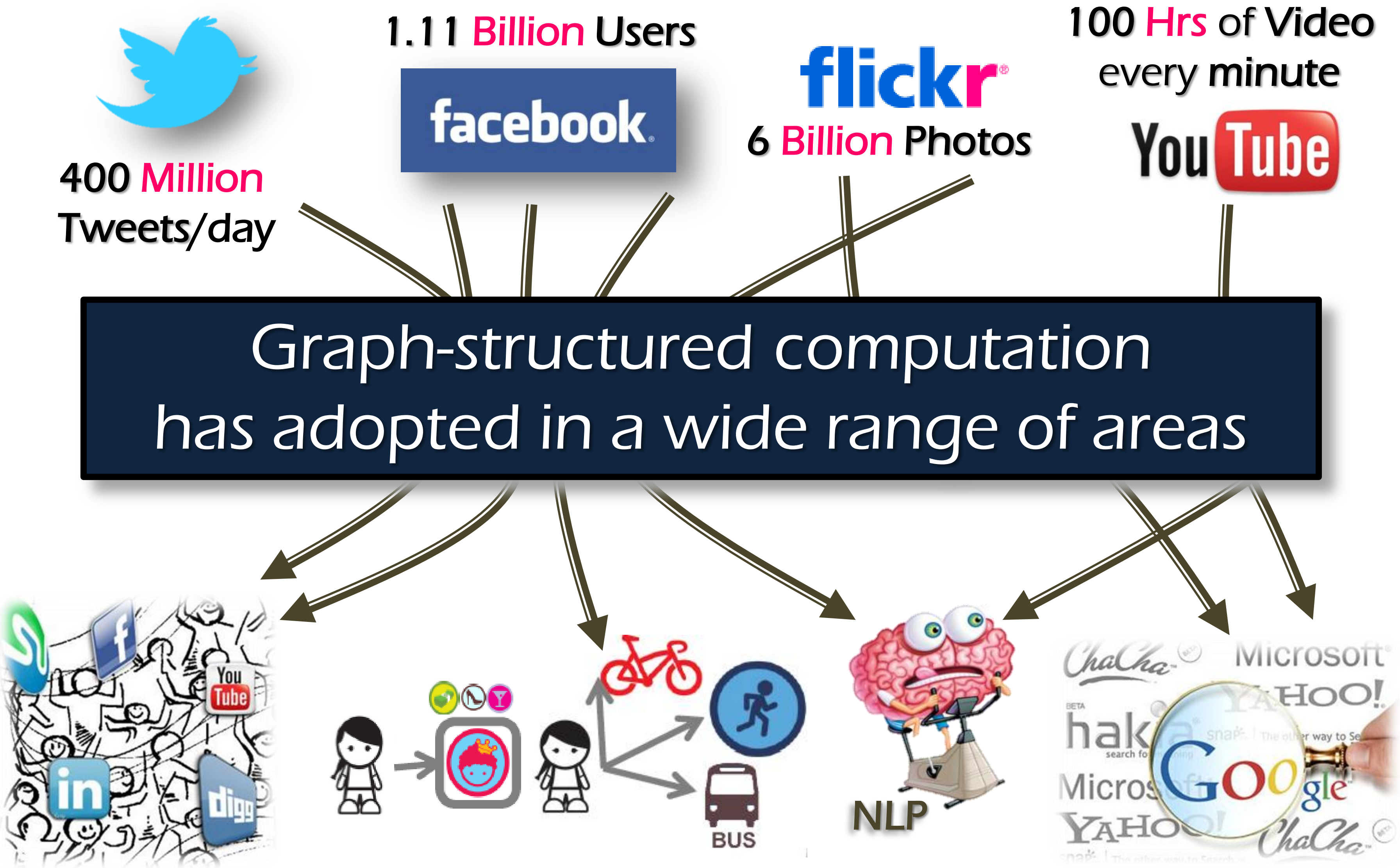
SYNC or ASYNC ? Time to Fuse for Distributed Graph-Parallel Computation

Chenning Xie[+], Rong Chen[+], Haibing Guan[*], Binyu Zang[+] and Haibo Chen[+]

Institute of Parallel and Distributed Systems [+]
Department of Computer Science [*]
Shanghai Jiao Tong University

Big Data → Graph Computation



400 Million Tweets/day

1.11 Billion Users

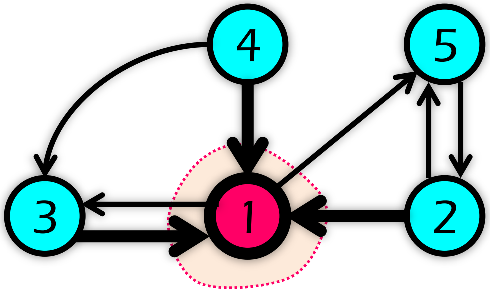
6 Billion Photos

100 Hrs of Video every minute

Graph-structured computation has adopted in a wide range of areas

Graph-parallel Computation

"Think as Vertex", e.g. **PageRank** : $R_i = \alpha + (1-\alpha) \sum_{i,j \in E} W_{i,j} R_j$



Characteristics

□ Linked set → data dependence

□ Rank of who links it → predictable accesses

□ Convergence → iterative computation

Distributed Graph Computation

- □ Larger Graph
- □ Complicated Computation
- □ Storage support

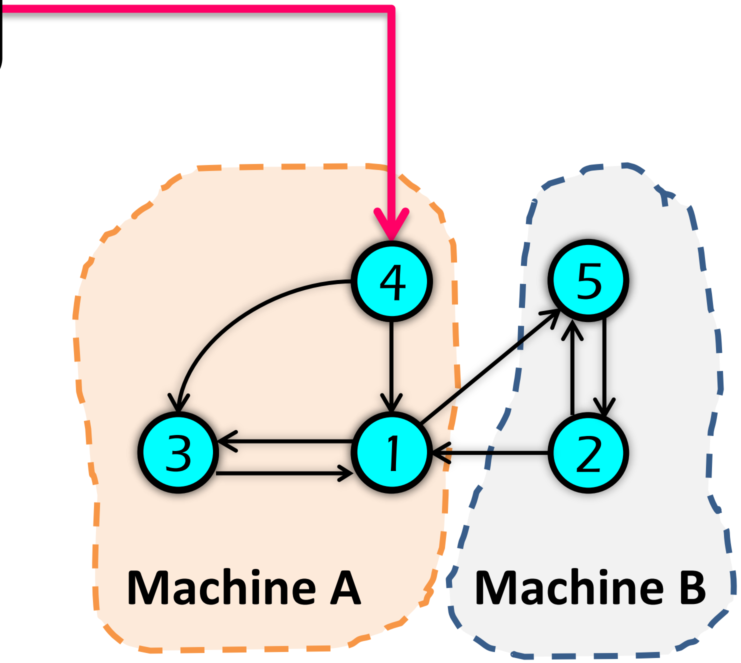
Distributed Graph Computation

User define the logic (e.g. Pagerank) :

□ Input: R_j (Data of neighbor j)

□ Compute():

$$Vertex\_Data = \alpha + (1-\alpha)\sum W_j R_j$$



Framework:

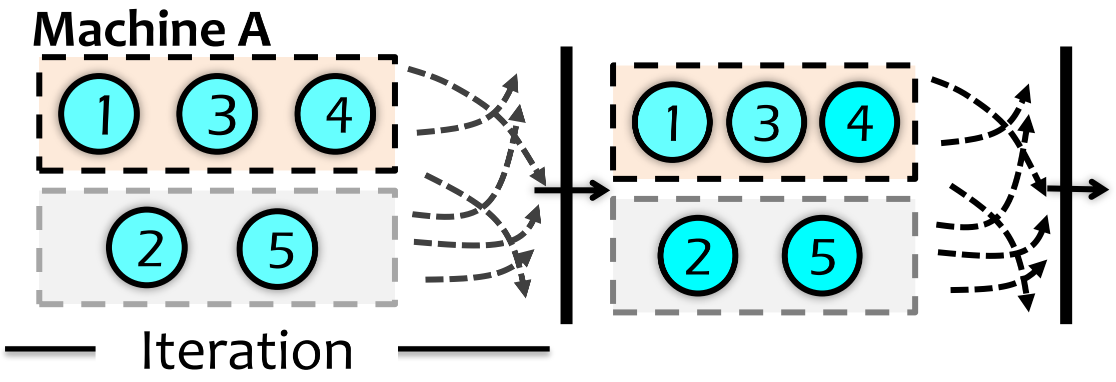
□ Load & partition over **cluster**

□ **Schedule** compute() **Repeatedly** until get convergence

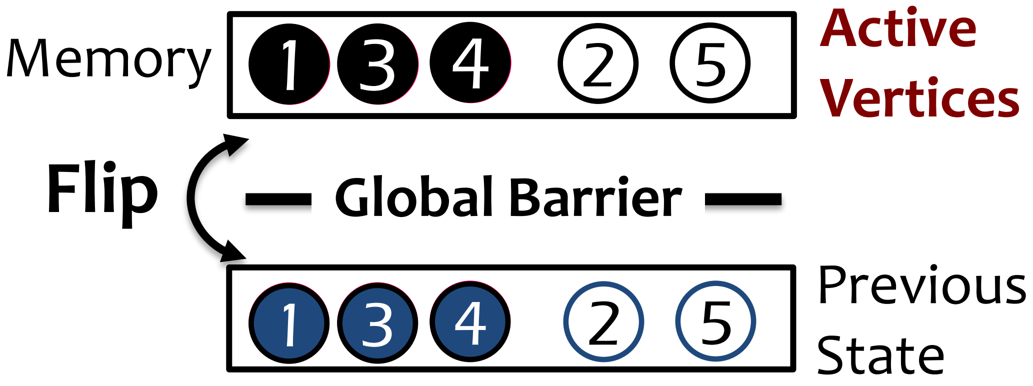
Existing Scheduling Modes - Synchronous

(Sync Mode)

Scheduling:



Internal state (e.g. Machine A):



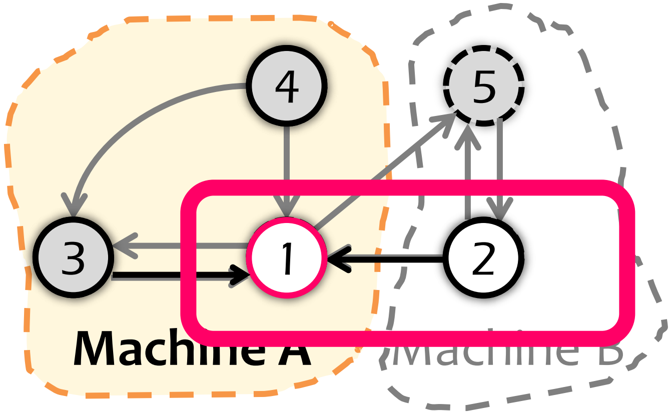
Pseudocode

```
while (iteration ≤ max) do
    if $V_a == \emptyset$ then break
    $V'_a \leftarrow \emptyset$
    foreach $v \in V_a$ do
        $A \leftarrow$ compute(v)
        $V'_a \leftarrow V'_a \cup A$
    barrier to update
    $V_a \leftarrow V'_a$
    iteration ++
```

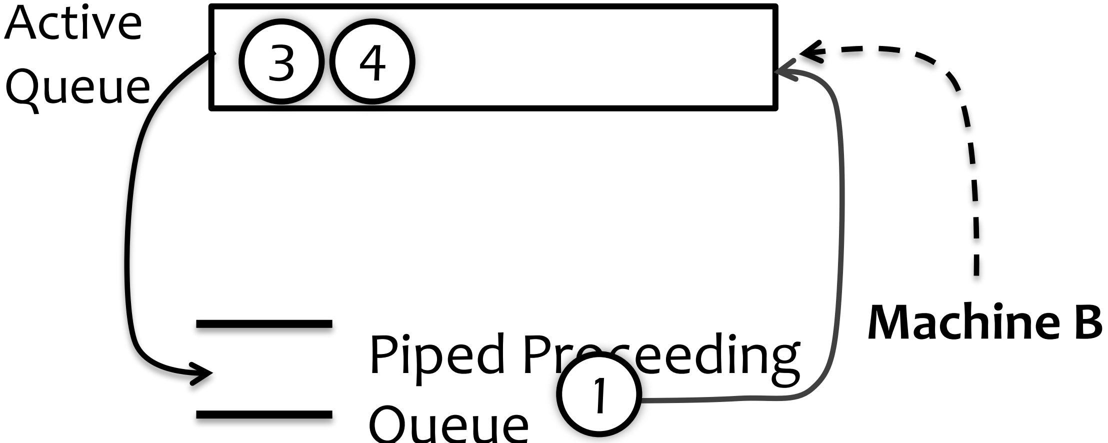
Existing Scheduling Modes - Asynchronous

(Async Mode)

Scheduling:



Internal State (e.g. Machine A):



Active Queue

Piped Proceeding Queue

Machine B

Pseudocode

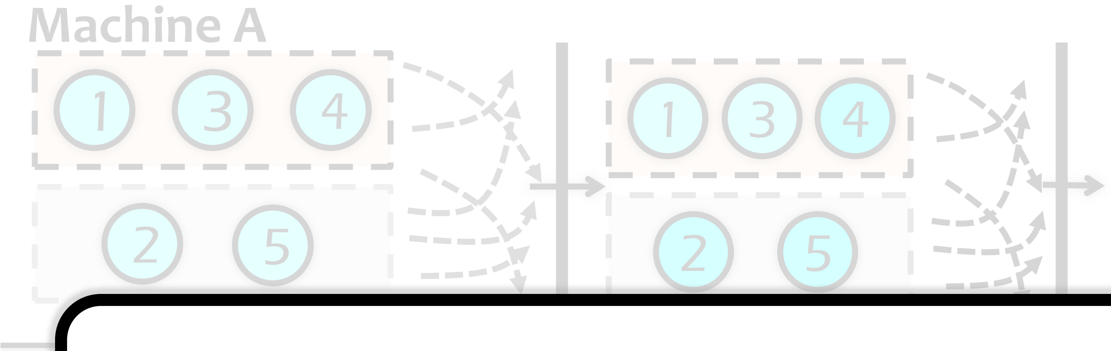
```
while ($V_a$ != ∅) do
    v = dequeue( $V_a$ )
    A ← compute(v)
    $V'_a$ ← $V'_a$ ∪ A
    signal across machines
```

