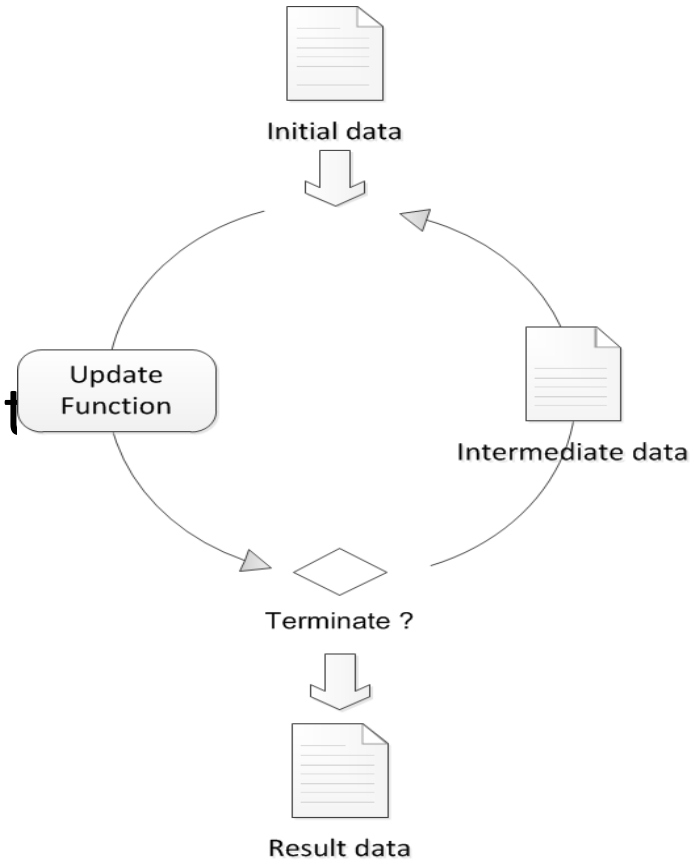
# i²MapReduce: Incremental Iterative MapReduce

Yanfeng Zhang
Computing Center
Northeastern University, China

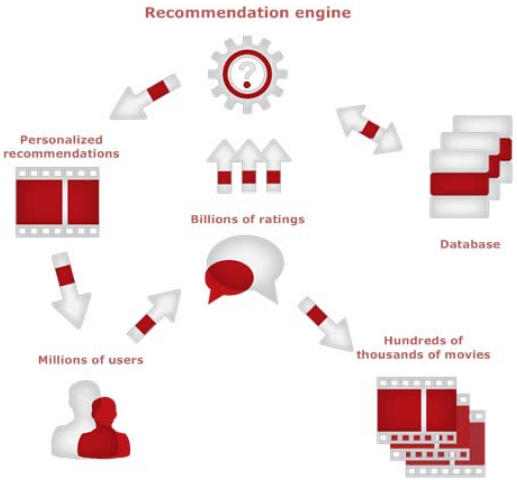
Shimin Chen
Institute of Computing Technology
Chinese Academy of Sciences

# Iterative Computation

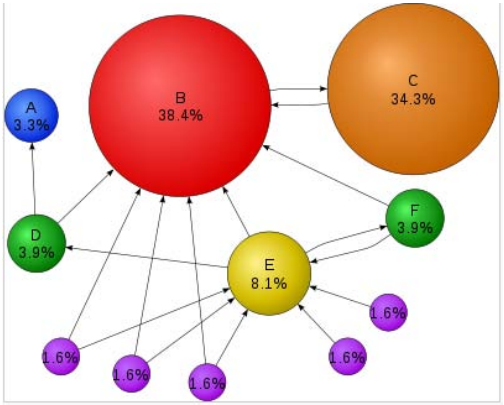
- Use the same computation logic (update function) to process the data many times
- The previous iteration's output is the next iteration's input
- Stop when the iterated result converges to a fixed point



