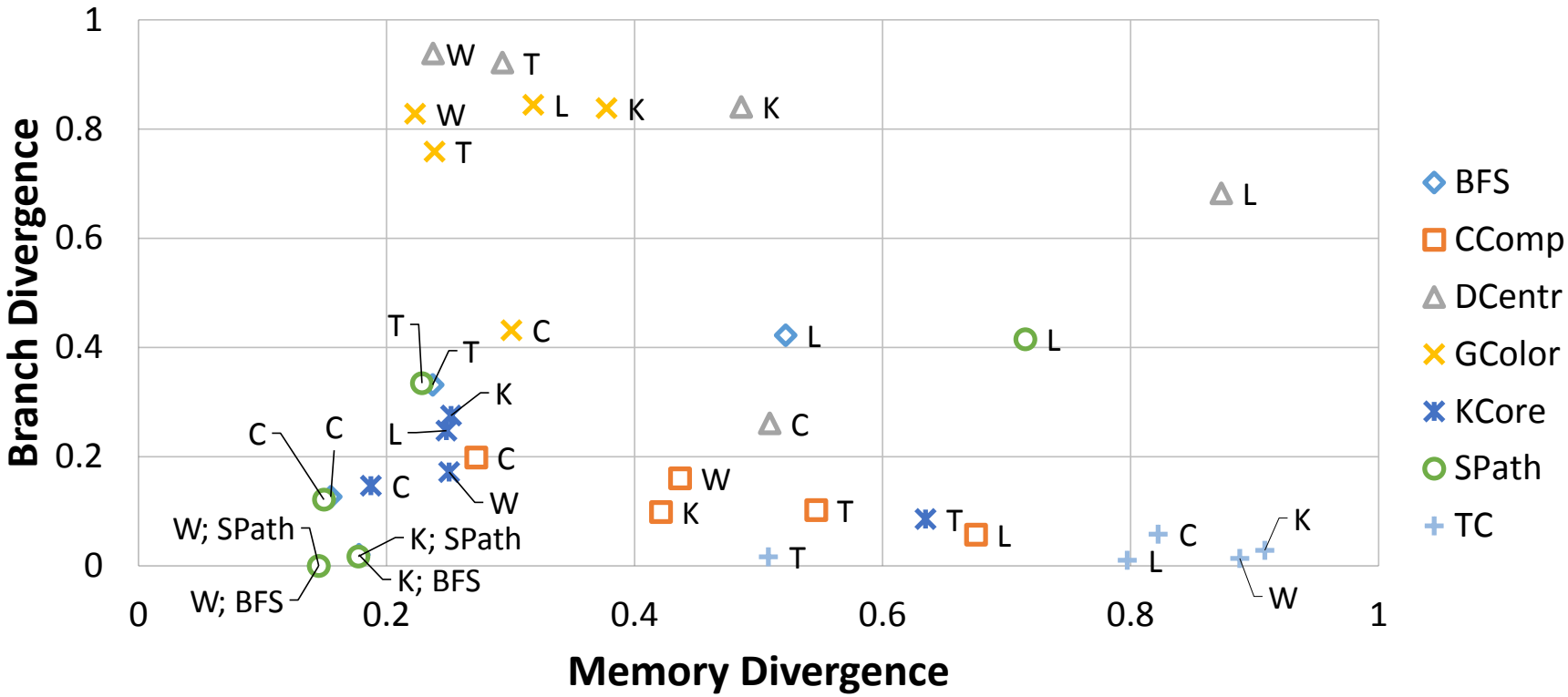
GPU ARCH BEHAVIOR

- Cannot fully utilize available memory bandwidth
- Significantly low IPC



GPU DATA SENSITIVITY

- Sensitive to input data
- Memory divergence shows higher sensitivity



L: LDBC-1M	C: CA-RoadNet	T: Twitter	W: Watson-Gene	K: Knowledge-Repo