Replication-based Fault-tolerance for Large-scale Graph Processing

Rong Chen

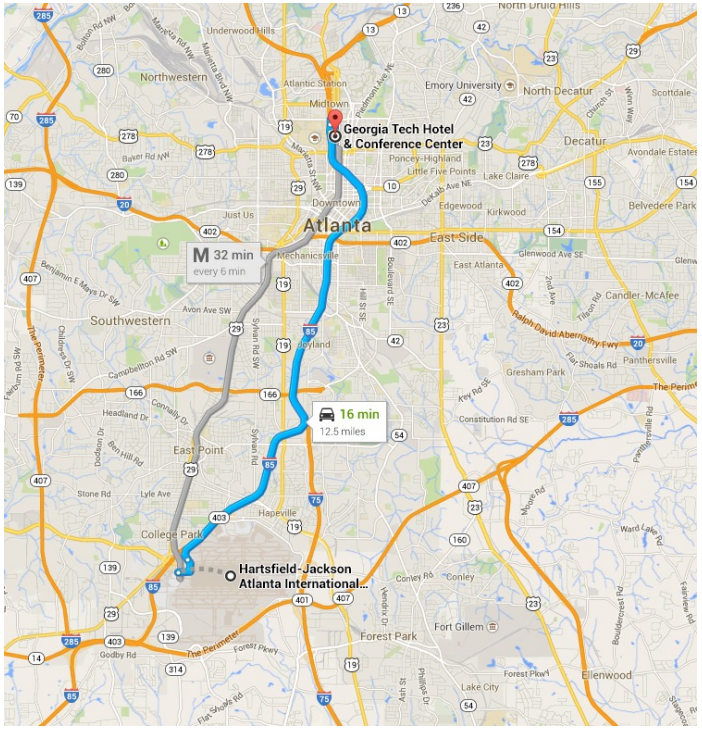
joint work with Peng Wang, Kaiyuan Zhang,
Haibo Chen, Haibing Guan

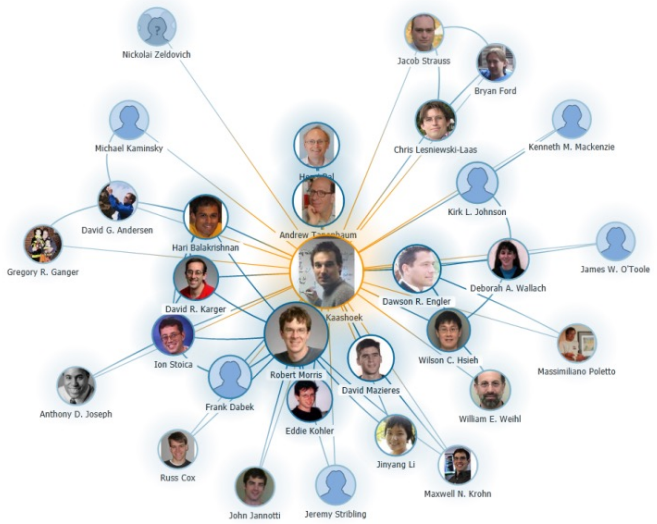
Graph

- Useful information in graph

- Many applications
  - SSSP
  - Community Detection

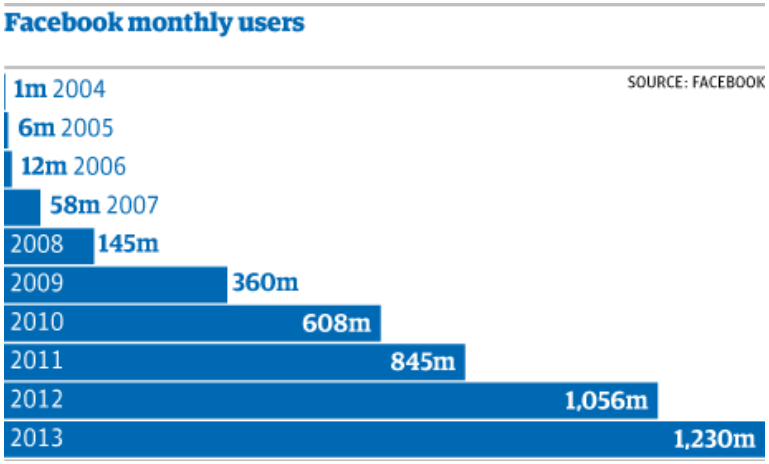
  ……





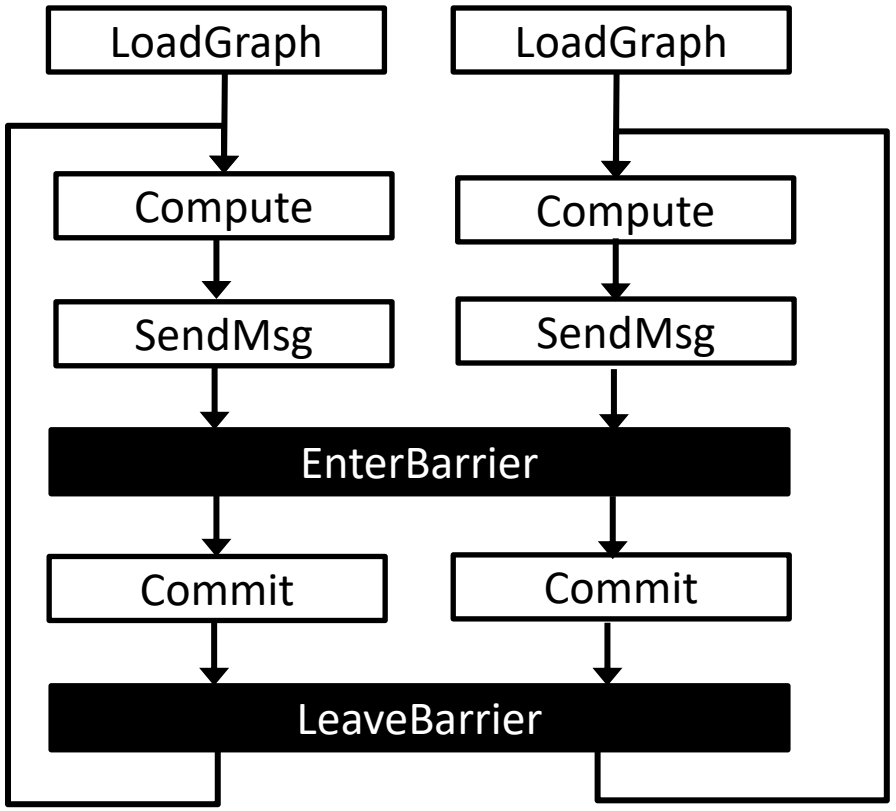
Graph computing

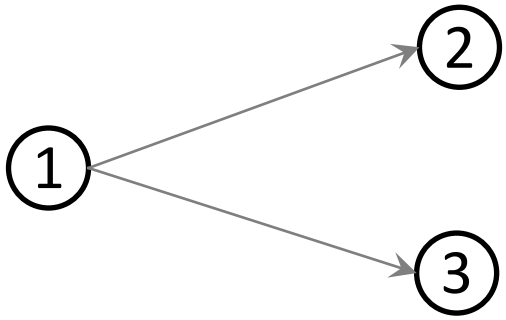
- Graphs are large
  - Require a lot of machines