# Iterative Cloud Intelligence Apps

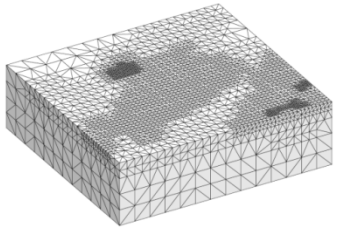



Collaborative filtering recommendation

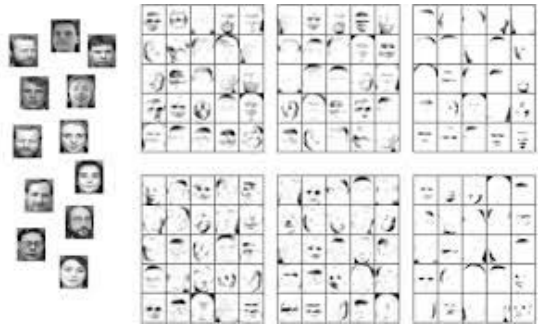



PageRank

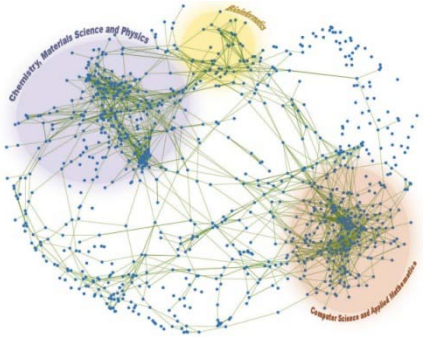

Shortest path



Earthquake/hurricane prediction



Non negative matrix factorization

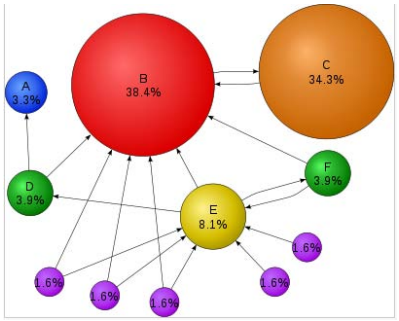



Data clustering

# Iterative Computation

$$v^k = F(v^{k-1}, D)$$

- v: state data (updated every iteration)
- D: structure data (static during iterative computation)
- F(): iterative update function