Propagate ASAP,

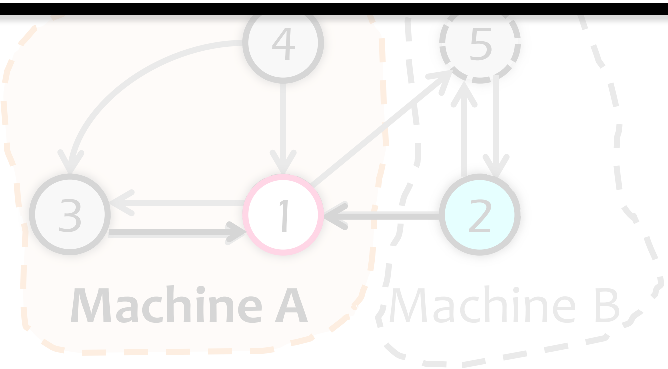
to converge faster

Existing Scheduling Modes

Synchronous



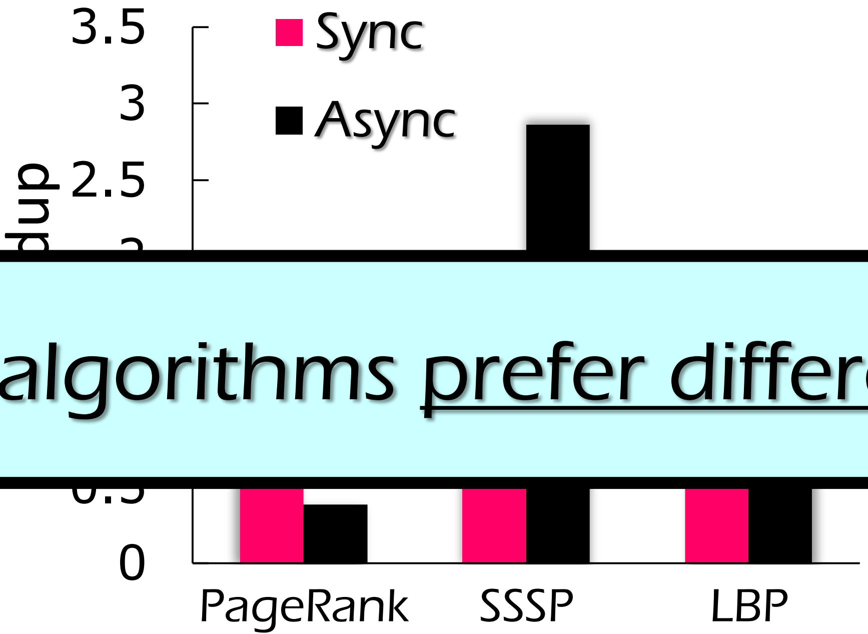
Which could get a **<u>better</u> performance**?



Asynchronous

Algorithms: Sync vs. Async

- □ Same Configuration + <u>Different Algorithms</u> ?



Different algorithms <u>prefer different modes</u>

<u>Large active vertex set</u> with collecting all data from neighbors

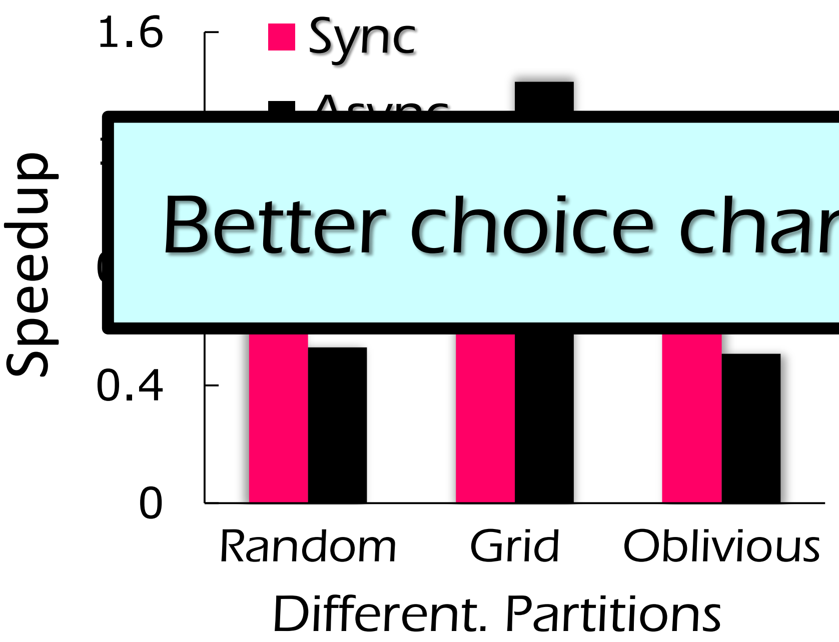
Require **<u>fast broadcast</u>** of shortest path value

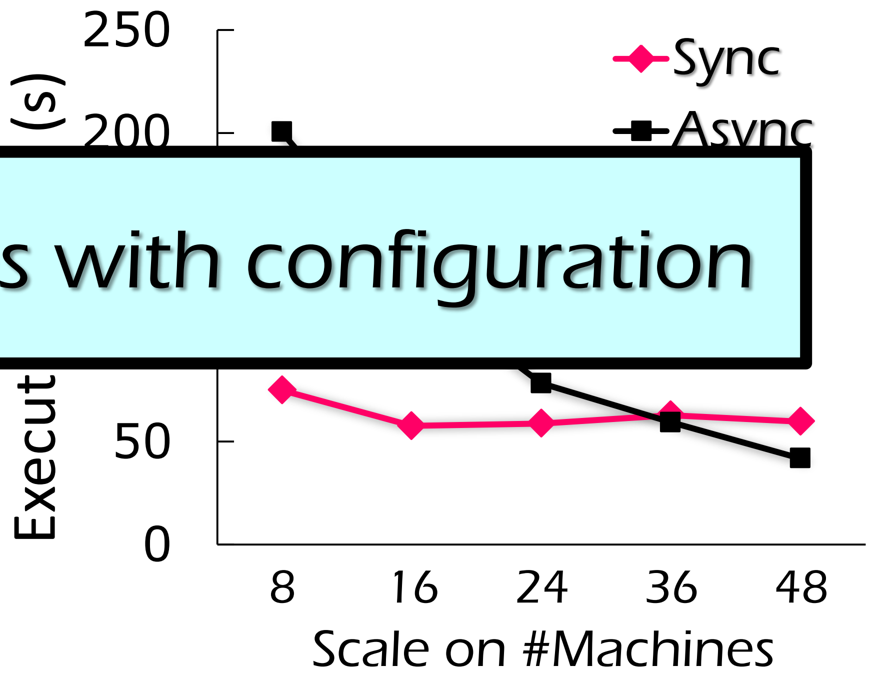
Belief Propagation Algorithm

Configuration: Sync vs. Async

□ Same Configuration + Different Algorithms: Uncertain

□ Different Configuration + Same Algorithms (LBP) ?





