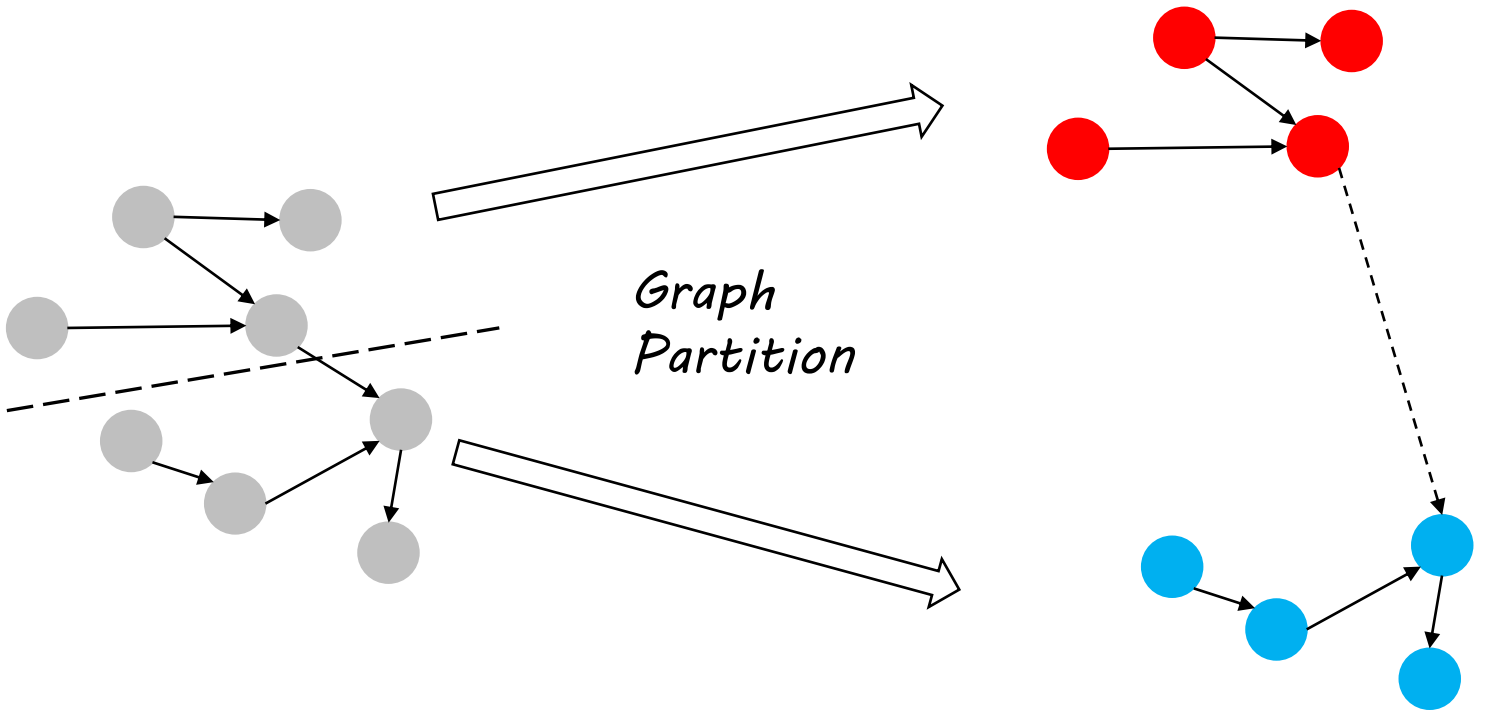
HBP: *Hotness Balanced Partition* for Prioritized Iterative Graph Computations