- Fault tolerance is important

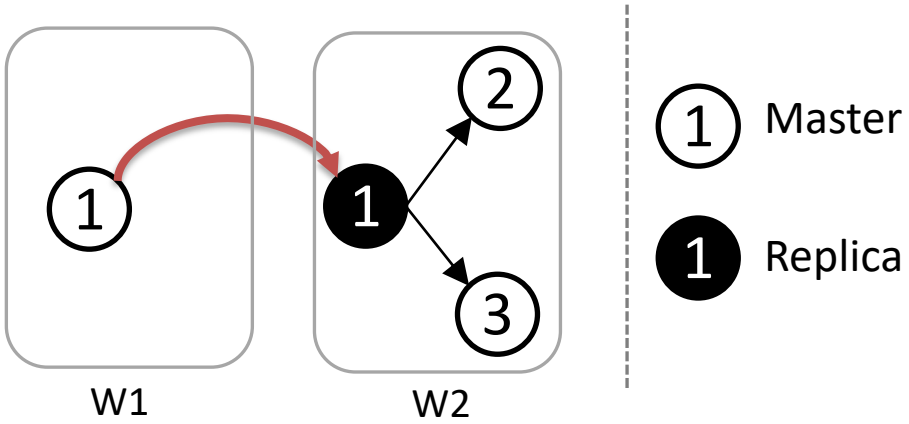
How graph computing works





PageRank: R[i] = 0.15 + $\sum_{j \in Nbs(i)} w_{ji} R[j]$

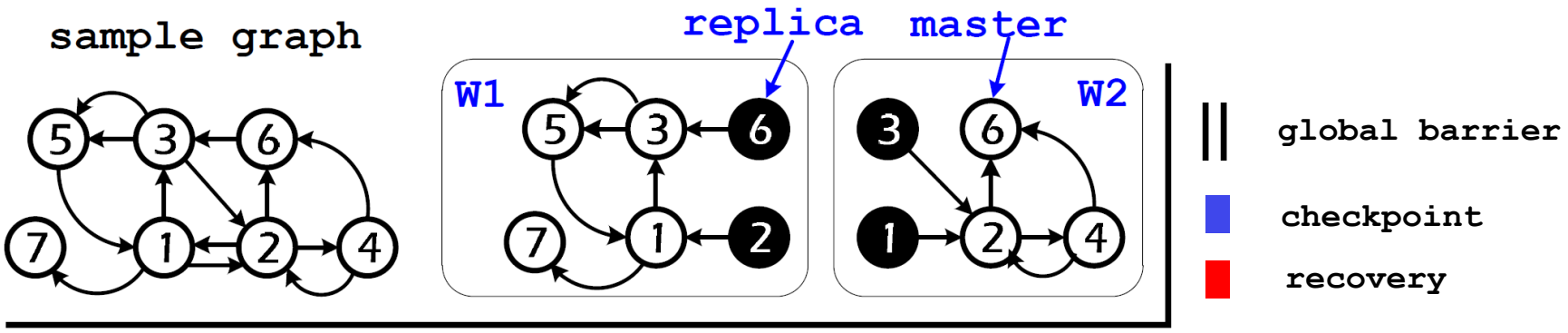
```
PageRank(i)
   // compute its own rank
   total = 0
   foreach  ( j in in_neighbors(i)) :
      total = total + R[j] * W_ji
   R[i] = 0.15 + total

   // trigger neighbors to run again
   if  R[i] not converged then
      activate all neighbors
```

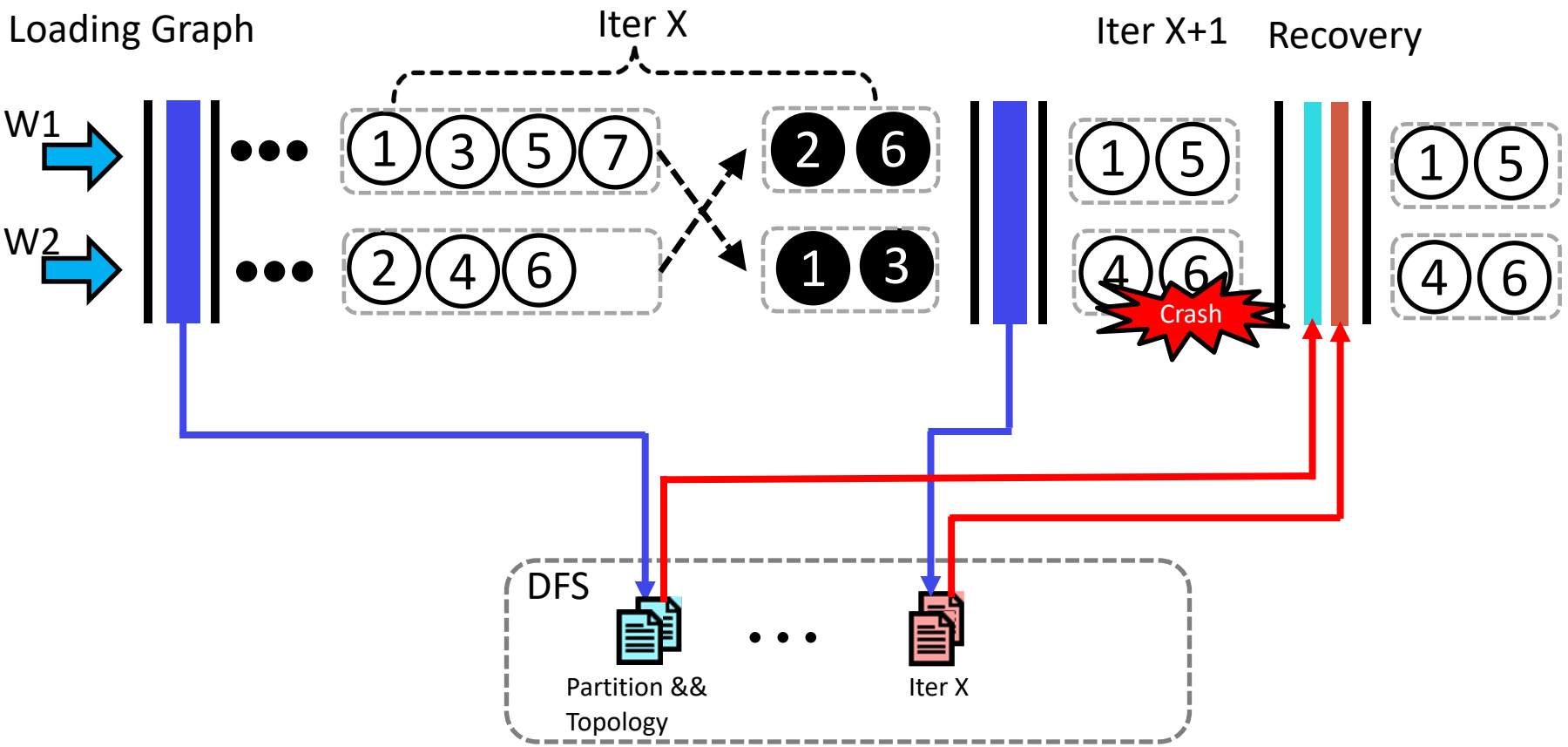


Related work about fault tolerance

- Simple re-execution (MapReduce) 
  - Complex data dependency
- Coarse-grained FT (Spark) 
  - Fine-grained update on each vertex
- State-of-the-art fault tolerance for graph computing
  - Checkpoint
  - Trinity, PowerGraph, Pregel, etc.

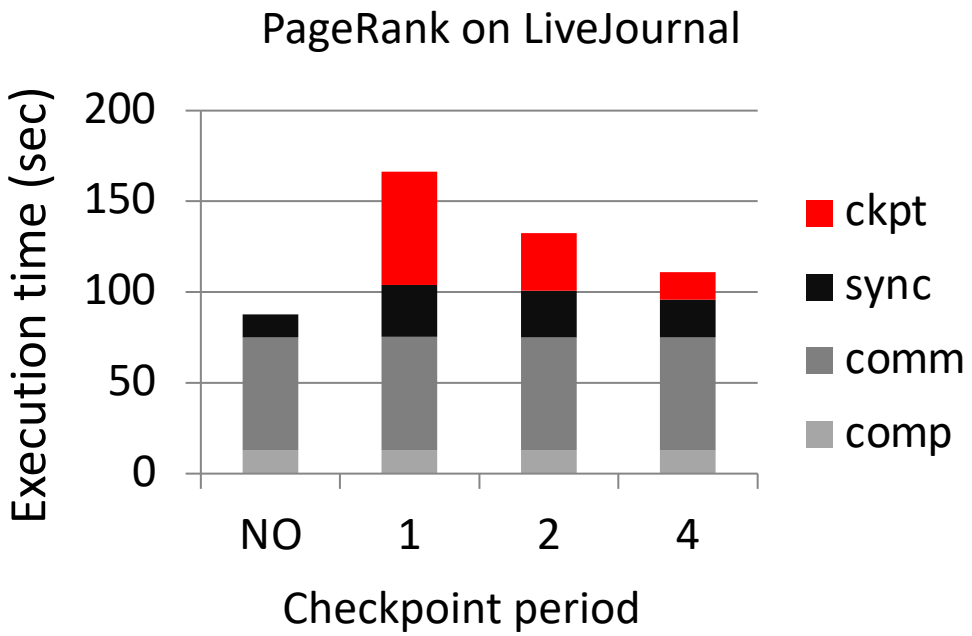
How checkpoint works





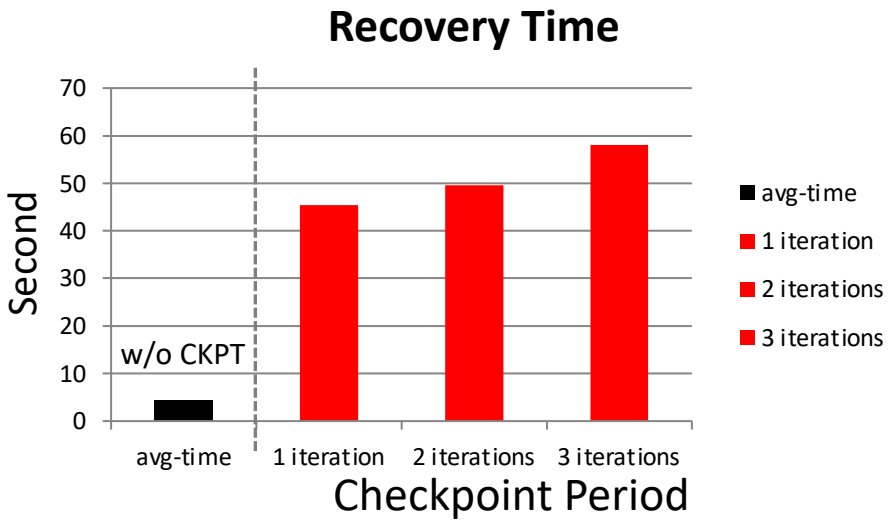
Problems of checkpoint

- Large execution overhead
  - Large amount of states to write
  - Synchronization overhead