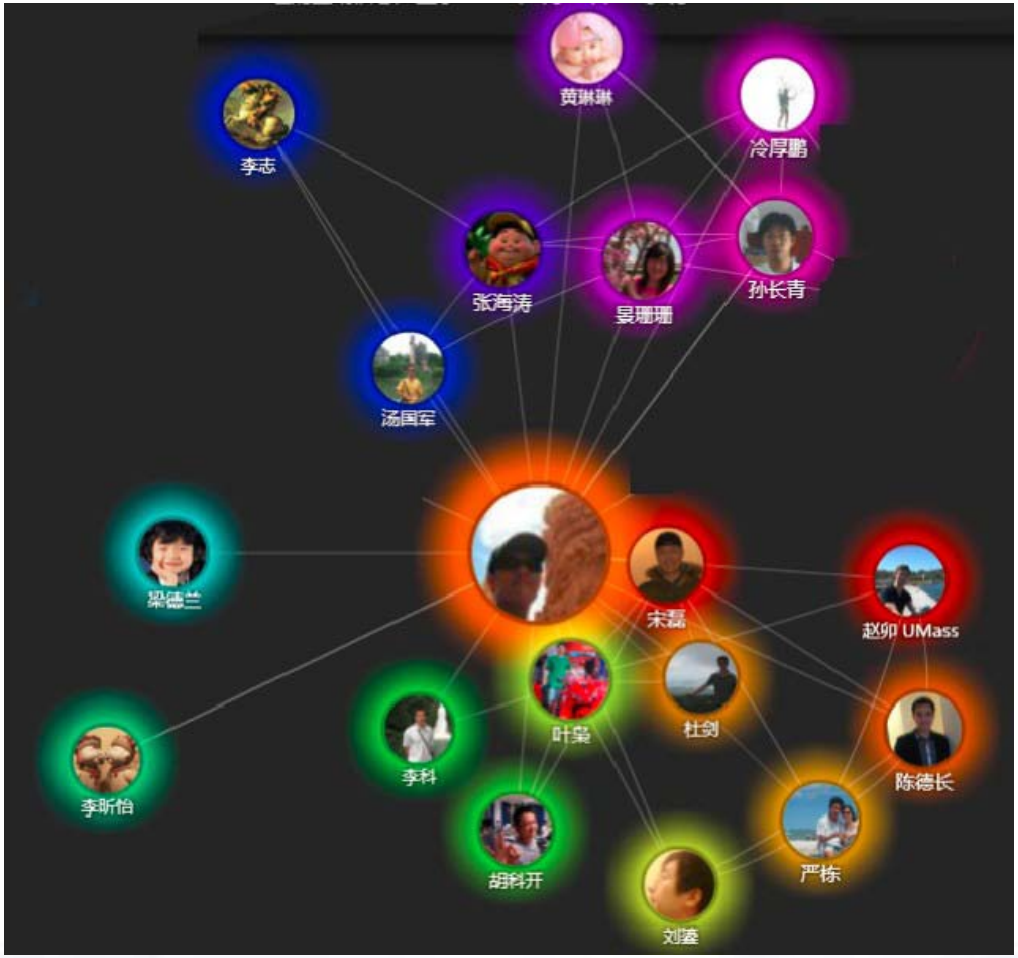



$$R^{(k)} = dWR^{(k-1)} + (1-d)E$$

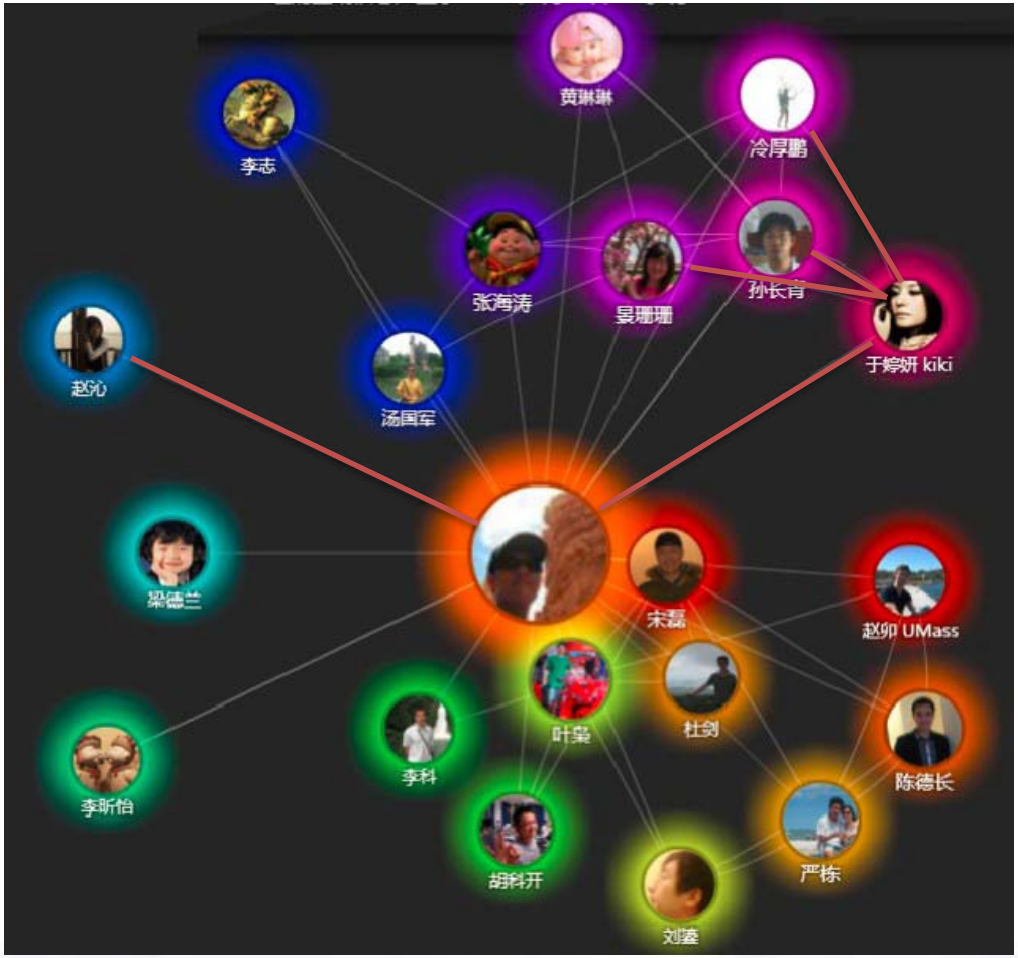
v: PageRank scores R

D: web graph matrix W

# Structure Data is Changing

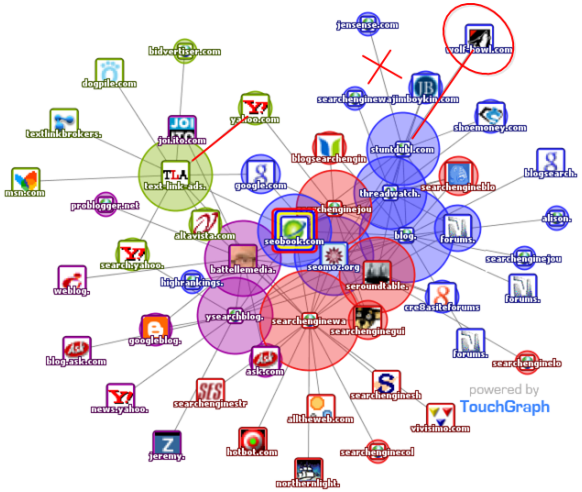


# Structure Data is Changing



# Structure Data is Changing


Changing social graph




Changing web graph

- Need to update the result to timely reflect the changing dataset
- Start from scratch? – heavy weighted
- Incremental processing

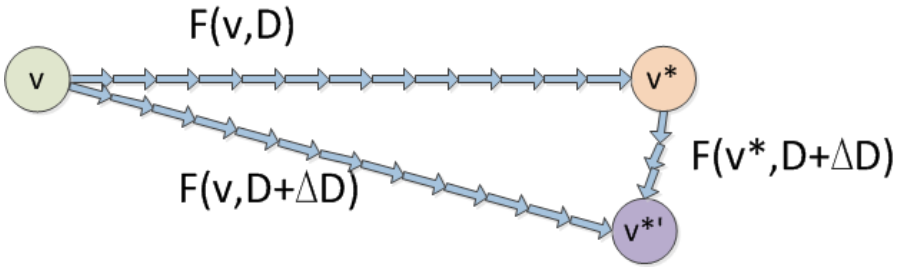
# Incremental Processing

$$v^k = F(v^{k-1}, D)$$

$$v^k = F(v^{k-1}, D + \Delta D)$$

Utilize the previous iterative computation's result:

1. Reduce the number of iterations
- Structure data is slightly changed <-> the result is slightly changed
- Start from the previously converged state rather than from a random start point




2. Reduce the workload of each iteration

$$O(|D + \Delta D|) \Rightarrow O(|\Delta D|)$$

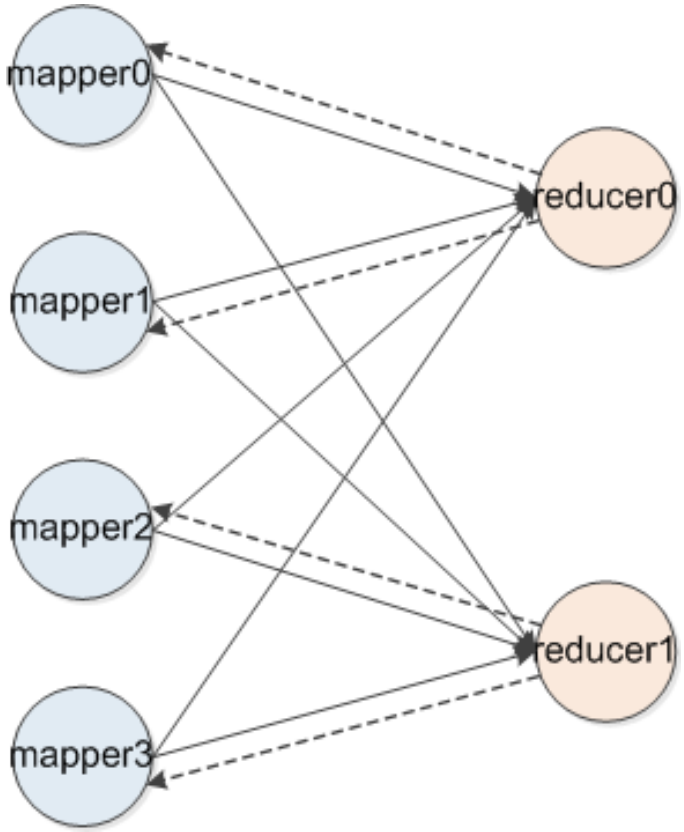
Restart computation → Incremental processing

# Related Works & Our Focus

- Incoop [SOCC 2011] (MPI-SWS)
- Naiad [CIDR 2013] (Microsoft)

- Our Focus: Incremental Iterative MapReduce
  - MapReduce is the most widely used big data processing tool
  - Compatible with existing MapReduce apps