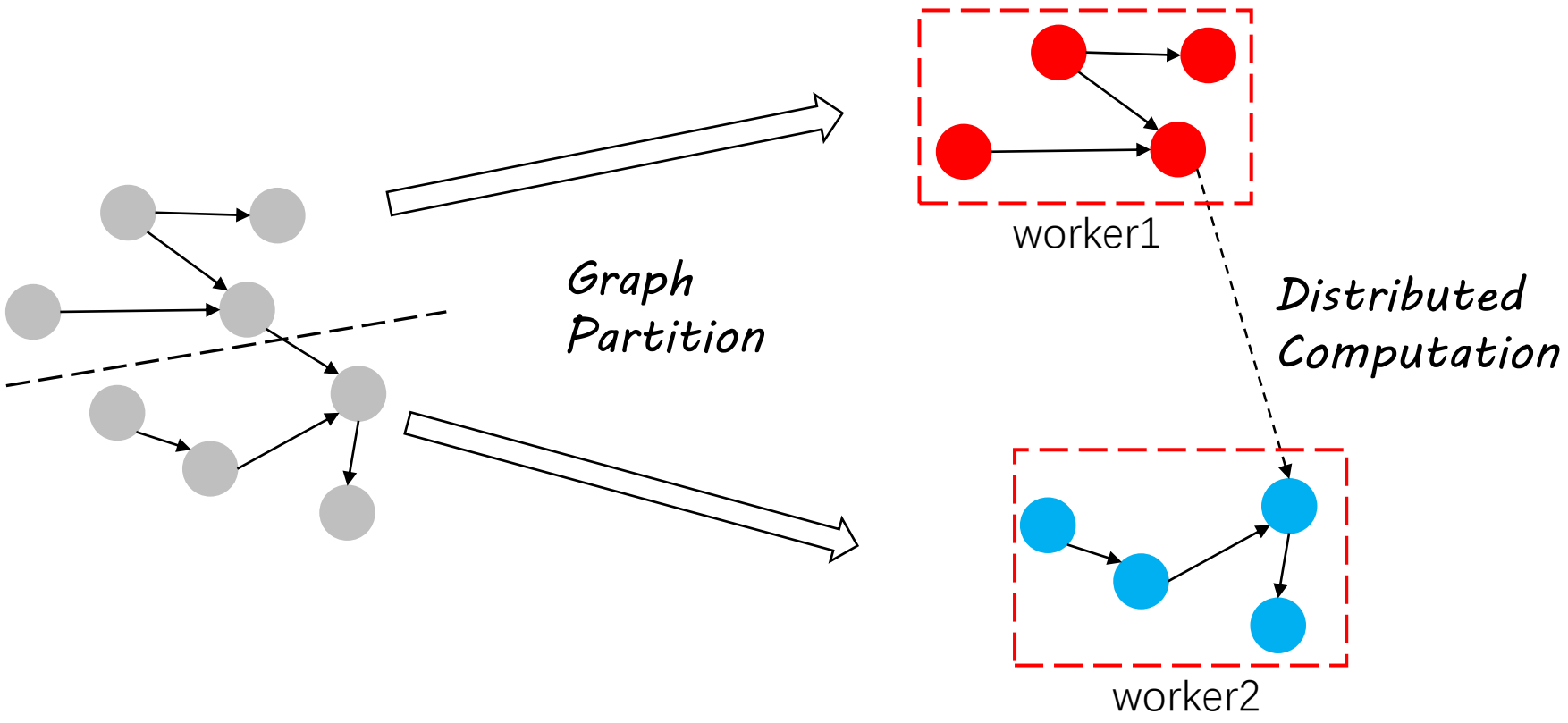
Shufeng Gong, Yanfeng Zhang, and Ge Yu

Northeastern University, China

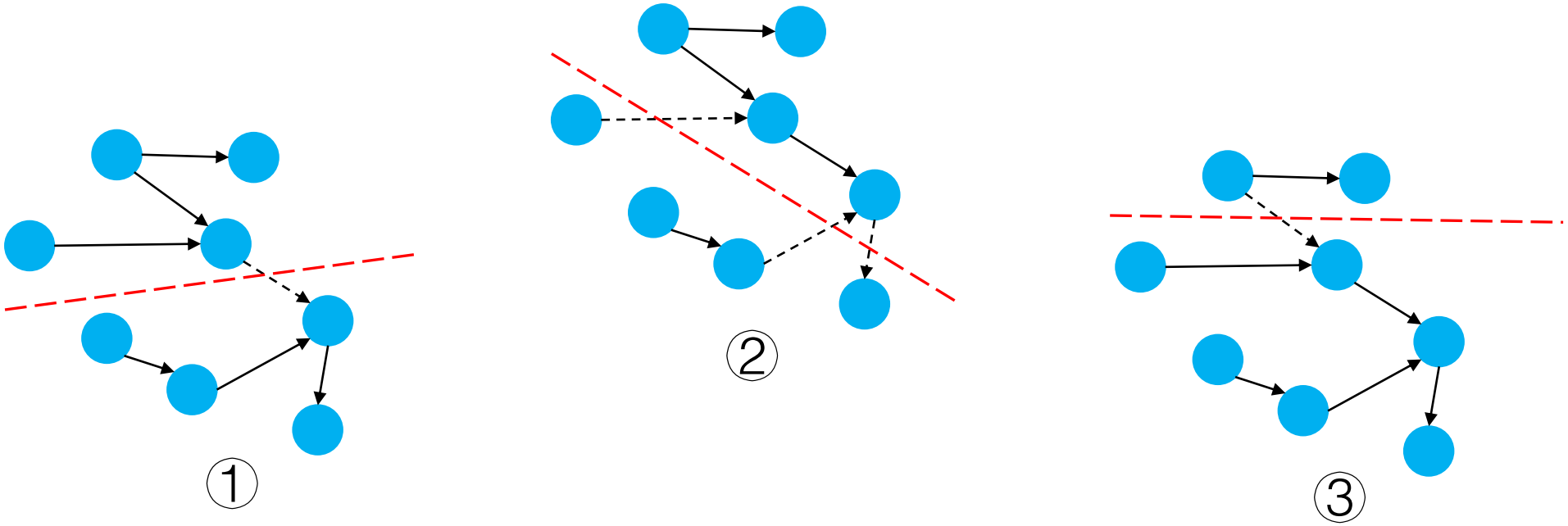
Distributed Graph Computation



Distributed Graph Computation

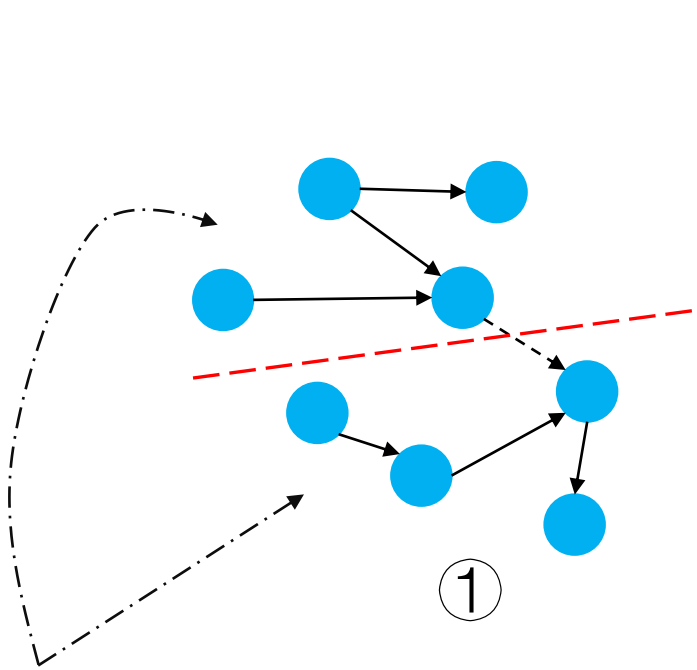


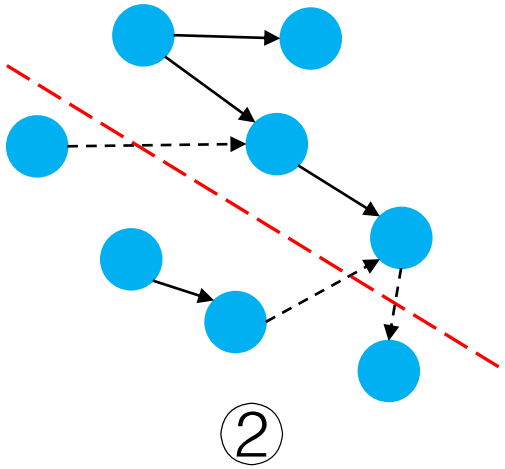
Graph Partition

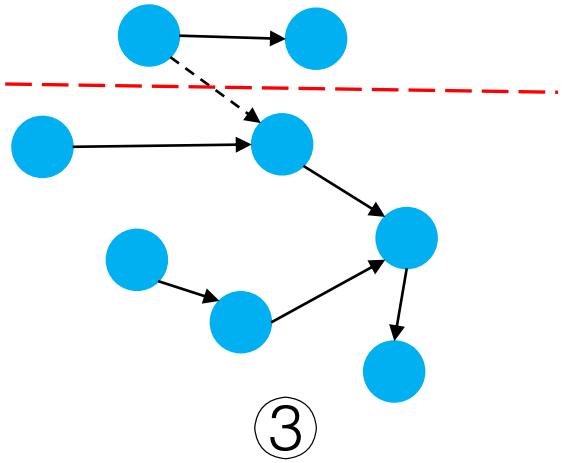


Different graph partition schemes may result in different graph processing performance.

Graph Partition

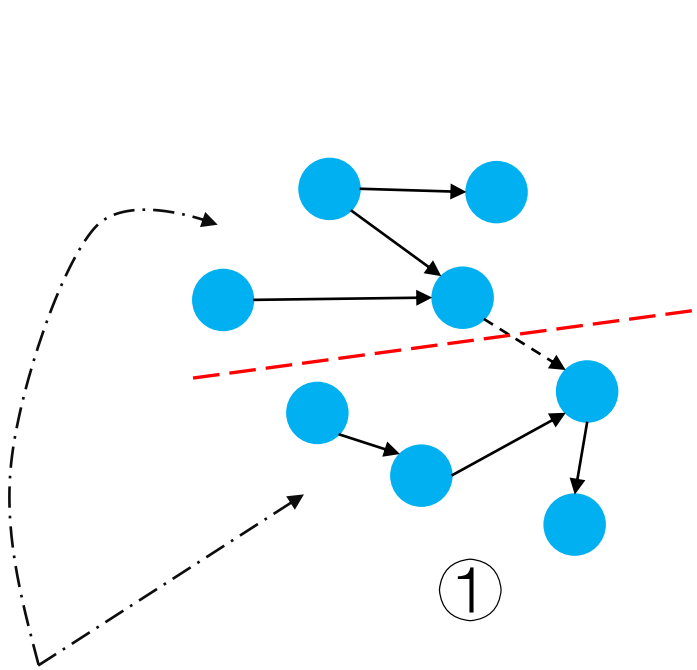


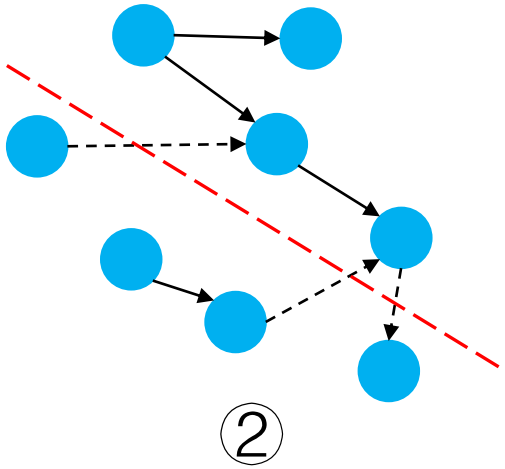


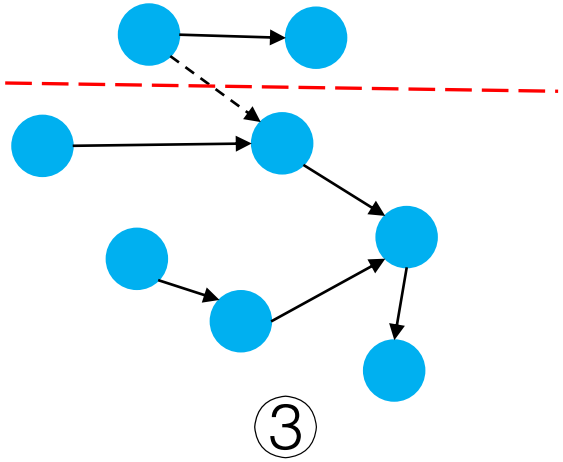


each partition has the same number of vertices (four vertices)