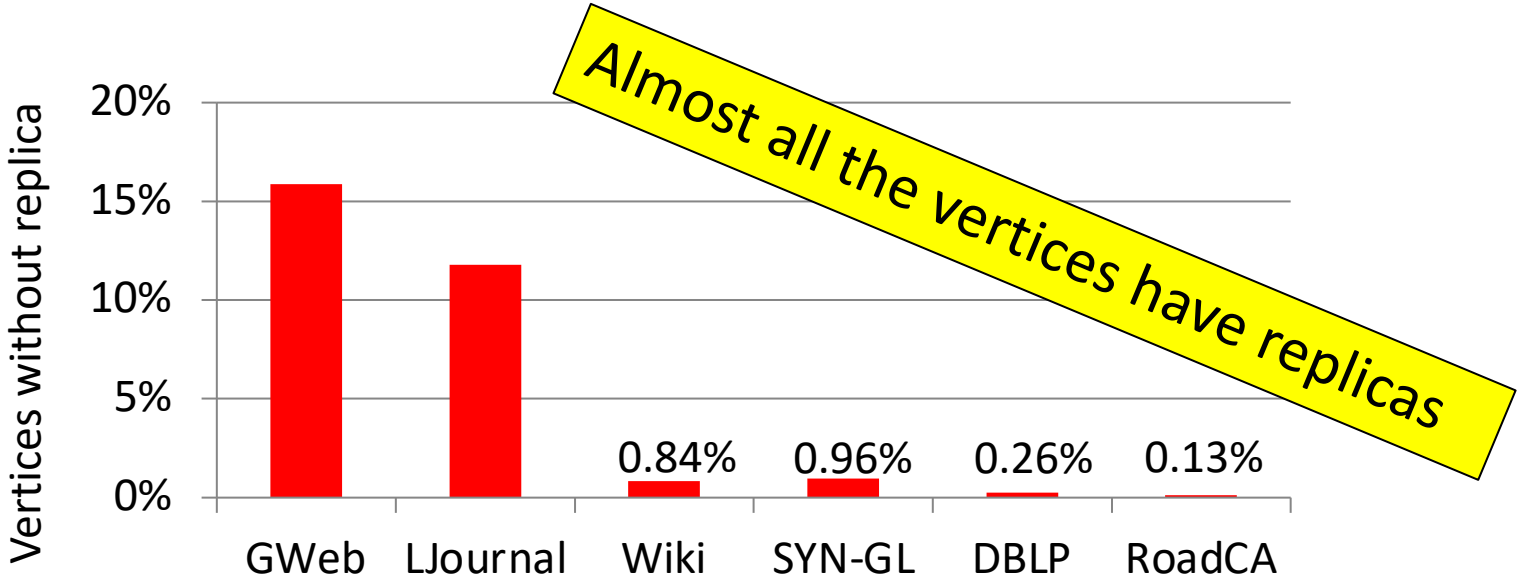
Problems of checkpoint

- Large overhead
- Slow recovery
  - A lot of I/O operations
  - Require standby node



Observation and motivation

- Reuse existing replicas to provide fault tolerance



- Reuse existing replicas → small overhead
- Replicas distributed in different machines → fast recovery

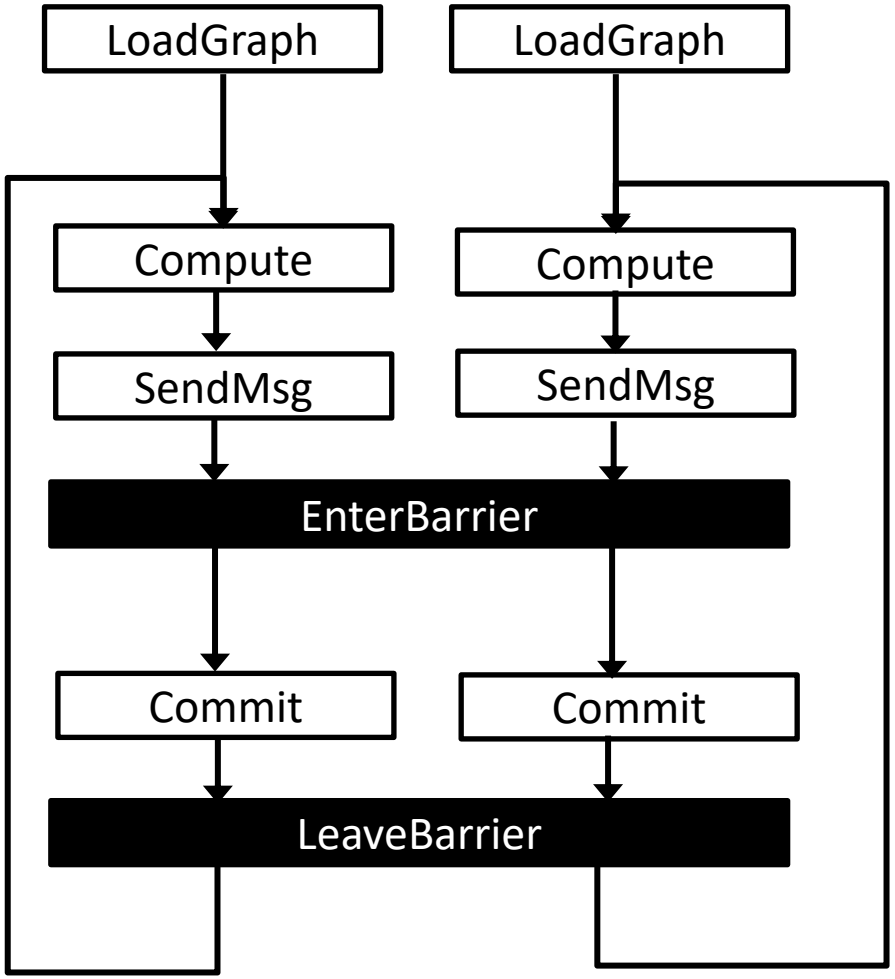
Contribution

- Imitator: a replication based fault tolerance system for graph processing
- Small overhead
  - Less than 5% for all cases
- Fast recovery
  - Up to 17.5 times faster than the checkpoint one