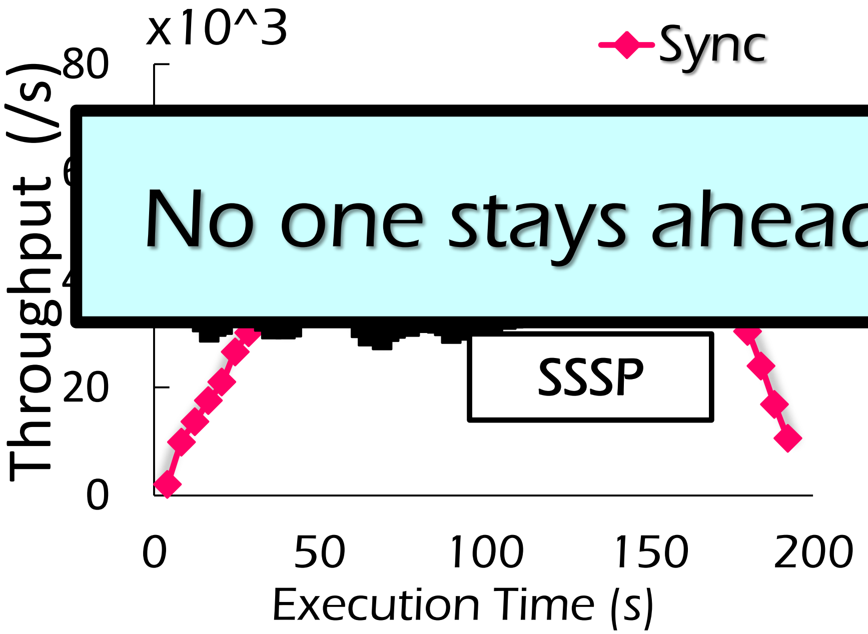
Better choice changes with configuration

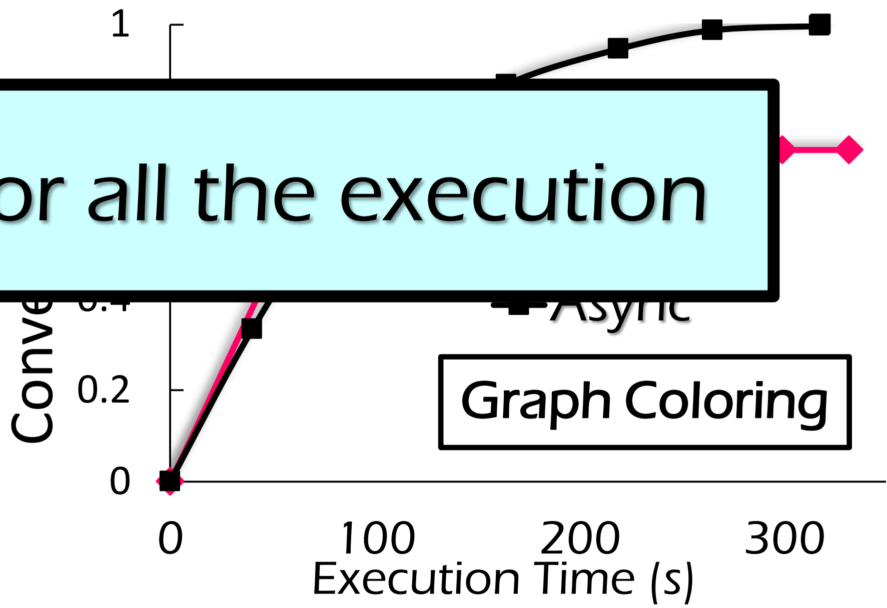
Partition methods affect load balance & communication

Sync mode batches heavy load , Async mode scales better.

Stages: Sync vs. Async

- □ Same Configuration + Different Algorithms: Uncertain
- □ Different Configuration + Same Algorithms: Uncertain
- □ Same Configuration + Same Algorithm ?





No one stays ahead for all the execution

Async mode starts **faster,**
Sync mode grows with **a peak**.

Sync **faster** but not converge,
Async slower but **converged**.

Summery: Sync vs. Async

Properties	SYNC	vs.	ASYNC
→ Communication	Regular		Irregular
→ Convergence	Slow		Fast
→ Workload	Heavyweight		Lightweight
→ Scalability	| Graph |		| Machines |

Better choice is **Unintuitive**

Single mode alone may be still **Suboptimal**

Contributions

First comprehensive study on Sync & Async modes

PowerSwitch – *adaptive, fast & seamless switches*

Hybrid Execution Mode (**Hsync** Mode):

- □ Dynamically and **transparently** support the correct mode switches

Switch Timing Model:

- □ Determine the more efficient mode combined with online sampling, offline profiling and heuristics

Agenda

How to Switch - the Hsync mode

- □ Internal state conversion
- □ Consistency & correctness

When to Switch – the timing model

- □ Performance Metrics
- □ Current mode prediction
- □ The other mode estimation