Graph Partition



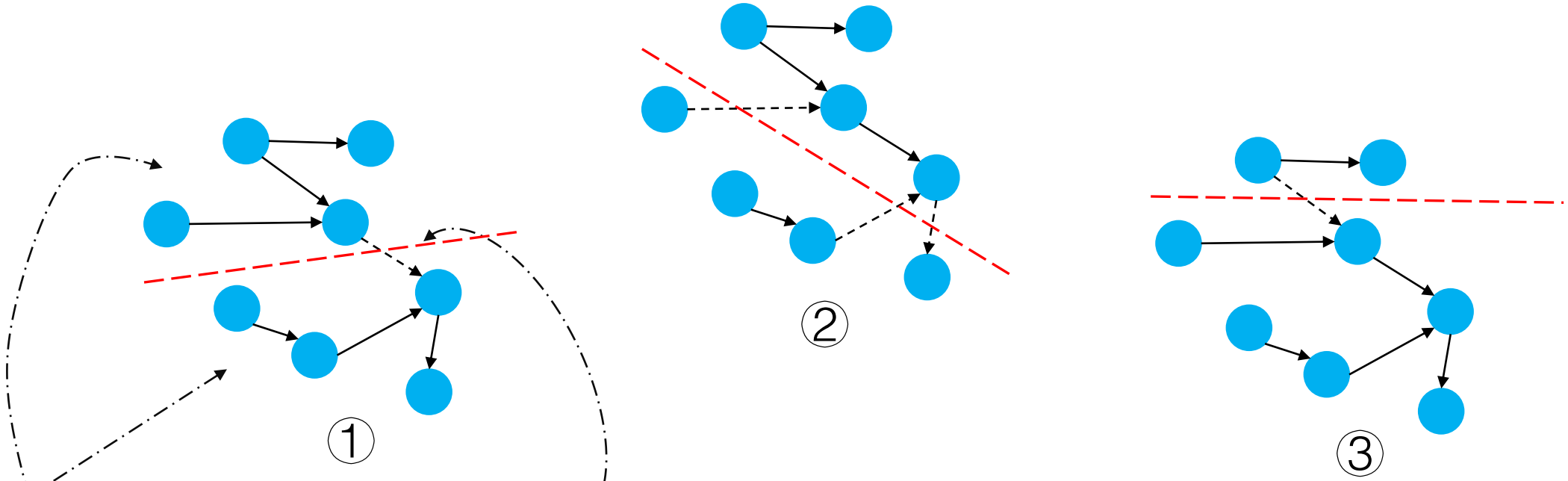




each partition has the same number of vertices (four vertices)

Workload Balance

Graph Partition

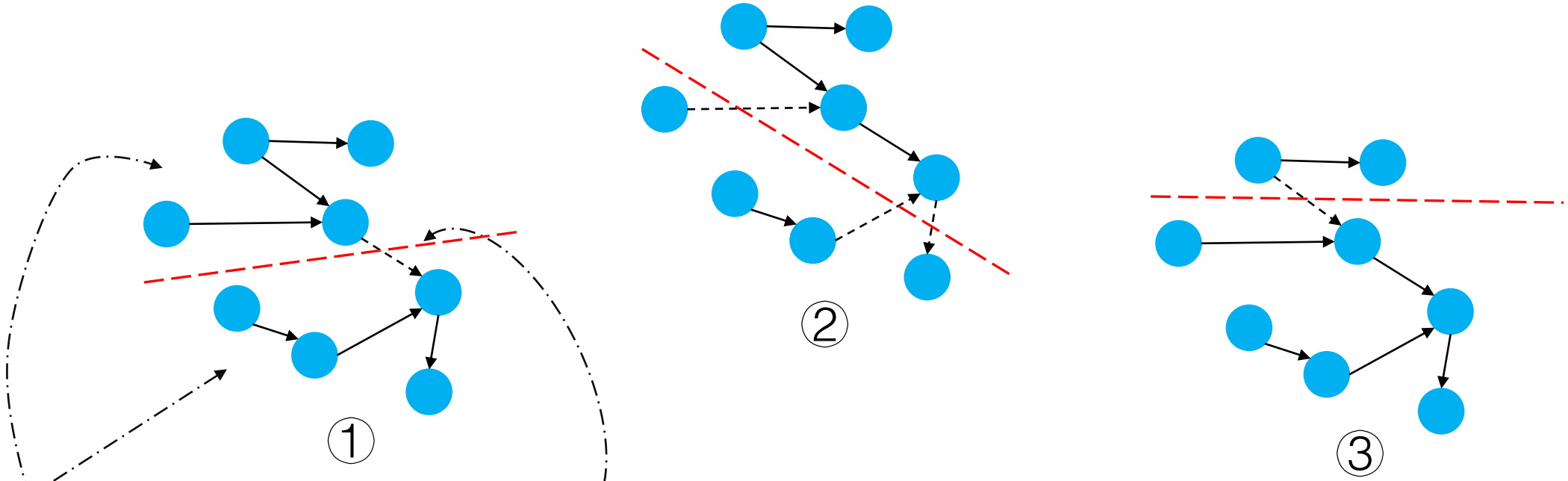


each partition has the same number of vertices (four vertices)

one pair of vertices communicate between different partitions

Workload Balance

Graph Partition



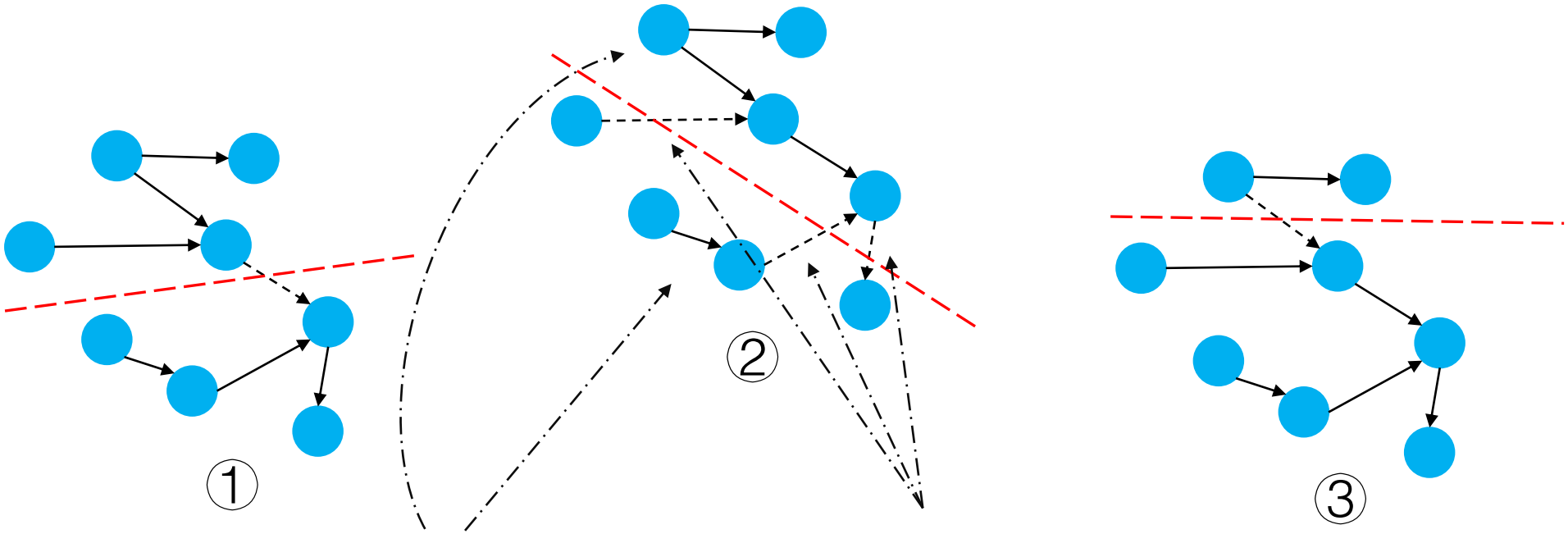
each partition has the same number of vertices (four vertices)