Outline

- Execution flow
- Replication management
- Recovery
- Evaluation
- Conclusion

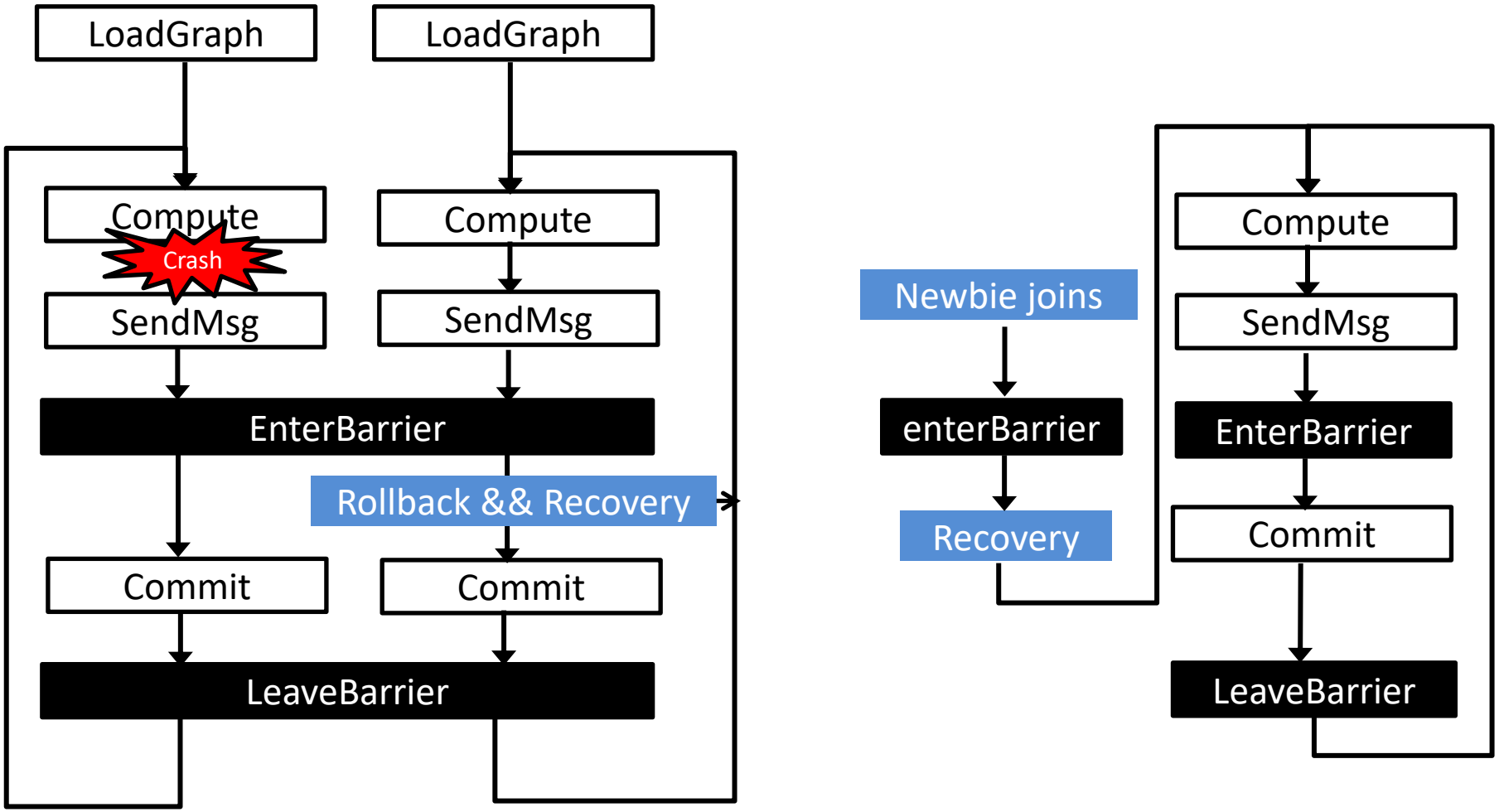
Normal execution flow



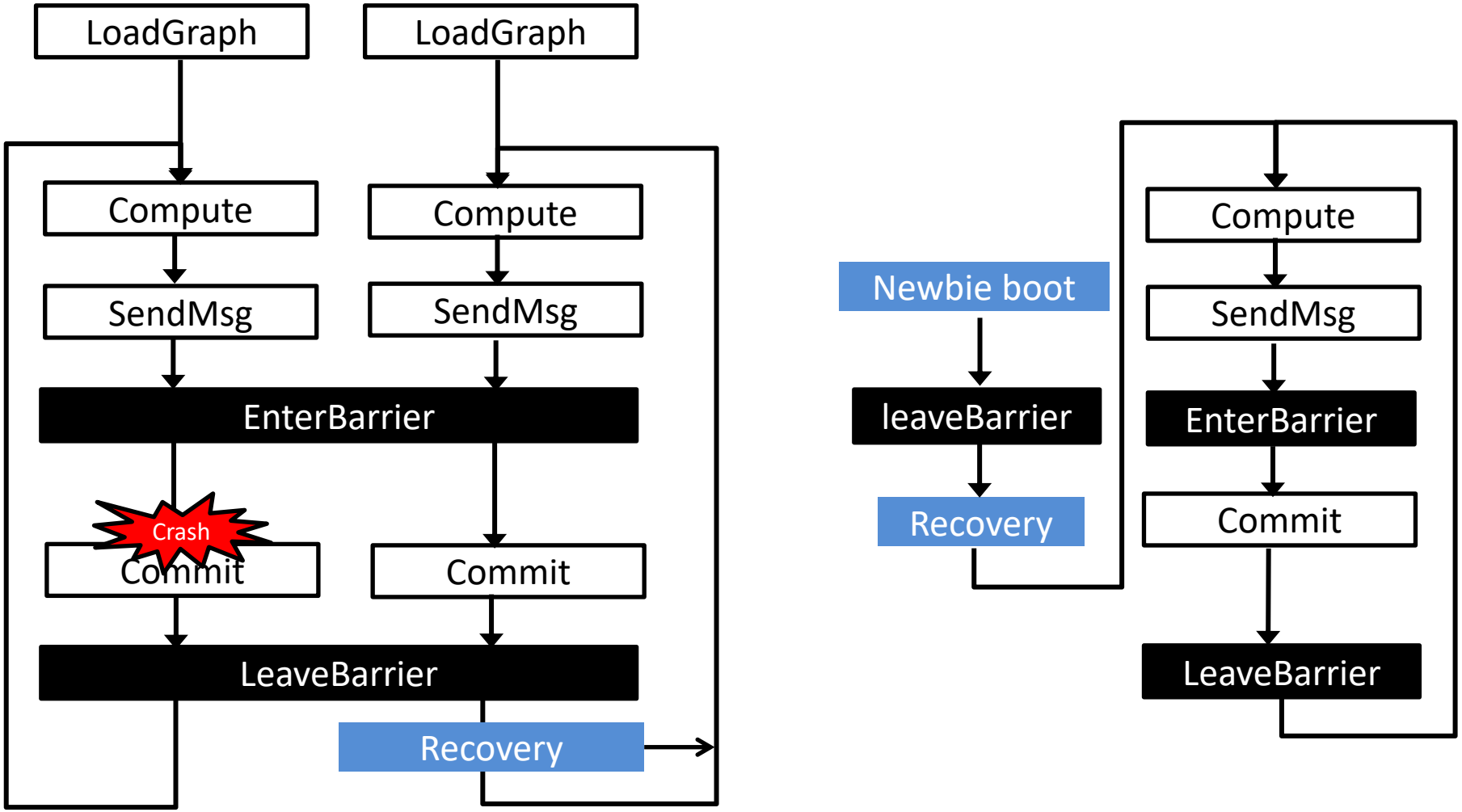
1. Adding FT support

2. Extending normal synchronization message

Failure before barrier

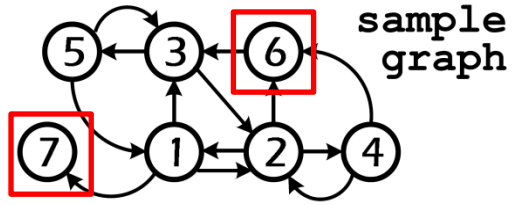


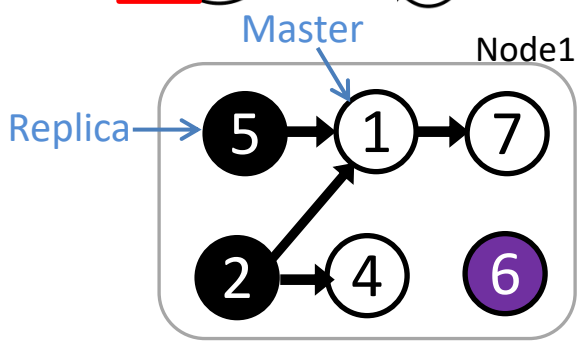
Failure during barrier

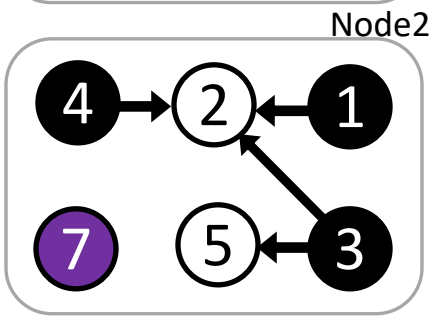


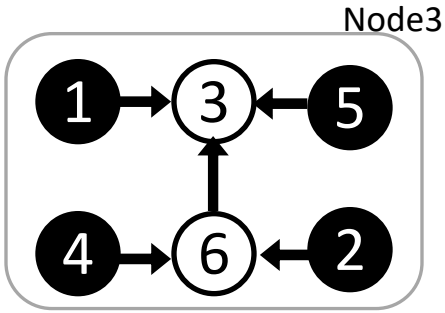
Management of replication

- Fault tolerance replicas
  - every vertex has at least f replicas to tolerate f failures
- Full state replica (mirror)
  - Existing replica lacks meta information
  - Such as replication location





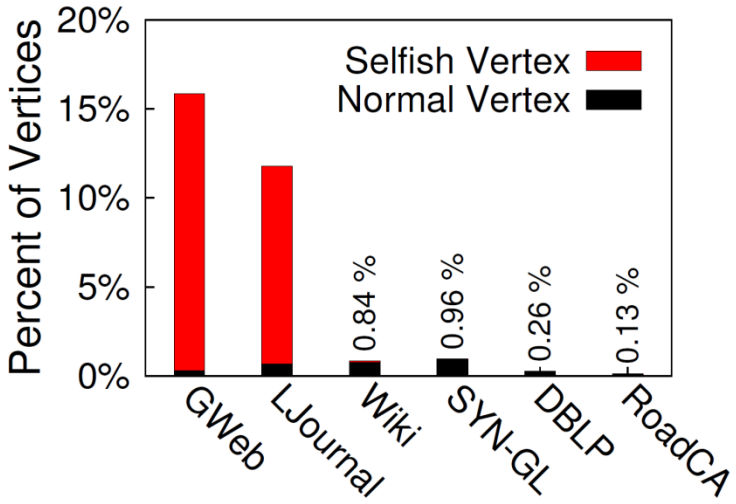


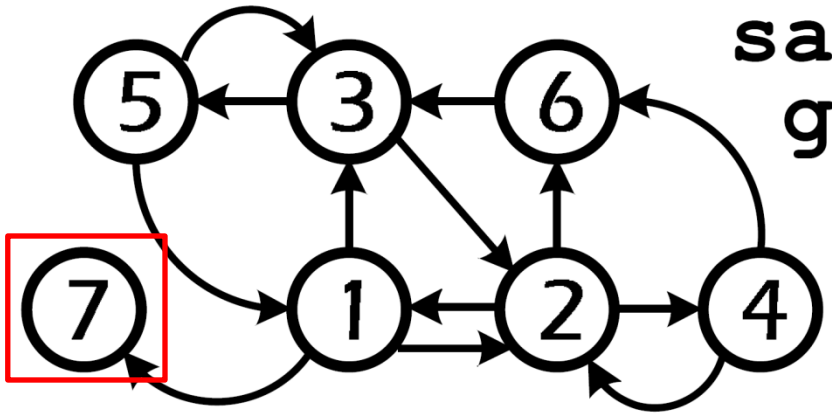


Vertex: 5
Master: n2
Replicas: n1 | n3

Optimization: selfish vertices

- States of selfish vertices have no consumer
- Their states may only decided by their neighbors
- Opt: get their states by re-computation