Implementation

Evaluation

Challenges of switches

- Convert state at Consistent switch points

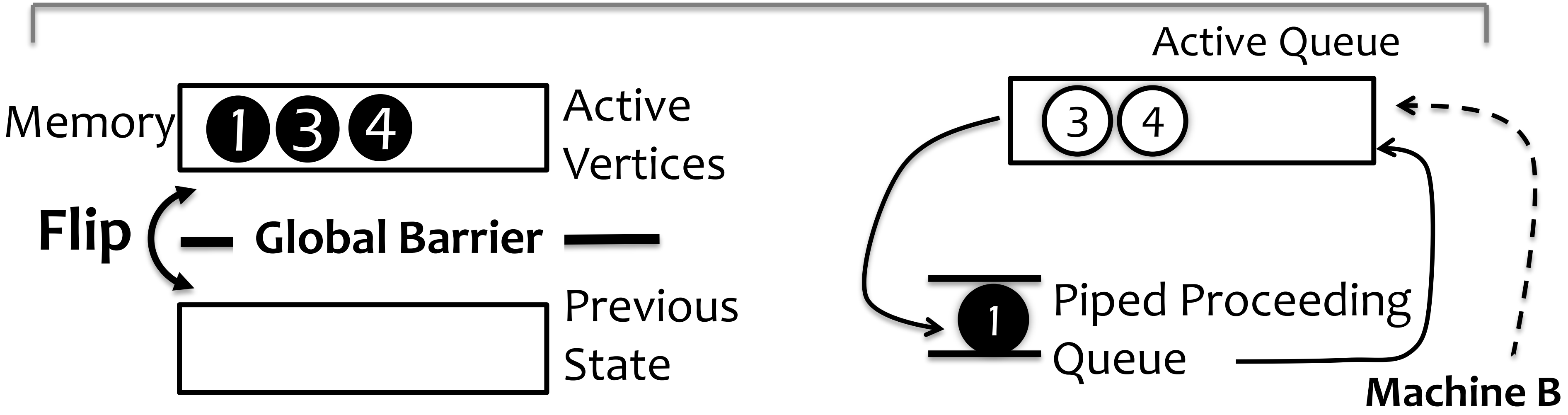
Sync mode

- □ Vertex update: **unordered**
- □ Flip in **global barrier**

Async mode

- □ **Priority/FIFO queue**
- □ Dequeue and enqueue

Internal state of one machine

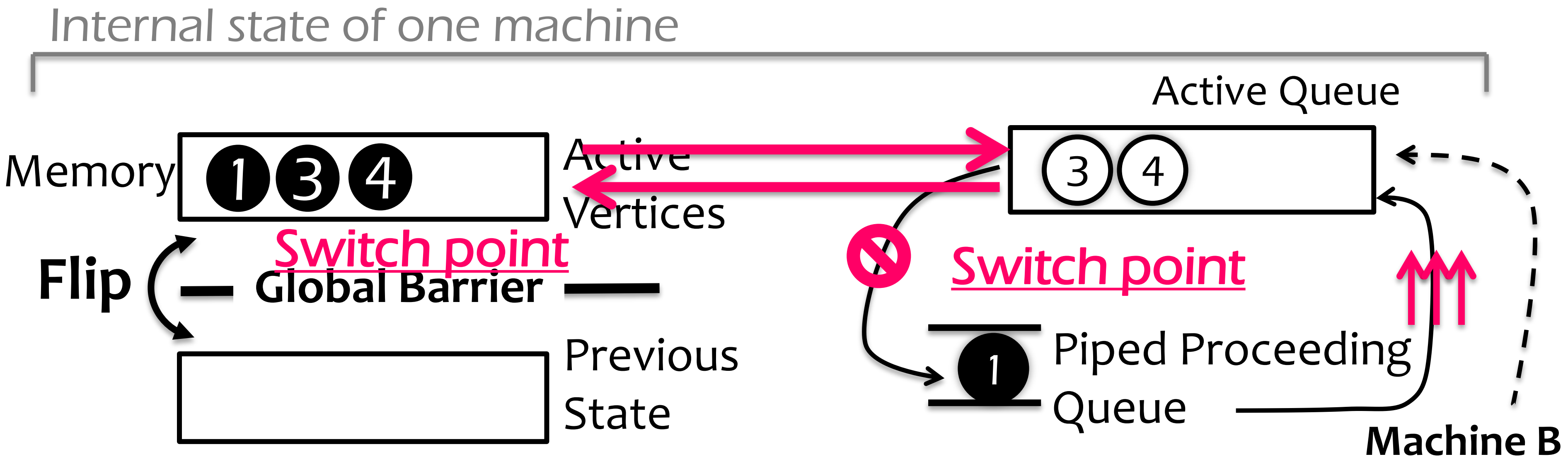


~~Challenges of switches~~ Hsync mode

Consistent switch points :

- □ Sync -> Async: global barrier
- □ Async -> Sync: suspend & wait

State transfer: active vertex set



Agenda

How to Switch - the Hsync mode

- □ Internal state conversion
- □ Consistency & correctness

When to Switch – the timing model

- □ Performance Metrics
- □ Current mode prediction
- □ The other mode estimation

Implementation

Evaluation

Switch timing - *affected by lots of factors*

Challenges:

□ How to quantify the real-time performance?

□ How to obtain the metrics？

Performance Metrics

□ Throughput $= \dfrac{|V_{compute}|}{T_{interval}} * \boldsymbol{\mu}$

Convergence ratio μ = $\dfrac{|NumTask_{async}|}{|NumTask_{sync}|}$

by sampling specific input pattern,

e.g. power-law, large diameter, high density…

Predict Throughput for Current mode

Sync

- Iteration as interval
- Throughput

$|V_{next}|$

Async

- Constant interval
- Throughput

1

Calculate the next interval based on:

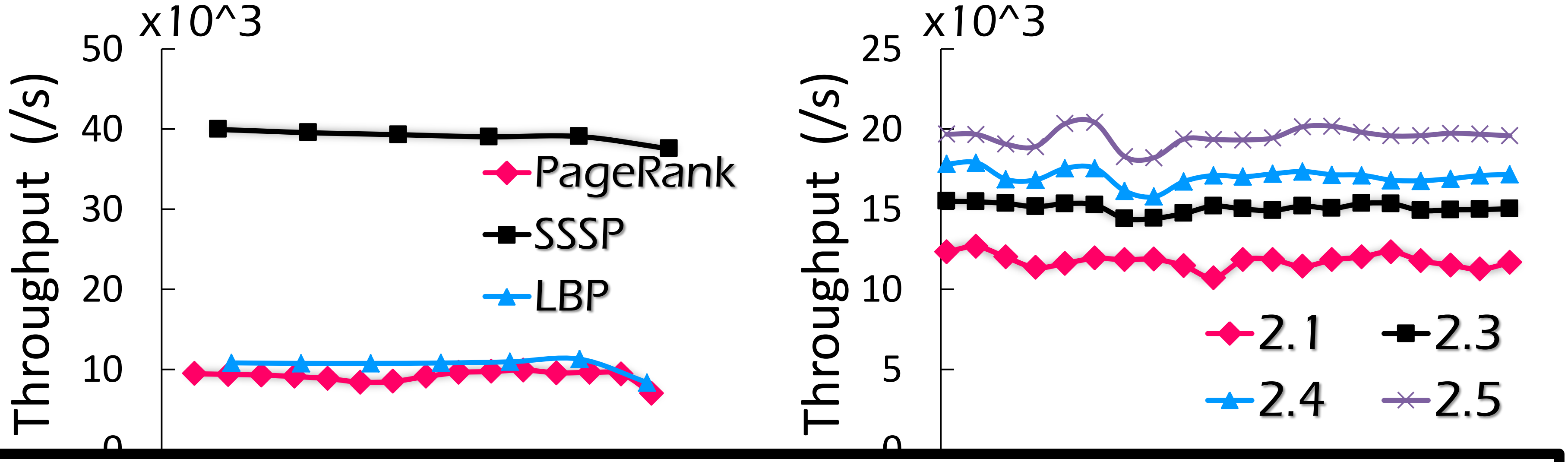
Current + History accumulation

$$T_{vert_{next}} = \alpha \cdot T_{vert_{current}} + (1 - \alpha) \cdot T_{vert_{history}}$$

Predict for Other offline mode

No more execution information

Predict **Async** when in sync mode:



Solution

Online sampling : on subset of input in Async before start

Offline profiling : build Neural Network model, refer to paper

Predict for Other offline mode

Predict **Sync** when in async mode:

- □ Hard to predict exactly
- □ Heuristic: Sync makes high utilization of resource.

$\text{Thro}_{\text{Sync}} > \text{Thro}_{\text{Async}}$, if workload is enough

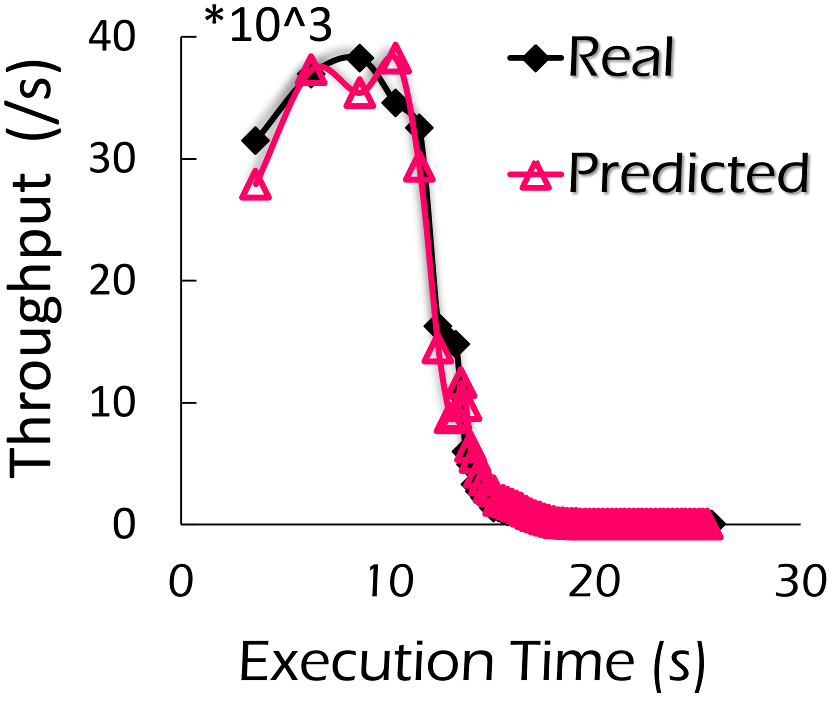
Condition:

1. Number of active vertices increases
2. Workload : $\dfrac{|V_{\text{new}}|}{T} > \text{Thro}_{\text{Async}}$

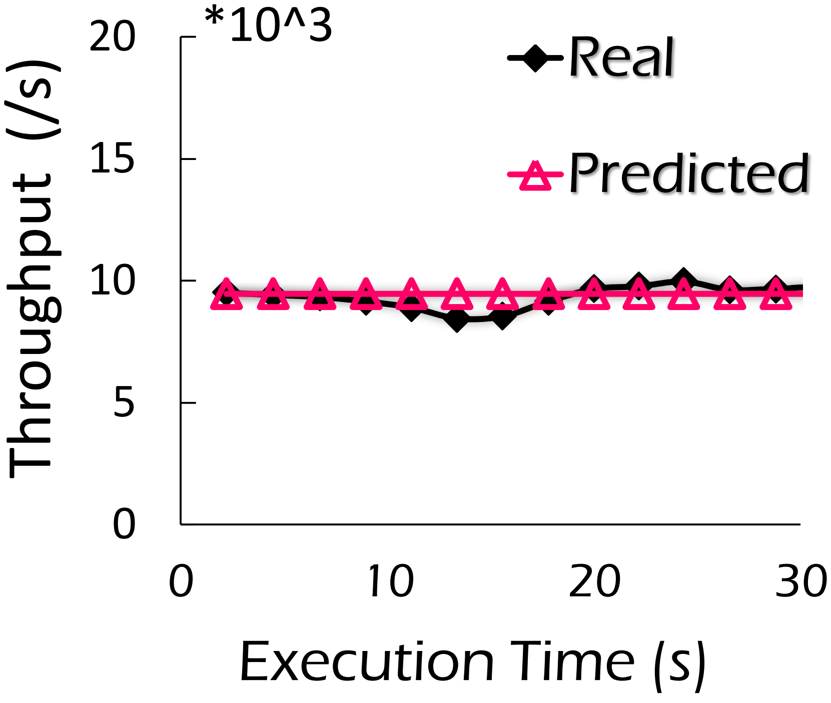
Async -> Sync

Prediction Accuracy

PageRank: Predicted throughput vs. Real sampled



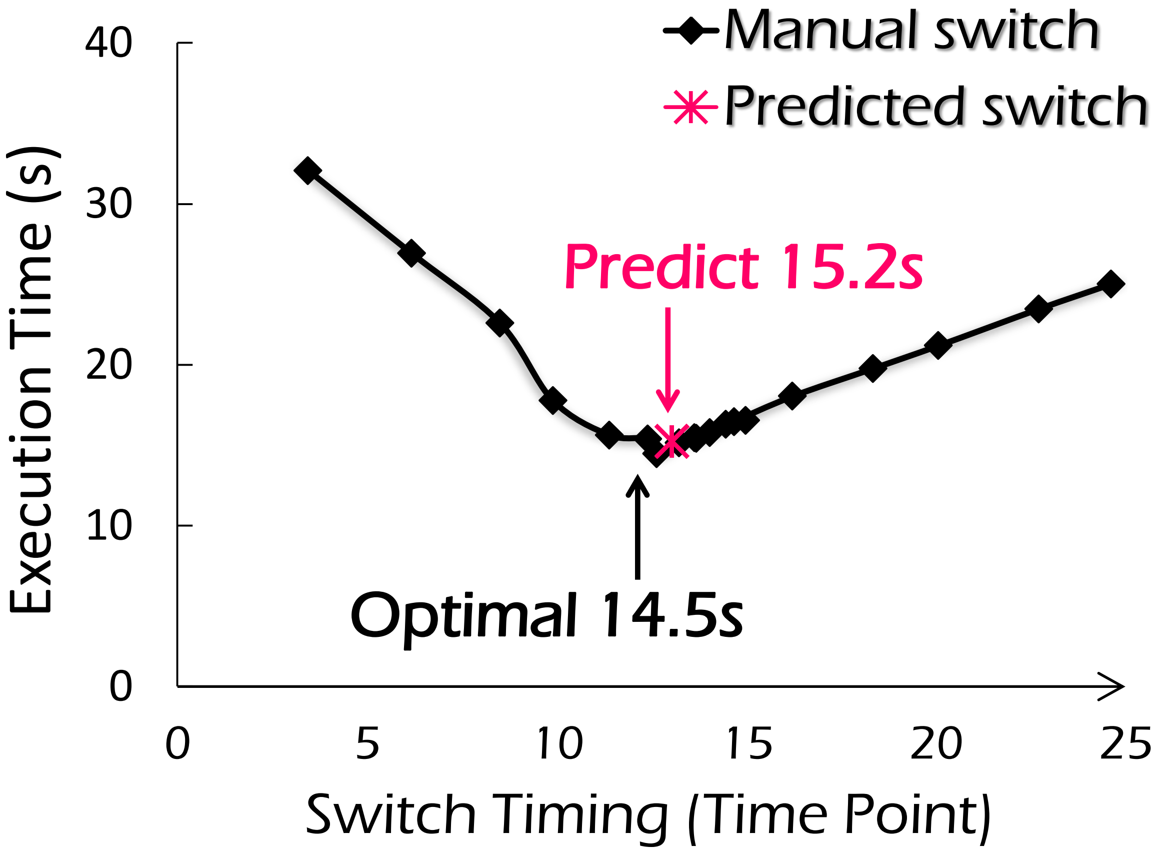
Sync mode



Async mode

Prediction Accuracy

PageRank: Predicted switch timing vs. Optimal



Implementation

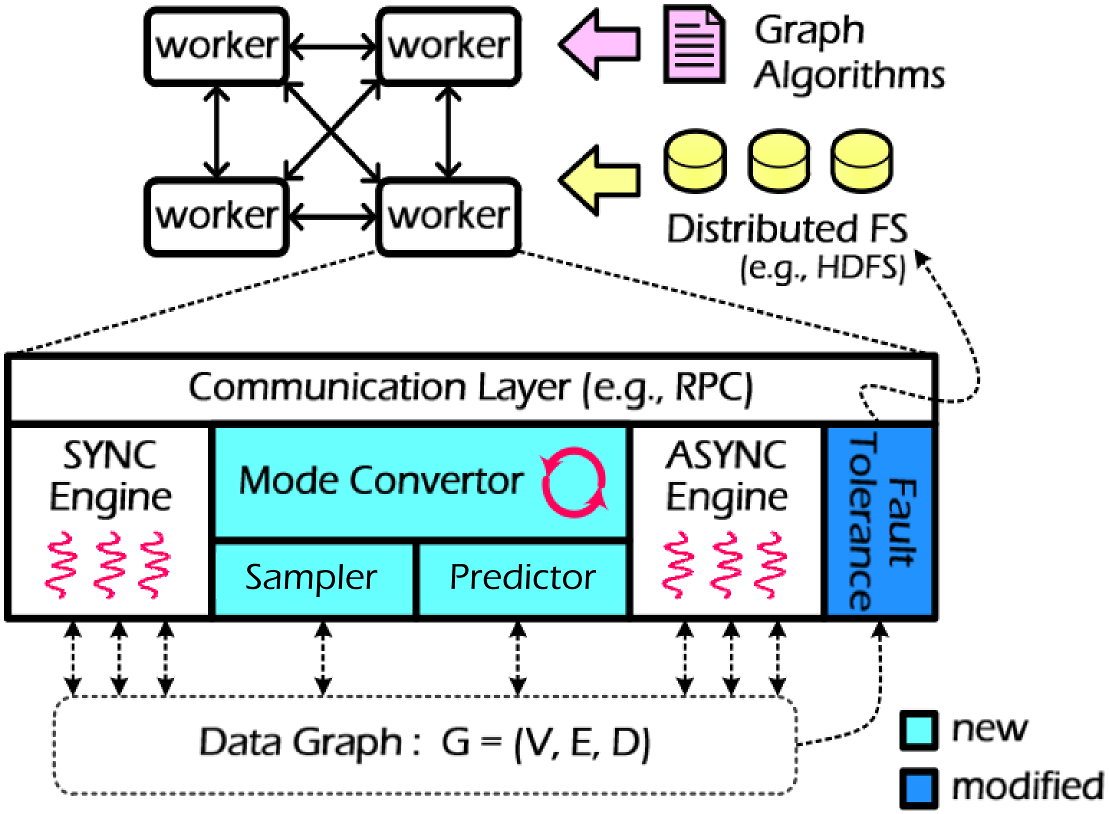
PowerSwitch:

- □ Based on latest GraphLab (PowerGraph) v2.2 with both Sync & Async modes.