Workload Balance

one pair of vertices communicate between different partitions

Less Communication

Graph Partition



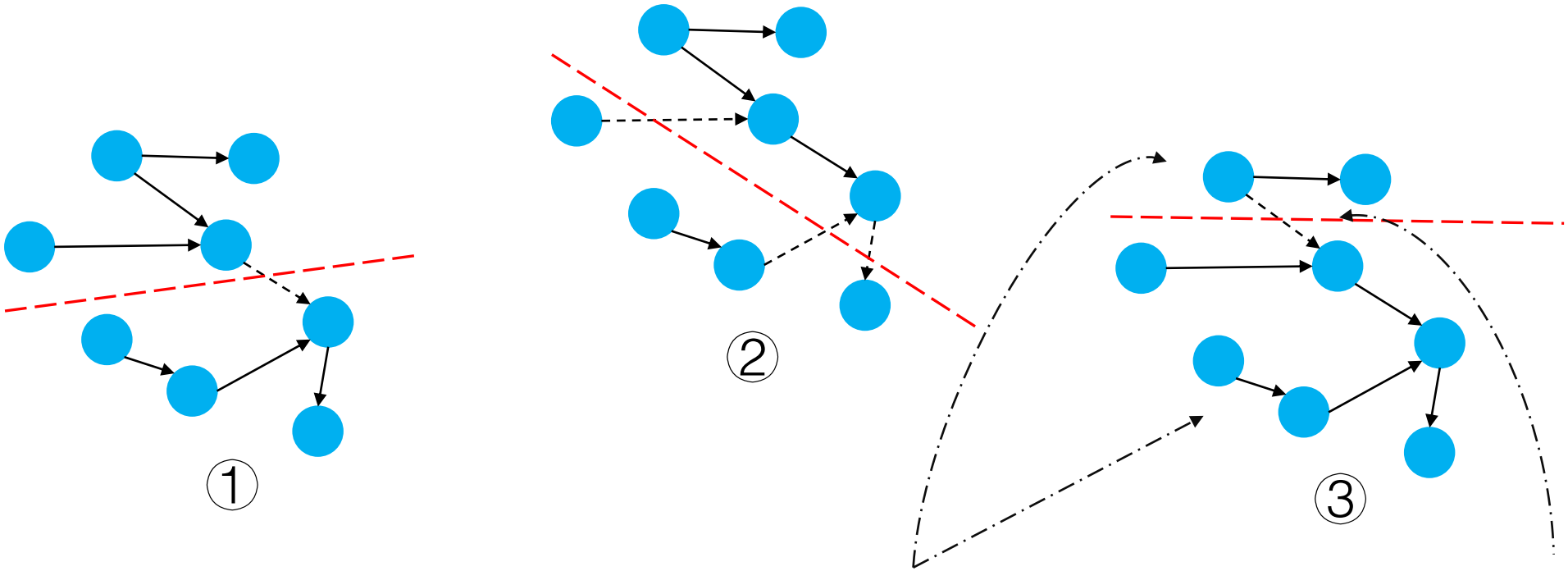
each partition has the same number of vertices (four vertices)

Workload Balance

three pairs of vertices communicate between different partitions

More Communication

Graph Partition



each partition has a different number of vertices (two vertices vs. six vertices)

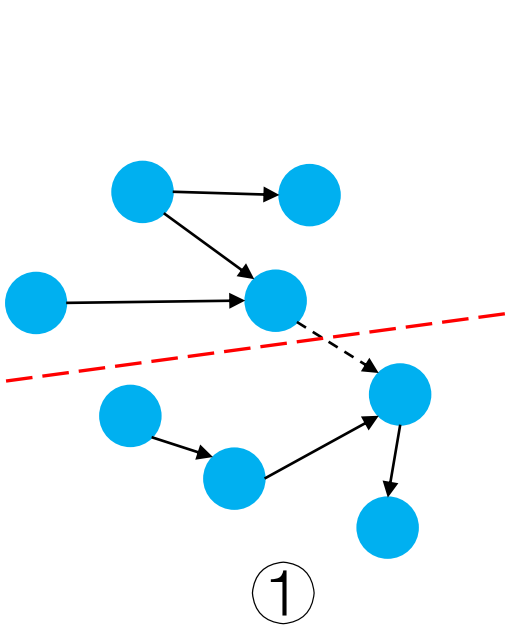
Workload Imbalance

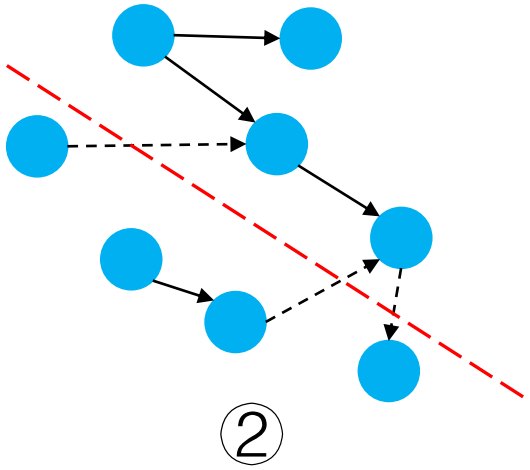
wait time

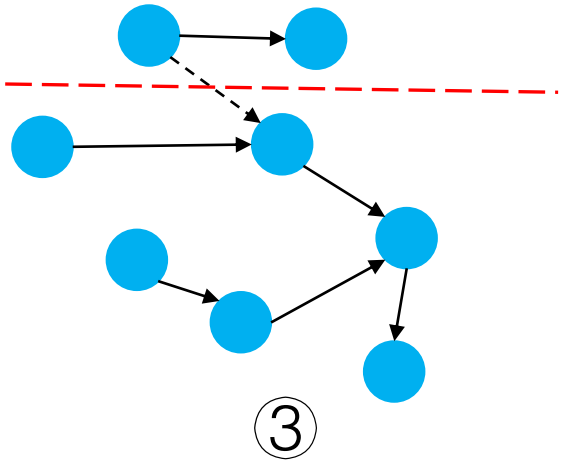
one pair of vertices communicate between different partitions