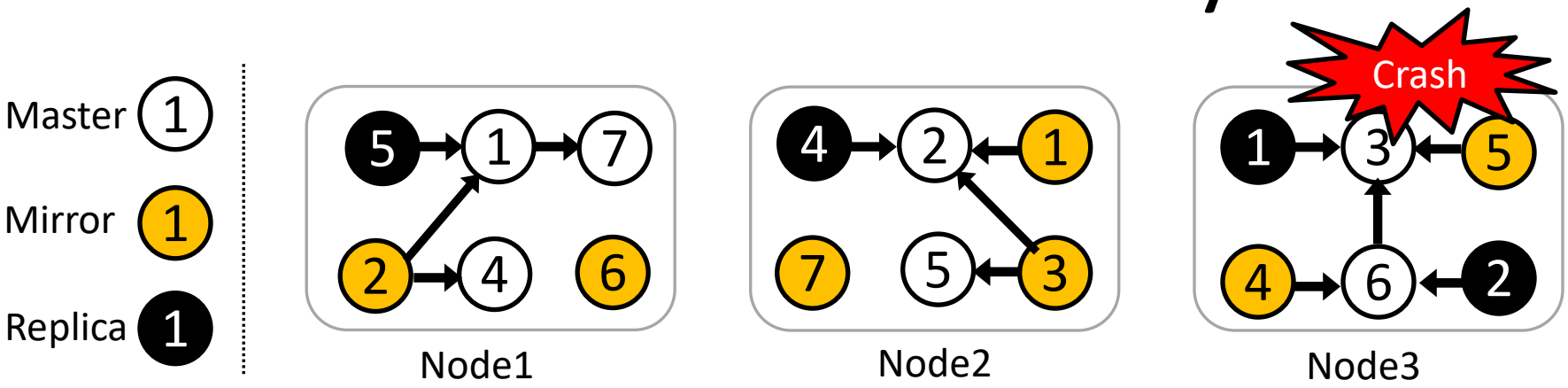
sample graph

PageRank: R[i] = 0.15 + $\sum_{j \in Nbs(i)} w_{ji} R[j]$

How to recover

- Challenges
  - Parallel recovery
  - Consistent state after recovery

Problems of recovery



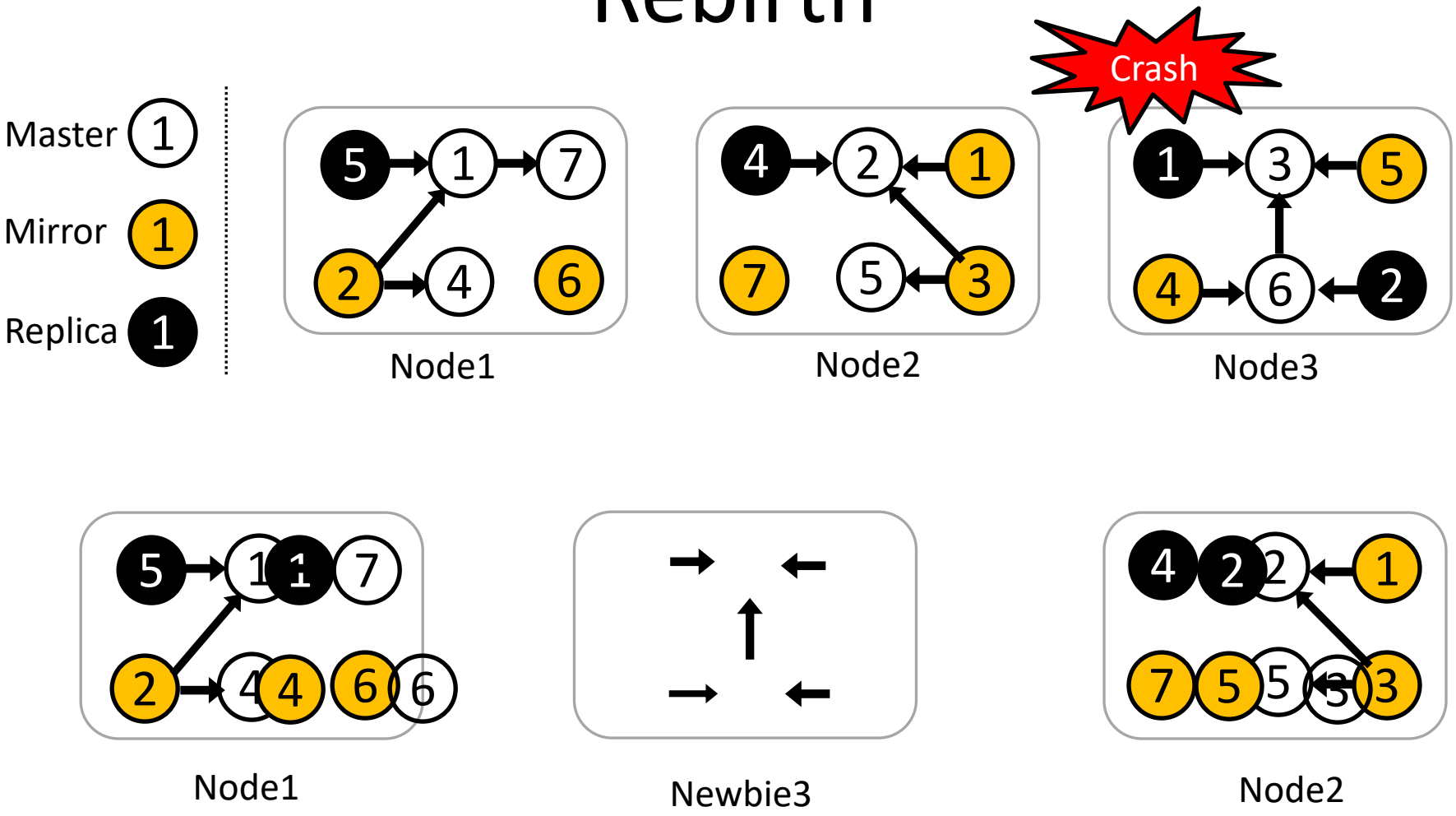
- Which vertices have crashed?
- How to recover without a central coordinator?

Rules:
1. Master recovers replicas
2. If master crashed, mirror recovers master and replicas

Replication Location

Vertex 3:
Master: n3
Mirror: n2

Rebirth



Rule:

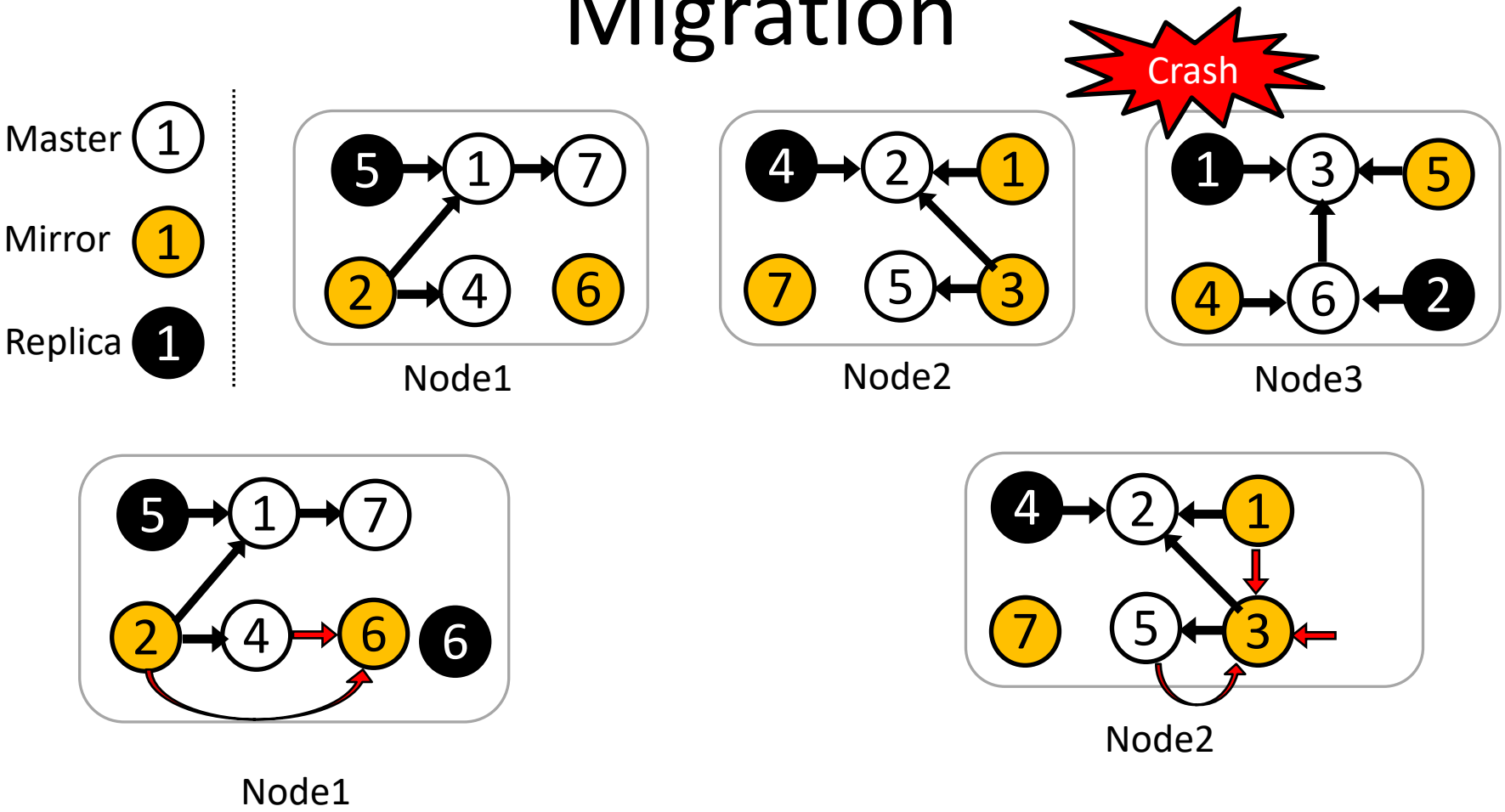
1. Master recovers replicas
2. If master crashed, mirror recovers master and replicas