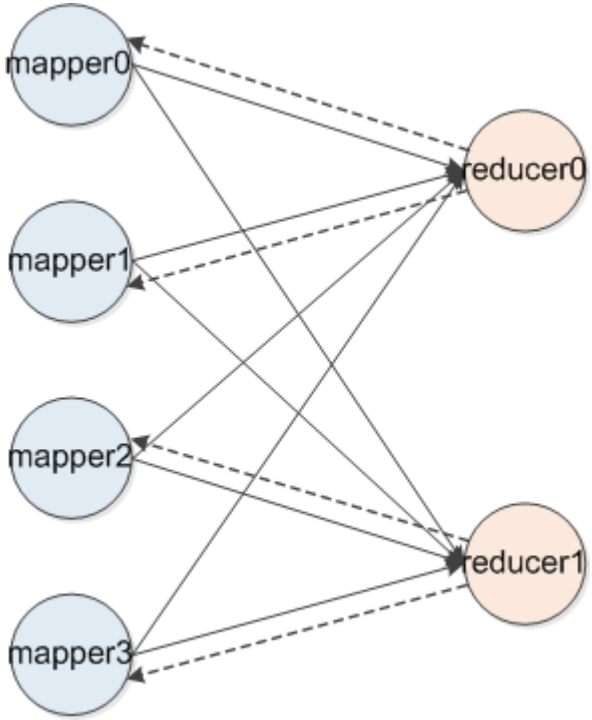
# Map-Reduce Bipartite Graph



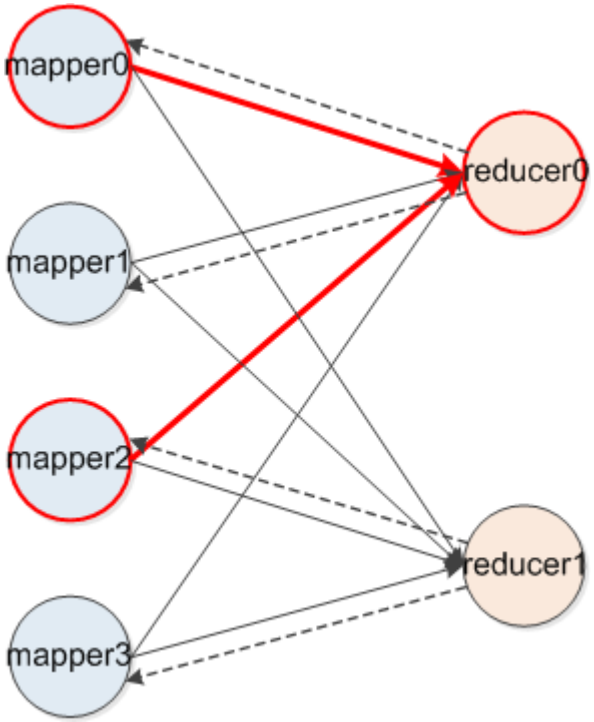

Iterative processing

# Throw a Pebble into Still Water

# Map-Reduce Bipartite Graph

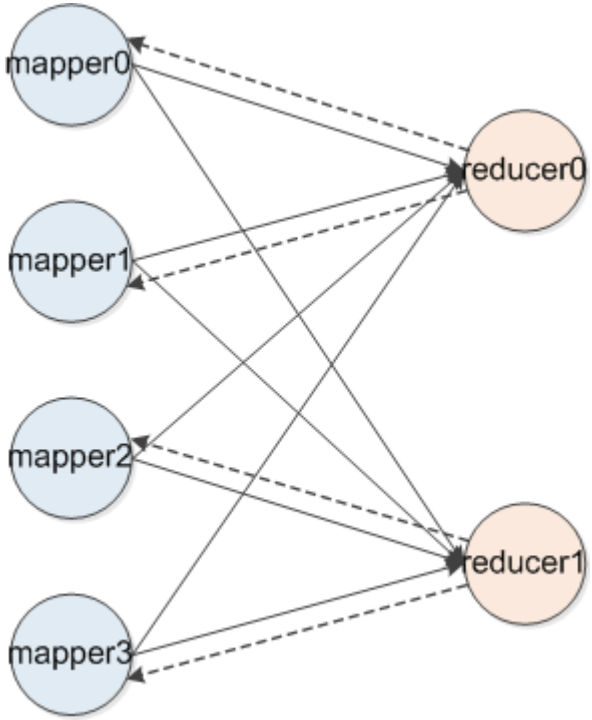



Iterative processing

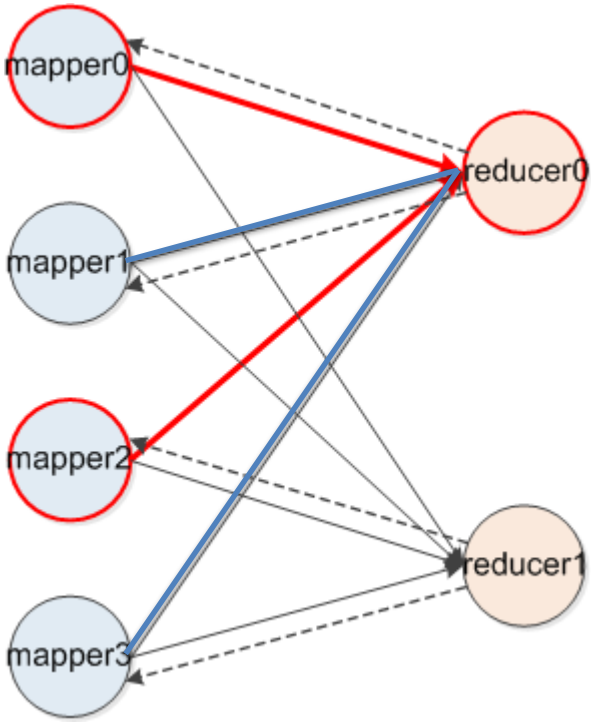



Incremental processing