Less Communication

Graph Partition



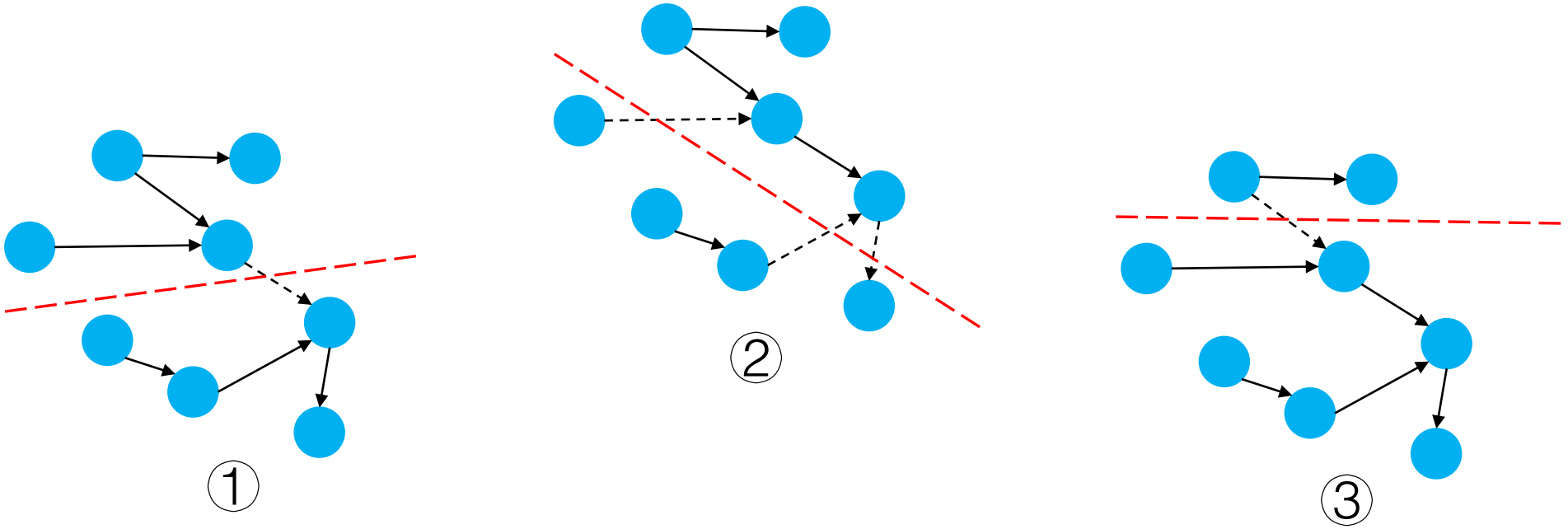




A good partition can

1. Reduce wait time
2. Minimize communication cost

Graph Partition



Goals:

1. Balance the number of vertices
2. Minimize the number of edge cuts

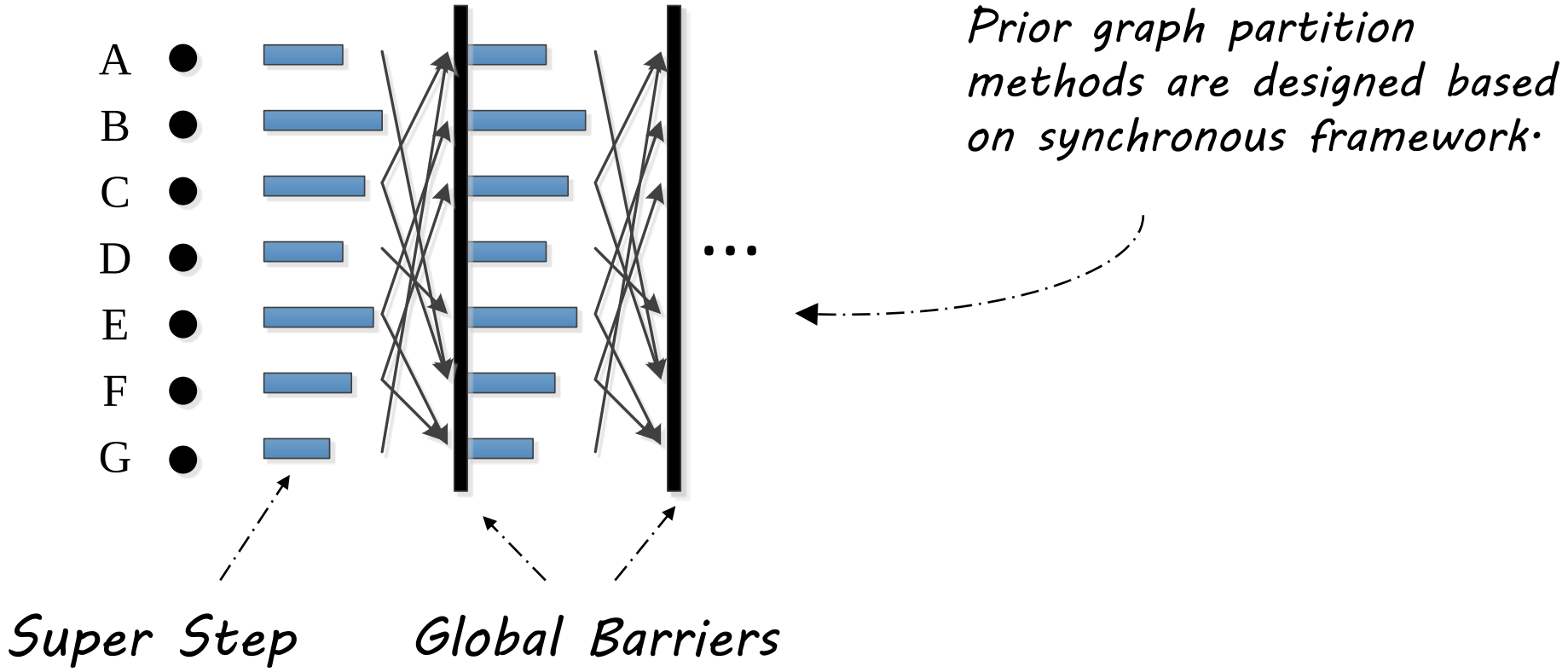
Related Works

Metis[1], Leopard[2], Fennel[3], HDRF[4],Spinner[5], Sheep[6], NE[7]

They assume the workload depends on vertices (vertex centric) or edges (edge centric), and the communication cost depends on the number of edge cuts or vertex replications.

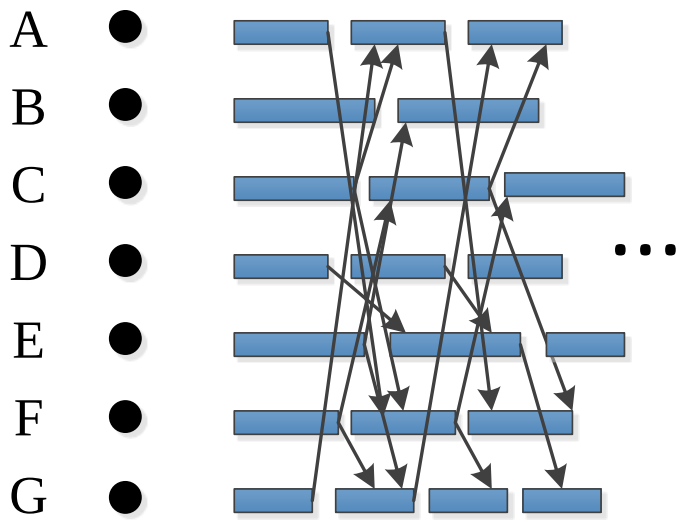
1. *Karypis, George, and Vipin Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs." SISC 20.1 (1998): 359-392.*
2. *Huang, Jiewen, and Daniel J. Abadi. "Leopard: Lightweight edge-oriented partitioning and replication for dynamic graphs." Proceedings of the VLDB 2016.*
3. *Tsourakakis, Charalampos, et al. "Fennel: Streaming graph partitioning for massive scale graphs." WSDM 2014.*
4. *Petroni, Fabio, et al. "HDRF: Stream-based partitioning for power-law graphs." CIKM 2015.*
5. *Martella, Claudio, et al. "Spinner: Scalable graph partitioning in the cloud." ICDE 2017.*
6. *Margo, Daniel, and Margo Seltzer. "A scalable distributed graph partitioner." VLDB 2015.*
7. *Zhang, Chenzi, et al. "Graph edge partitioning via neighborhood heuristic." KDD 2017.*