- □ Provide the same graph abstraction transparent & compatible to all apps of GraphLab

Open Source

http://ipads.se.sjtu.edu.cn/projects/powerswitch.html

Implementation - Architecture



New

- Mode switcher
- Sampler
- Predictor

Extension

- Fault tolerance

Evaluation

Baseline: original SYNC & ASYNC mode

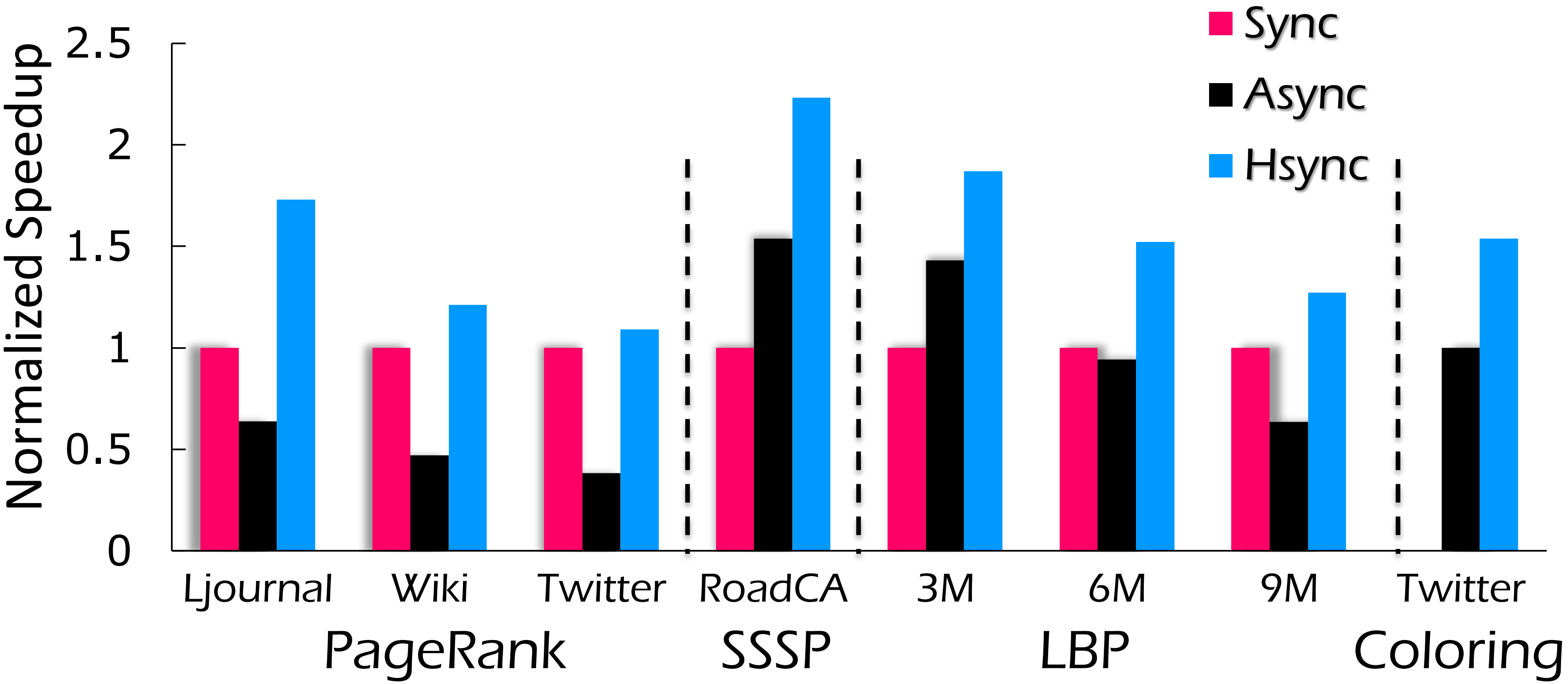
Configuration

- 48-node EC2-like cluster (VM based).
- Each node has 4 AMD Opteron cores, 12GB of RAM, connected with 1 GigE network.