# Map-Reduce Bipartite Graph

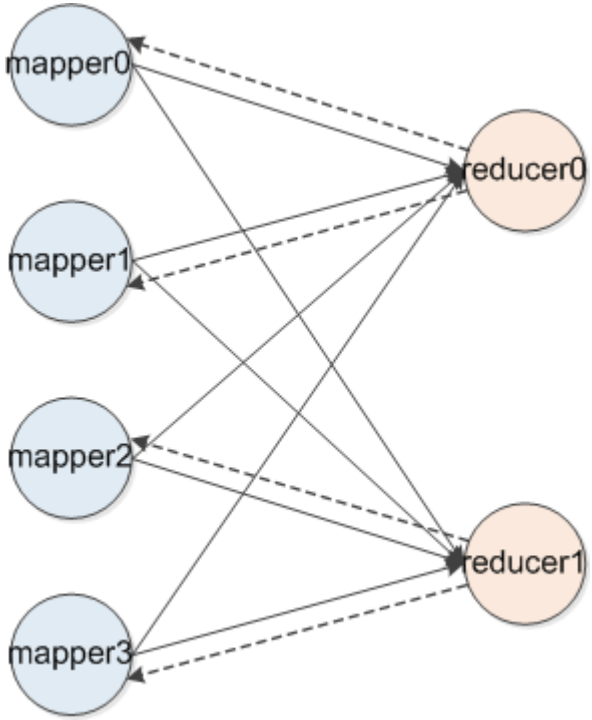



Iterative processing

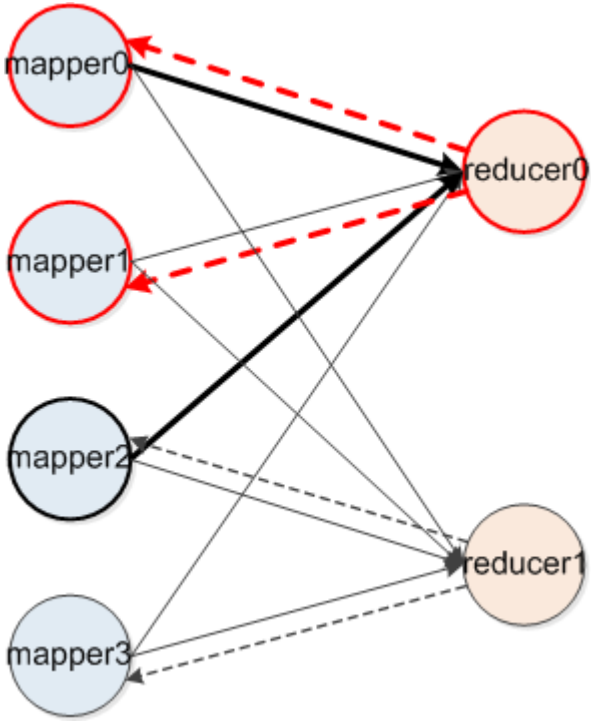



Incremental processing

# Map-Reduce Bipartite Graph



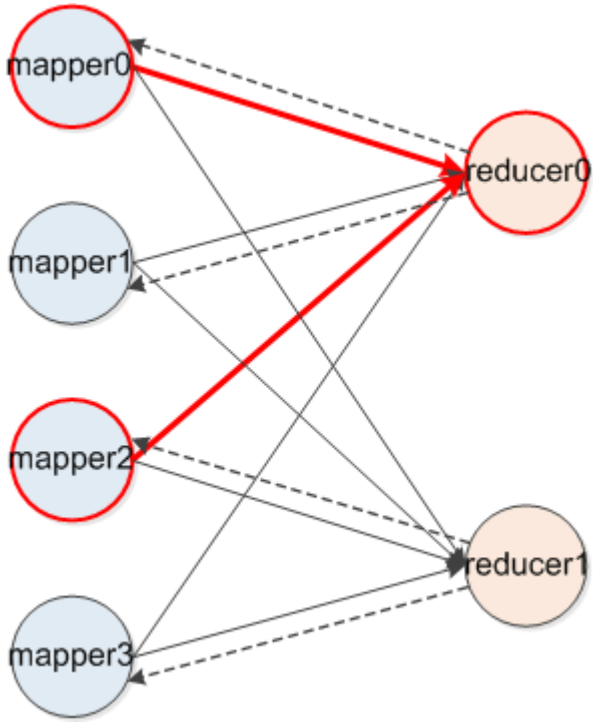

Iterative processing




Incremental processing

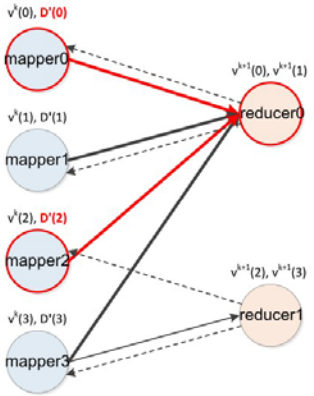