Synchronous Frameworks

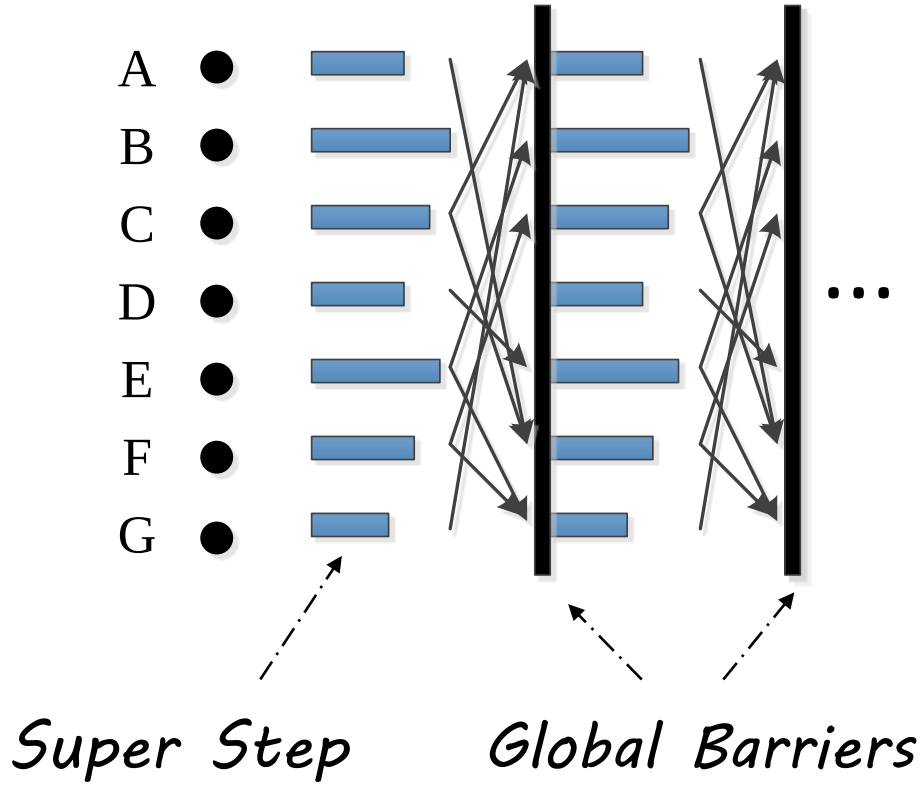


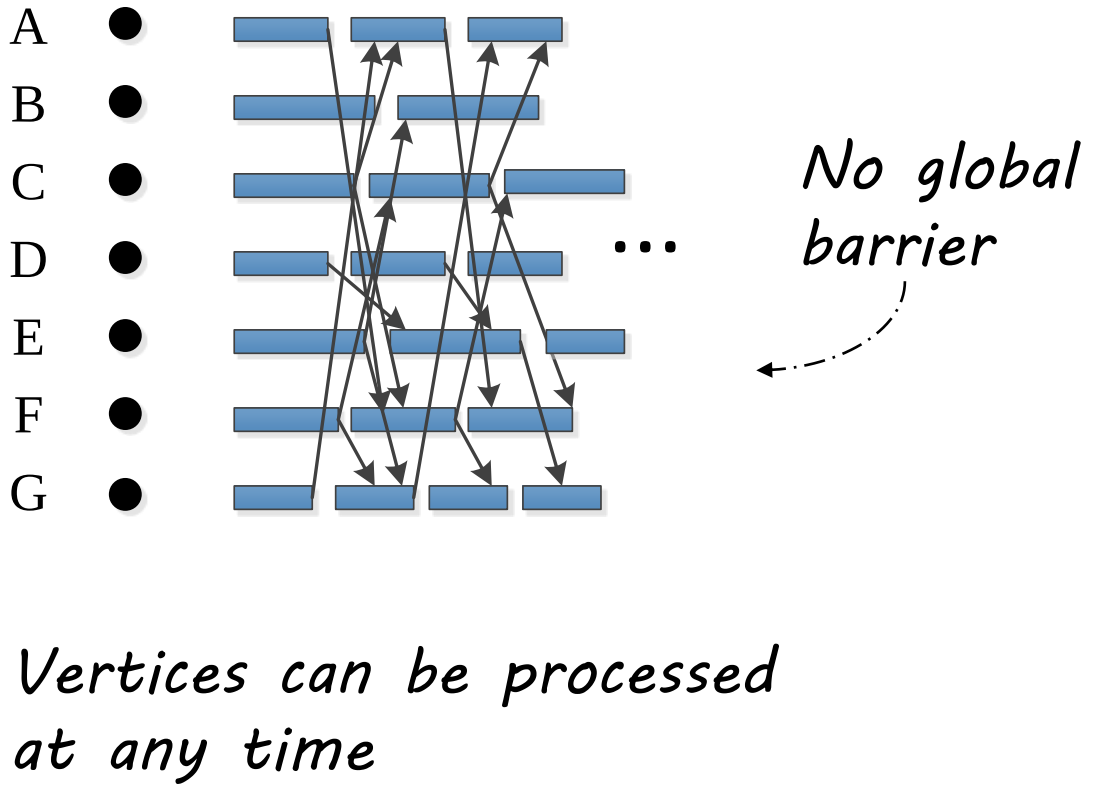
Sync & Async

Some work pays attention to asynchronous frameworks since they can accelerate a class of iterative algorithms.