Problems of Rebirth

- Standby machine
- A single newbie machine

Migrate tasks to surviving machines

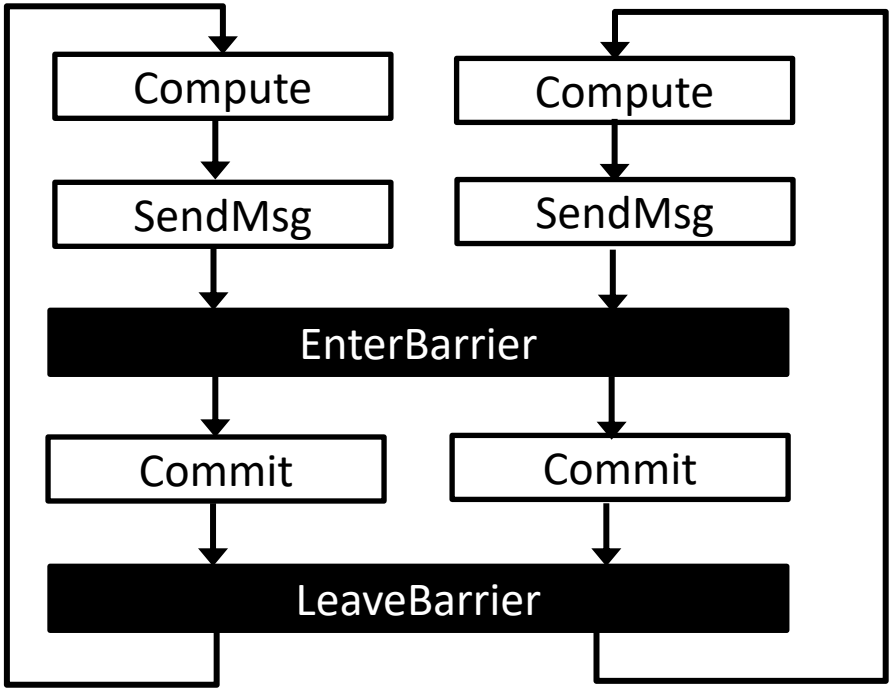
Migration

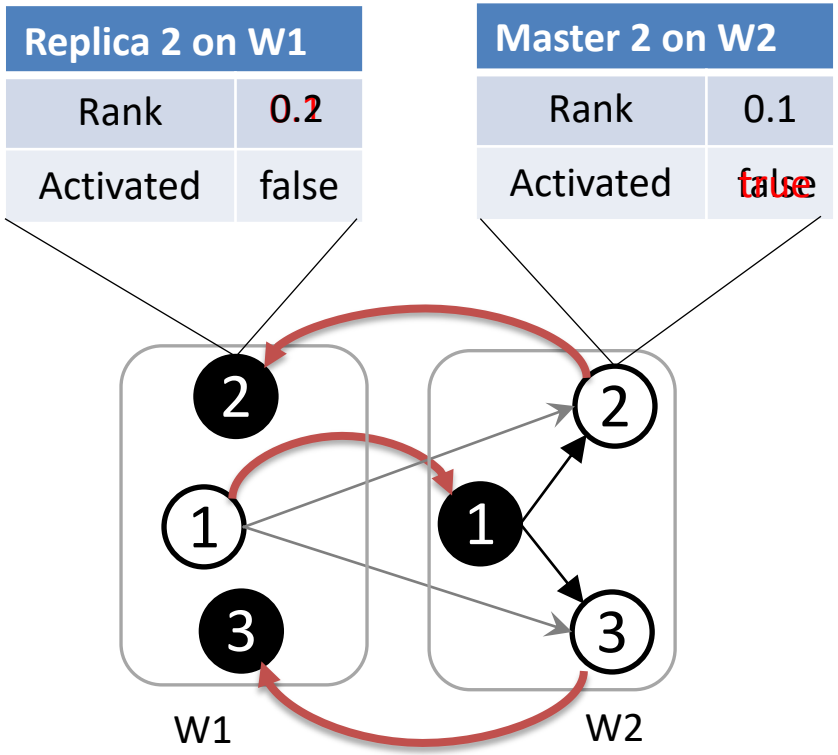


Procedure:

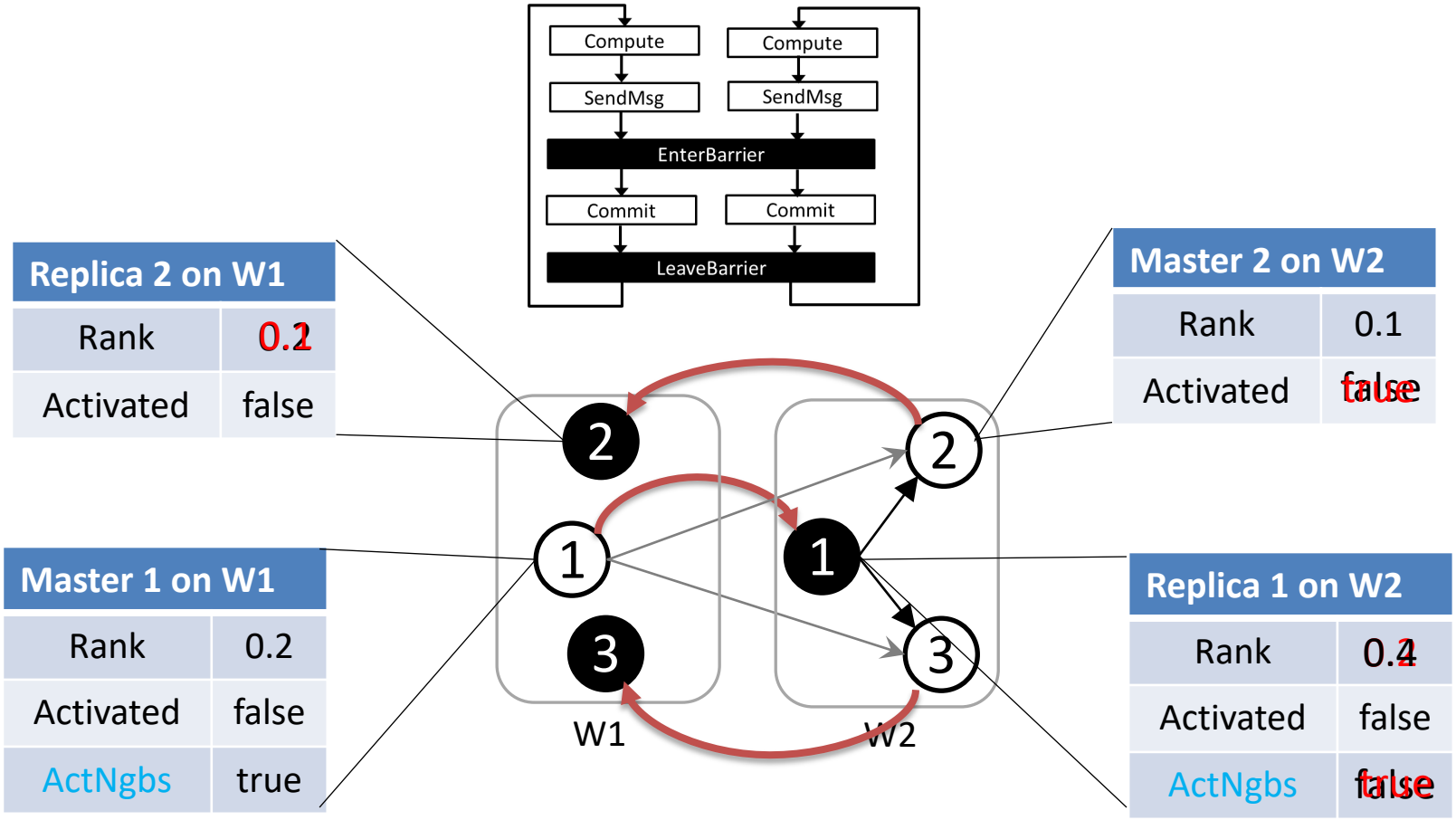
1. Mirrors upgrade to masters and broadcast
2. Reload missing graph structure and reconstruct