# i²MapReduce: Incremental Processing

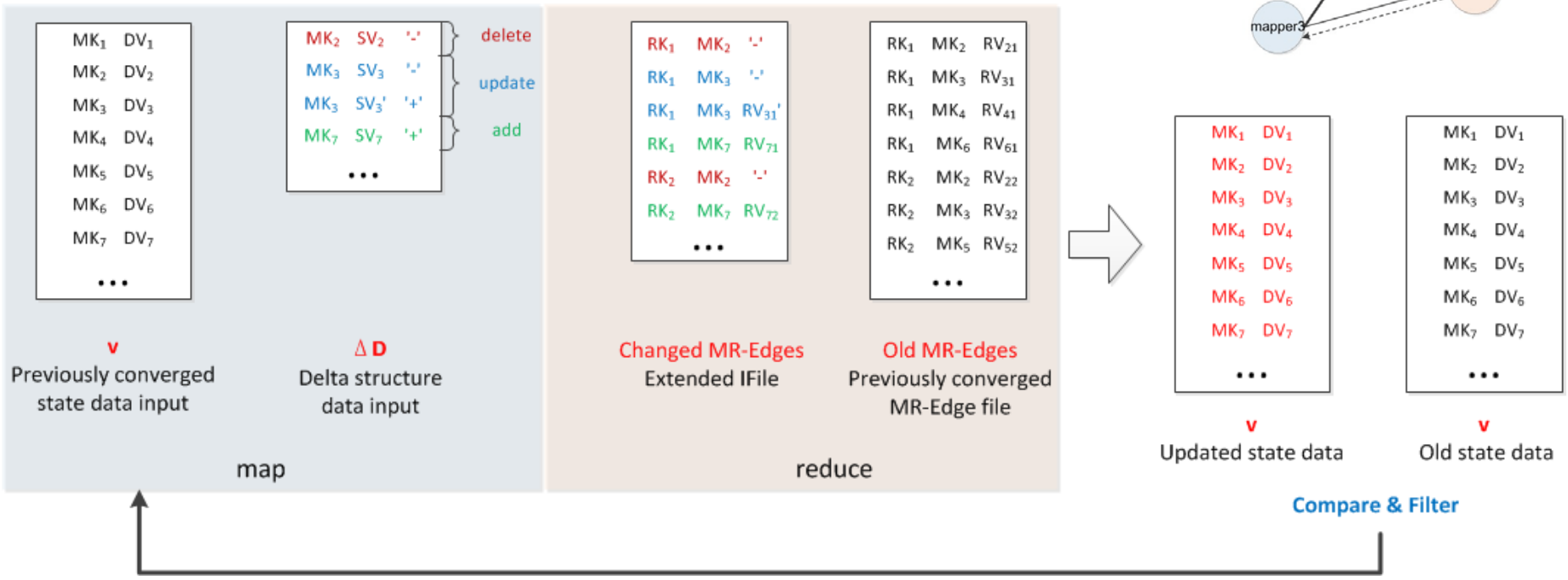
1. Start from the previously converged state data
   - Reduce the number of iterations
2. Only execute the changed mappers/reducers and utilize the converged MR-Edge/RM-Edge state
   - Reduce the workload of each iteration
3. Filter the converged reducers
   - Avoid changes propagation