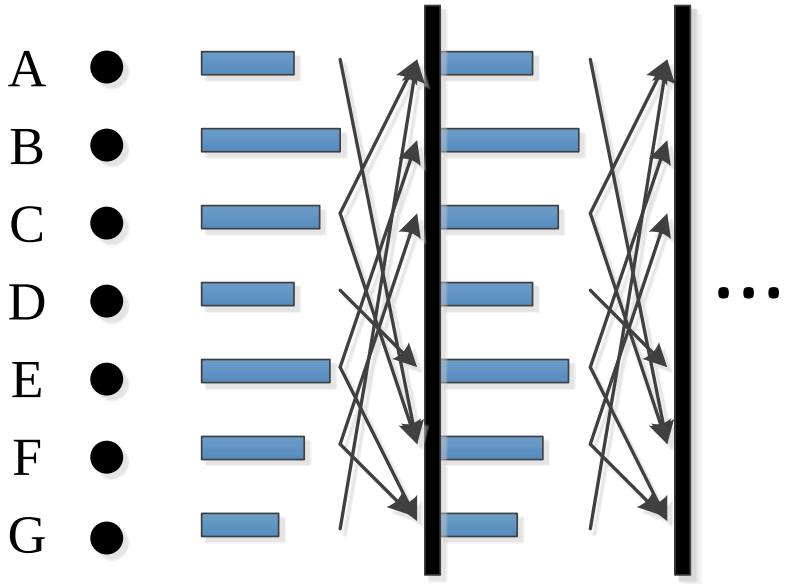


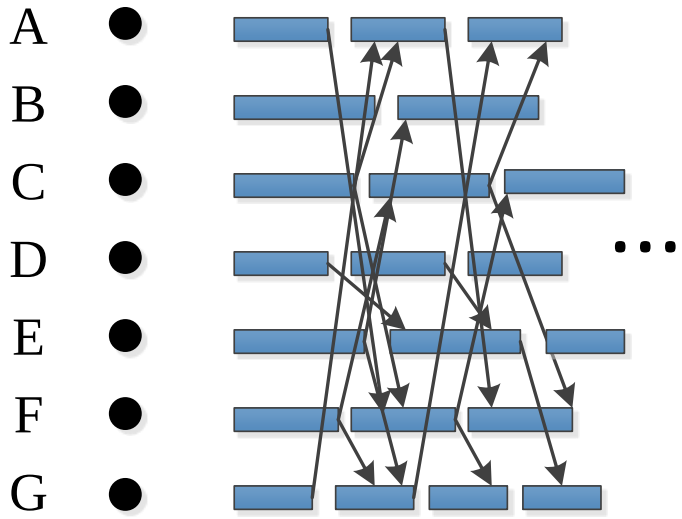
Sync & Async


A
B
C
D
E
F
G
Super Step
Global Barriers


A
B
C
D
E
F
G
No global barrier
Vertices can be processed at any time

Sync & Async

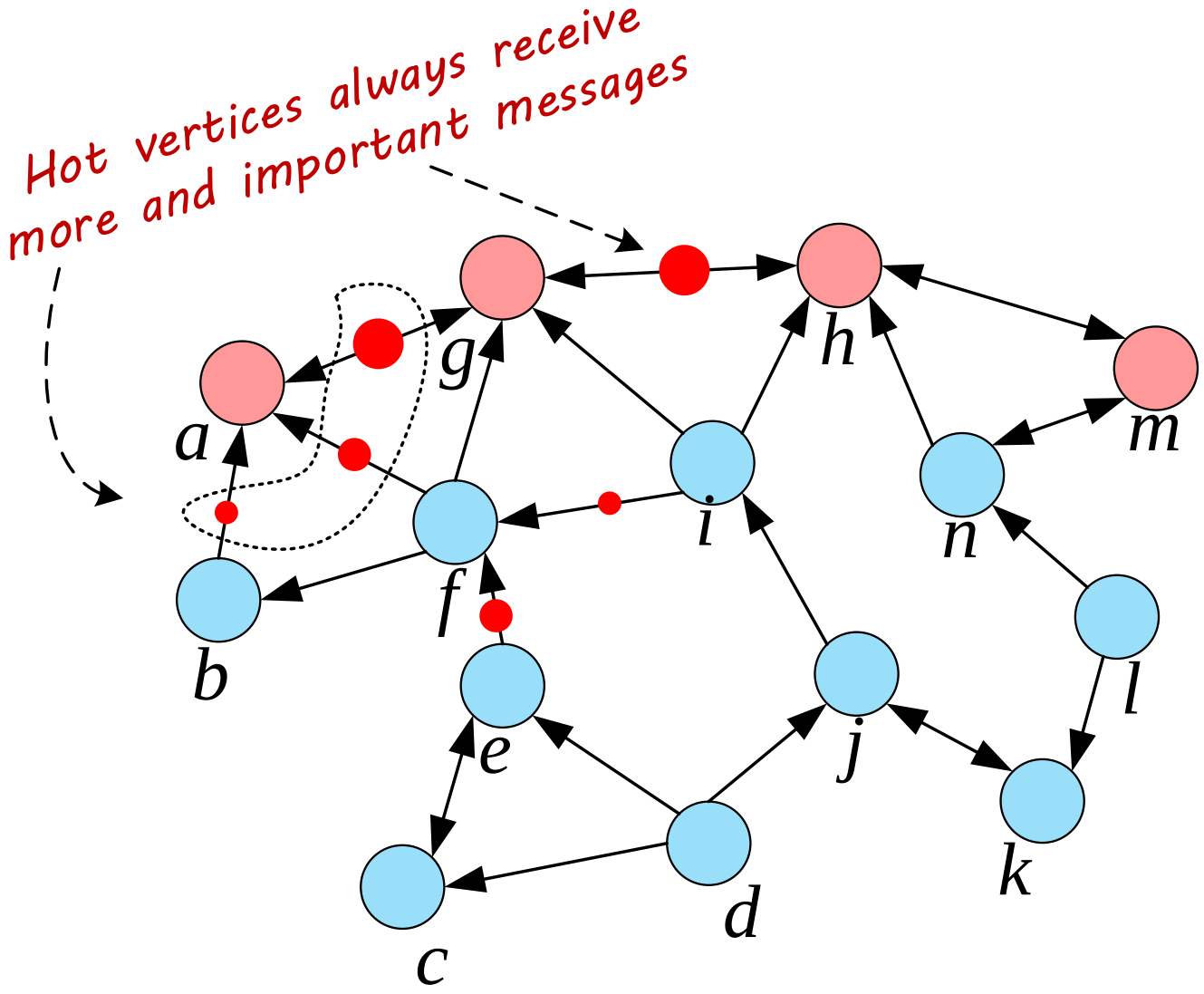




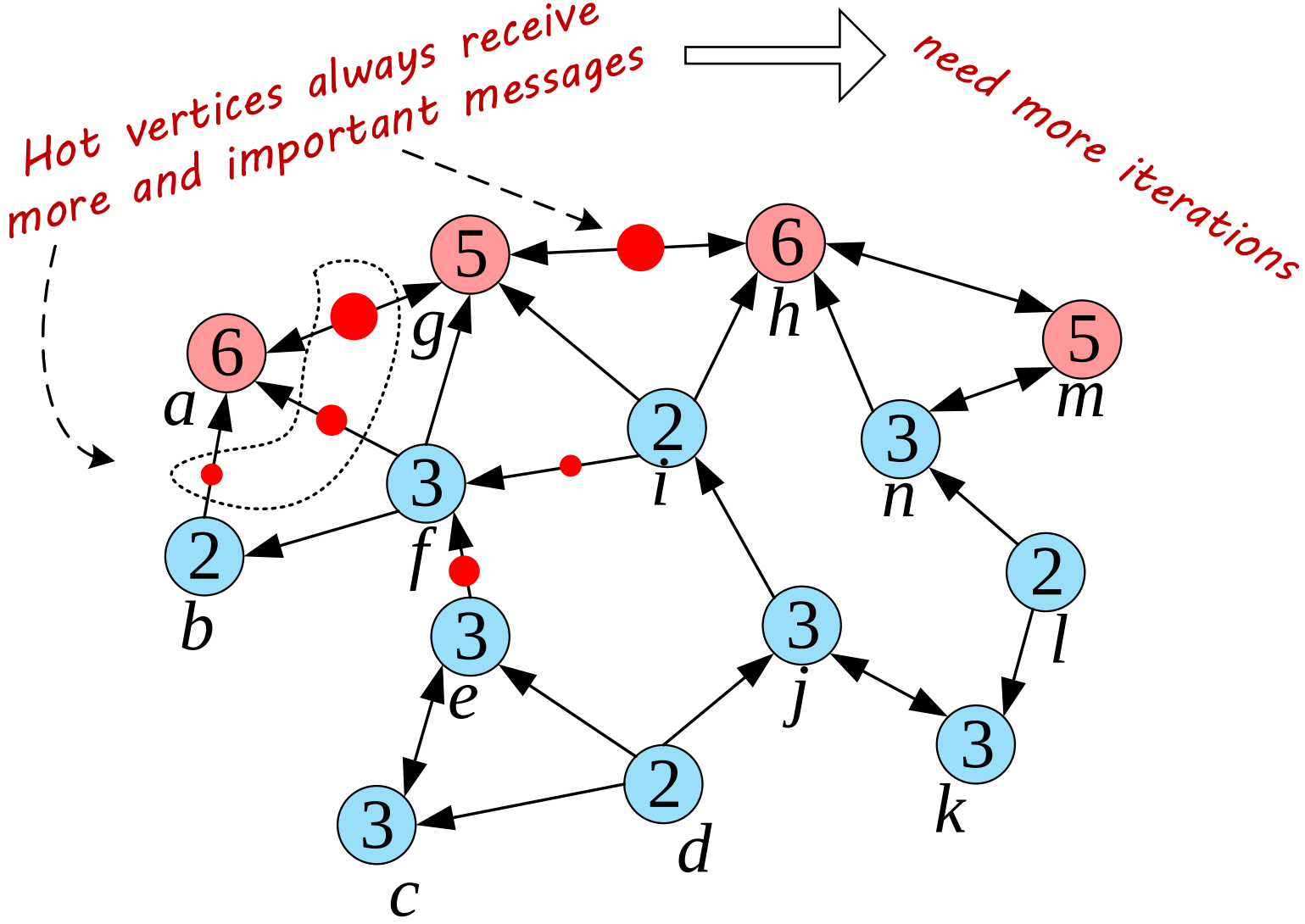
In each super step, each vertex is only processed once, and each edge only delivers one message.