Algorithms and Data Set

Algorithm	Graph	|V|	|E|
PageRank	LJournal	5.4M	79M
	Wiki	5.7M	130M
	Twitter	42M	1.47B
LBP	SYN-ImageData	1-12M	2-24M
SSSP	RoadCA	1.9M	5.5M
Coloring	Twitter	42M	1.47B

Performance Overview

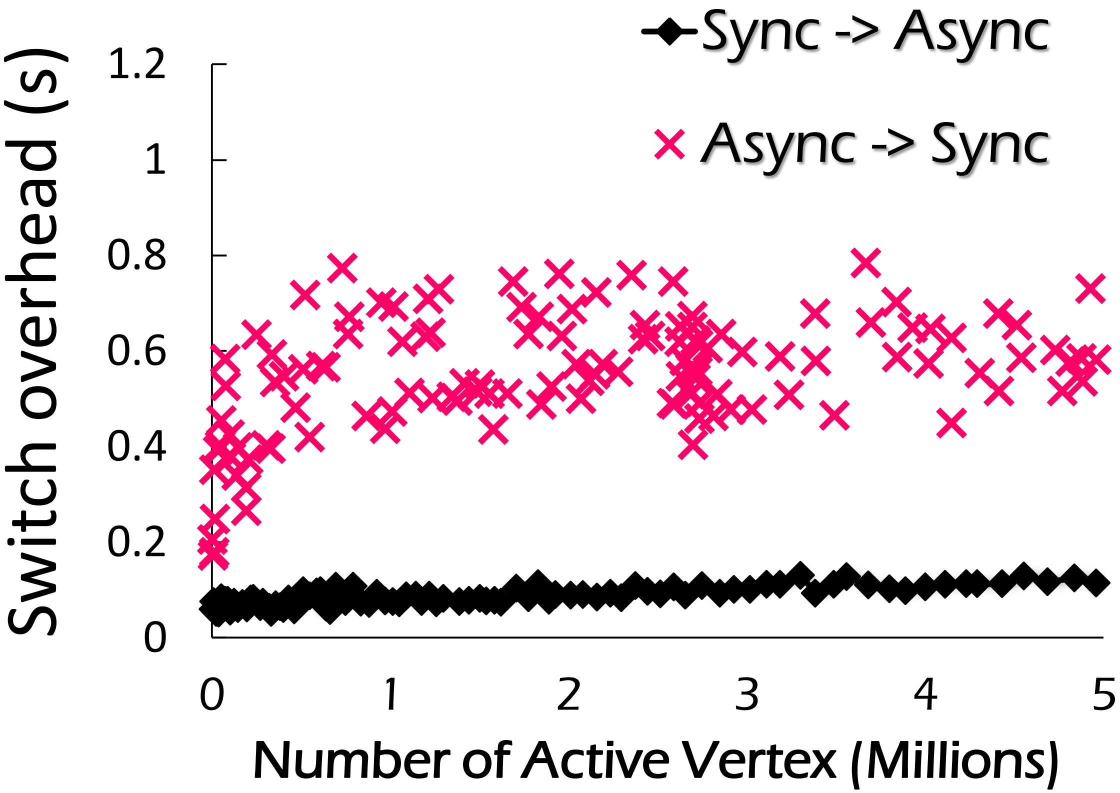


Outperform the baseline with best mode from 9% to 73% for all algorithms and dataset

Switch Overhead

Sync->Async : 0.1s

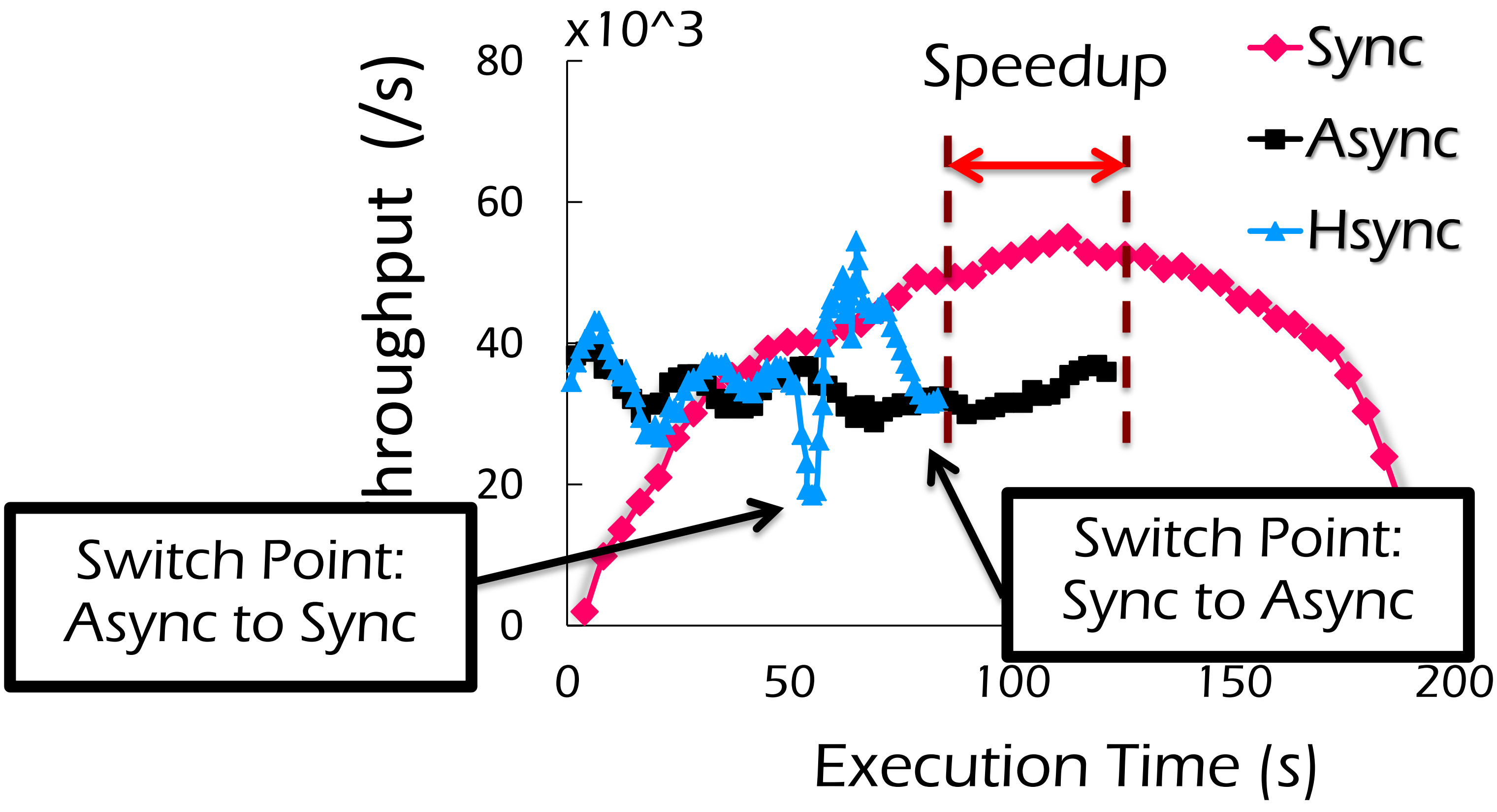
Async->Sync : 0.6s



<u>Overhead grows slightly</u>

with active vertex number increasing.

Case: Single Source Shortest Path (SSSP)



Switch Point:
Async to Sync

Switch Point:
Sync to Async

Execution Mode: Async -> Sync-> Async

Conclusion

PowerSwitch

- □ A comprehensive analysis to the performance of Sync and Async modes for different algorithms, configuration and stages
- □ A Hsync mode that dynamically switch modes between Sync & Async to pursue optimal performance
- □ An effective switch timing model to predict suitable mode with sampling & profiling
- □ Outperforms GraphLab with best mode from 9% to 73% for various algorithms and dataset

Thanks

PowerSwitch

http://ipads.se.sjtu.edu.cn/projects/powerswitch.html

Institute of Parallel and Distributed Systems

Questions