# i²MapReduce: Prototype Implementation

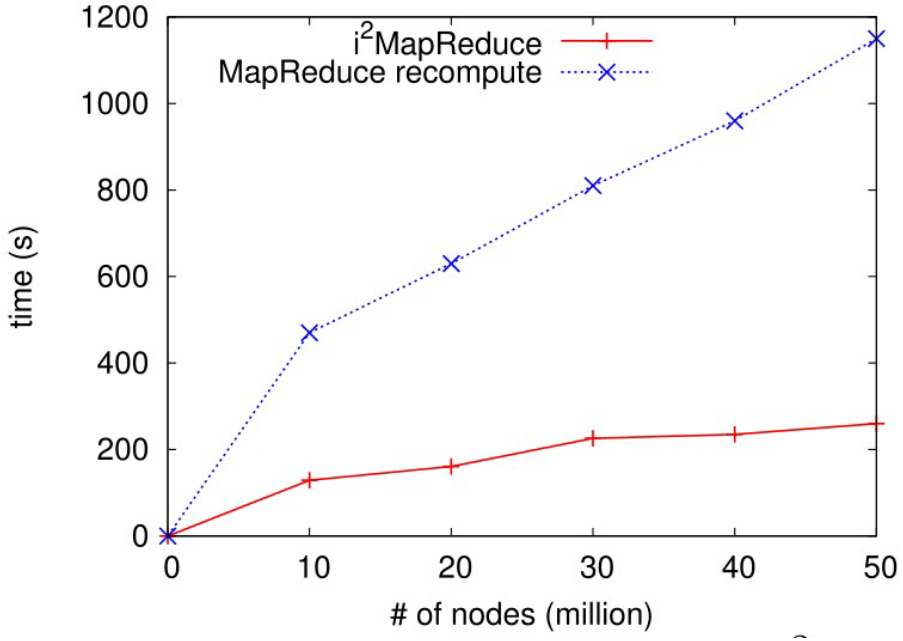
- Hadoop extension



# i²MapReduce vs. MapReduce compute

- 20-node cluster
- App: PageRank
- Synthetic power-law graph
  - Degree: log-normal dist.
  - Avg. degree 5.18
- Fixed change size
  - Randomly change 10K edges
- Varying input size
  - From 10M nodes to 50 nodes




The time of incremental processing does not change much as input size grows

# Conclusions & Future Work

- Conclusions
  - Incremental processing with MRBGraph
  - i$^2$MapReduce: a MapReduce based framework for incremental iterative computations in the cloud
- Future work
  - Indexing mechanism for querying MRBGraph file
  - Cost-aware execution plan


Thank You!

# Backup Slides: Related Work

- Incoop [SOCC 2011] (MPI-SWS)
- Spinning Fast Iterative Data Flows [VLDB 2012] (TU Berlin)
- REX [VLDB 2012] (U. Penn)
- Naiad [CIDR 2013] (Microsoft)
- Incremental Recomputions in MR [CloudDB 2011] (U. Kaiserslautern)
- IncMR [Cloud 2012] (Donghua U. China)

# Backup Slides: Building MRBGraph

- MapReduce extension

| | MapReduce | i²MapReduce |
|---|---|---|
| Map input | <MK, MV> | <MK, SV, DV> |
| Map output | <RK, RV> | <RK, MK, RV> |
| Reduce input | <RK, [RV]> | <RK, [MK, RV]> |
| Reduce output | <DK, DV> | <DK, DV> |

Mapper key & value: MK, MV
Reducer key & value: RK, RV
D: Structure data key & value: SK, SV
v: State data key & value: DK, DV