Inconsistency after recovery





Replay Activation

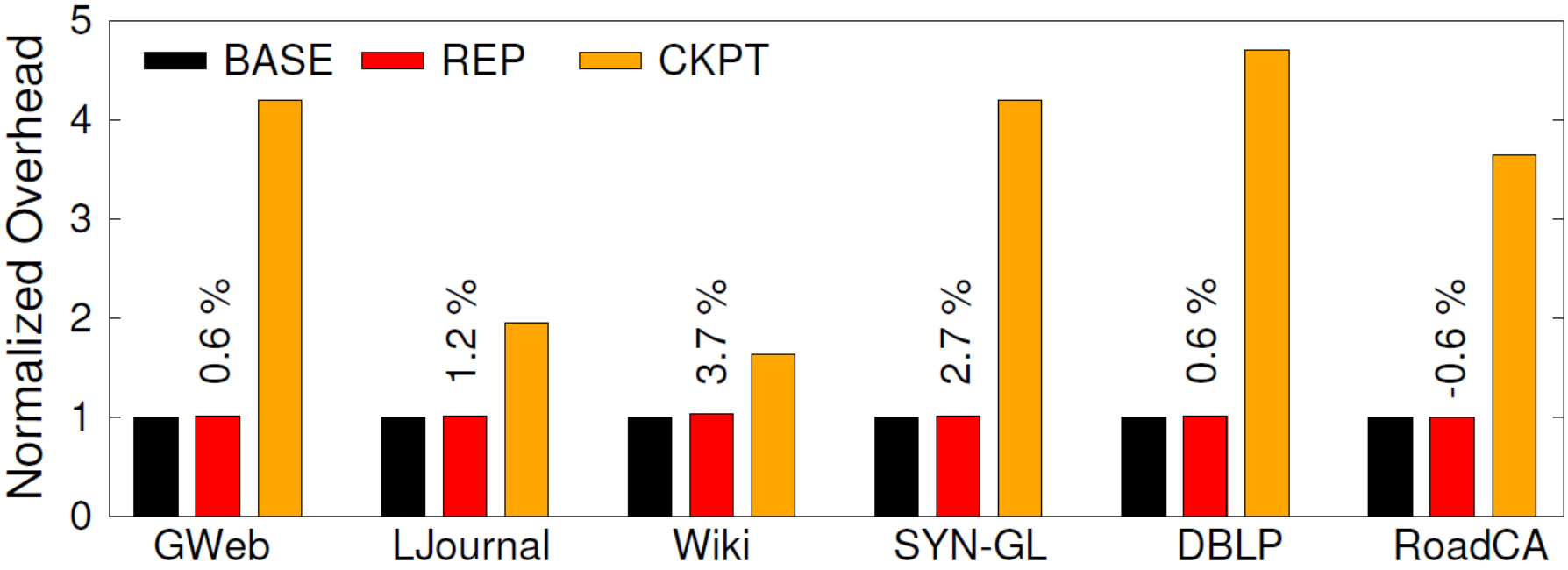


Evaluation

- 50 VMs (10G memory 4cores)
- HDFS (3 Replications)
- Applications

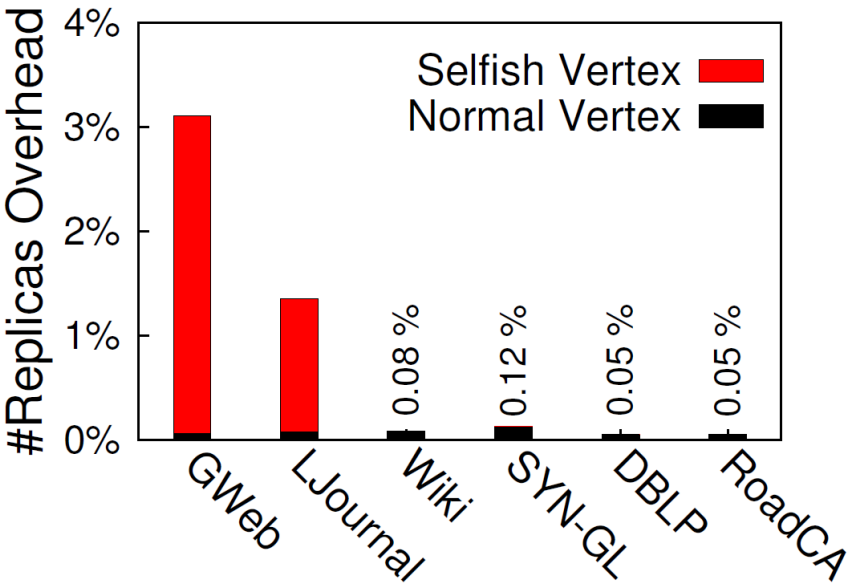
Application	Graph	Vertices	Edge
PageRank	GWeb	0.87M	5.11M
	LJournal	4.85M	70.0M
	Wiki	5.72M	130.1M
ALS	SYN-GL	0.11M	2.7M
CD	DBLP	0.32M	1.05M
SSSP	RoadCA	1.97M	5.53M

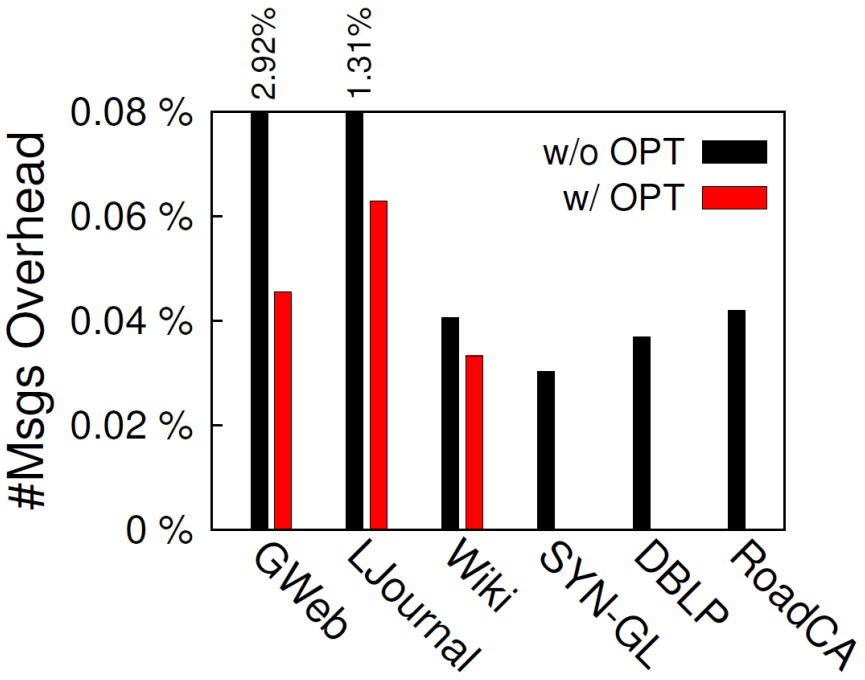
Normal execution overhead



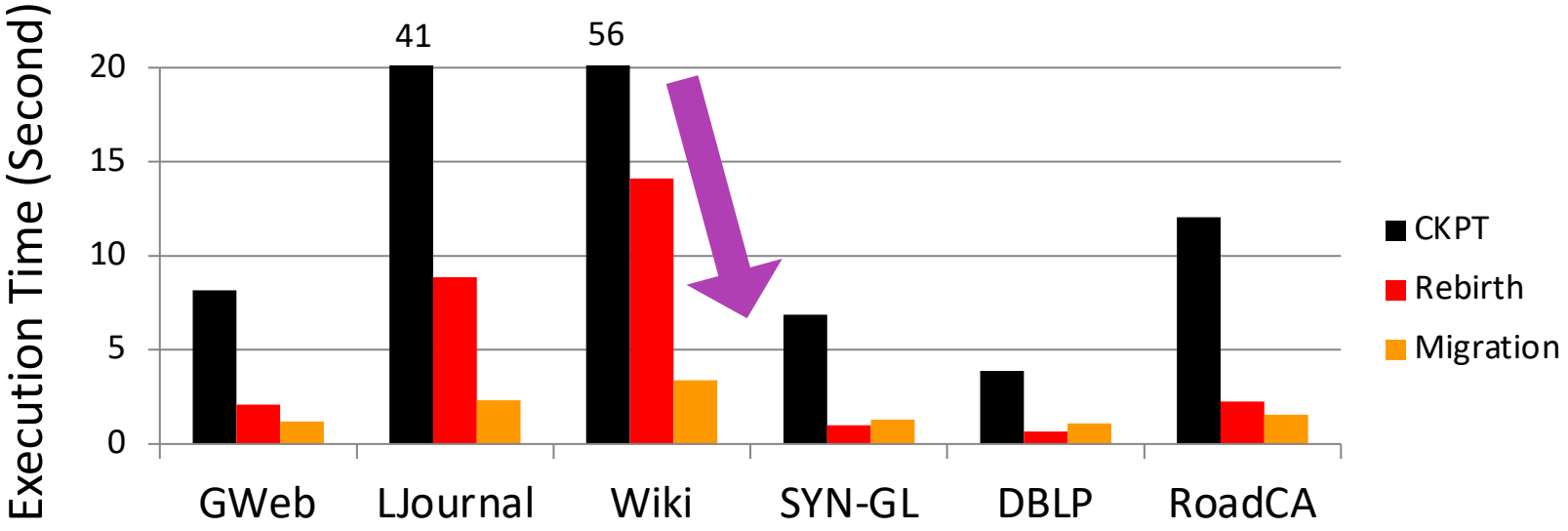
Replication has negligible execution overhead

Communication Overhead

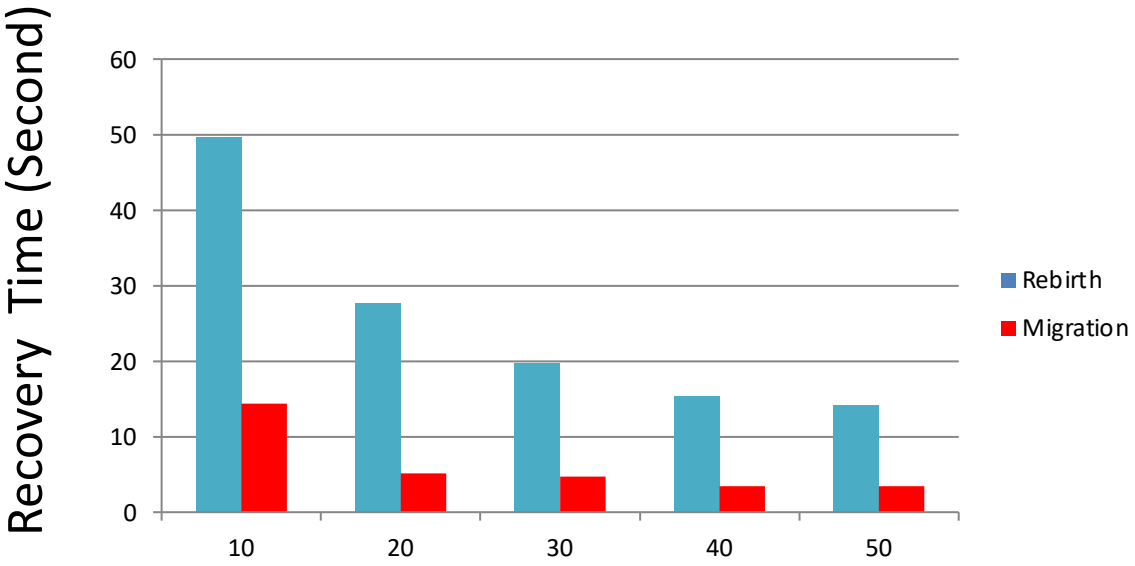




Performance of recovery

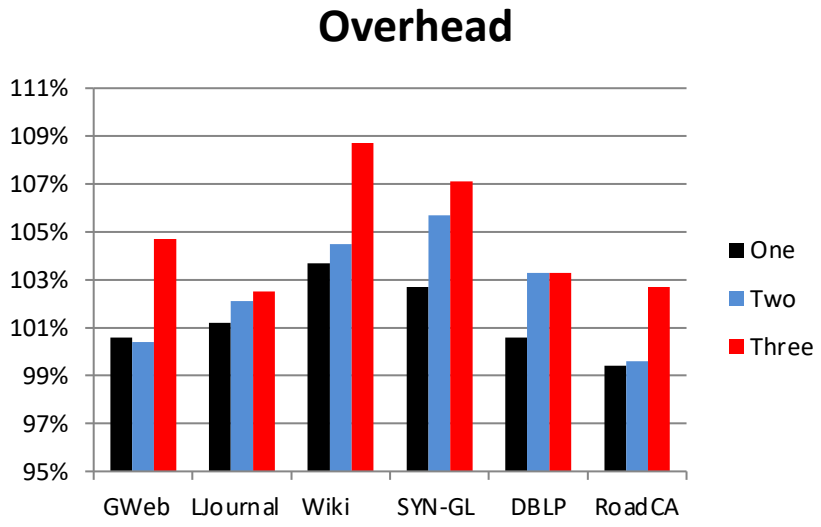


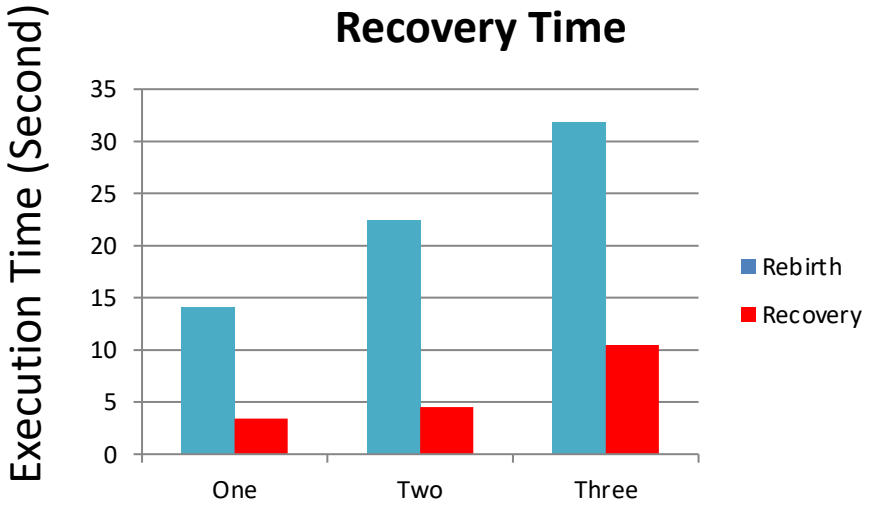
Recovery Scalability



The more machines, the faster the recovery

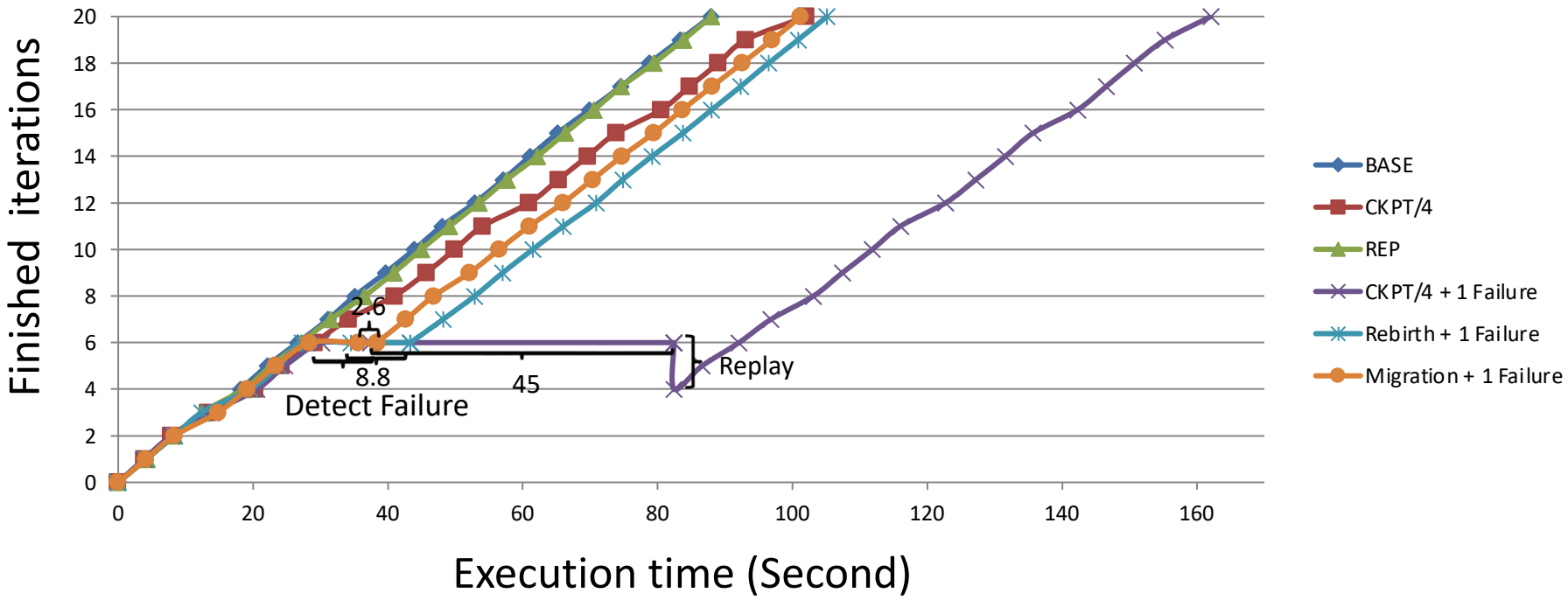
Simultaneous failure





Add more replicas to tolerate more than 1 machines simultaneous failure

Case study



- Application: PageRank on the dataset of LiveJournal
- A checkpoint for every 4 iterations
- A Failure is injected between the 6th iteration and the 7th iteration

Conclusion

- Imitator: a graph engine which supports fault tolerance
- Imitator’s execution overhead is negligible because it leverages existing replicas
- Imitator’s recovery is fast because of its parallel recovery approach

Next Lecture

Topic: Graph Neural Network Training

THANKS