The number of updates on each vertex and the number of messages passed through each edge are not consistent.

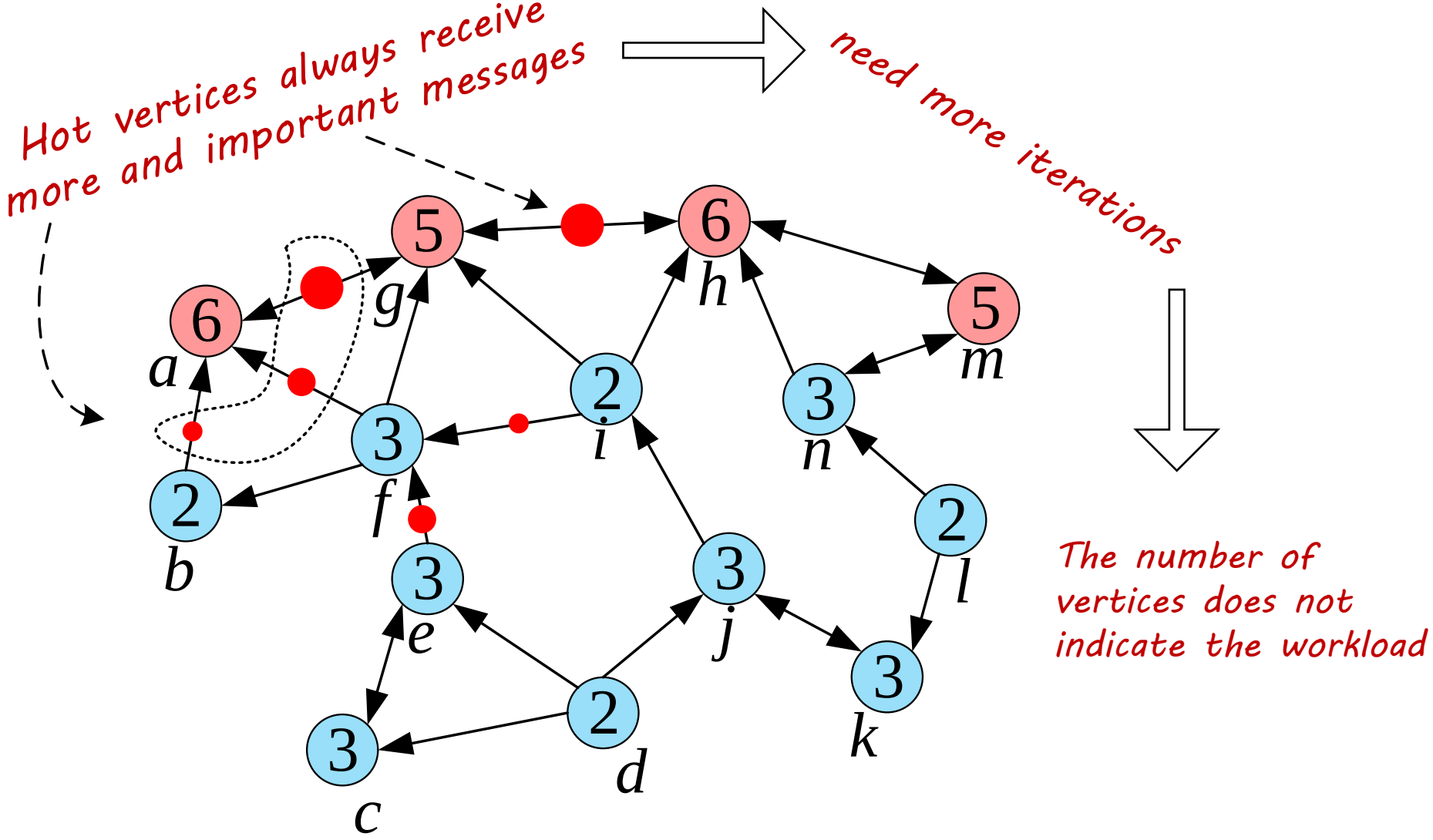
Hotness of Vertices



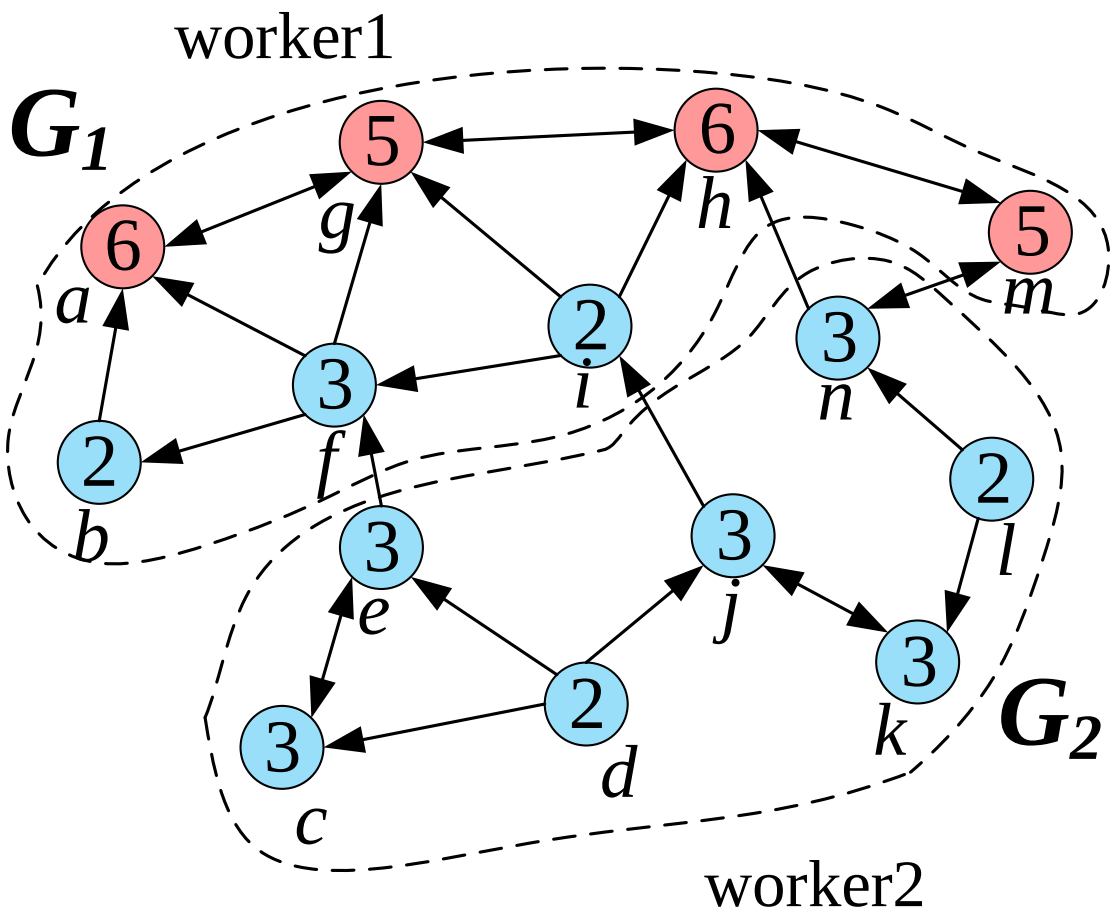
Hotness of Vertices



Hotness of Vertices



Vertex Balanced Partition

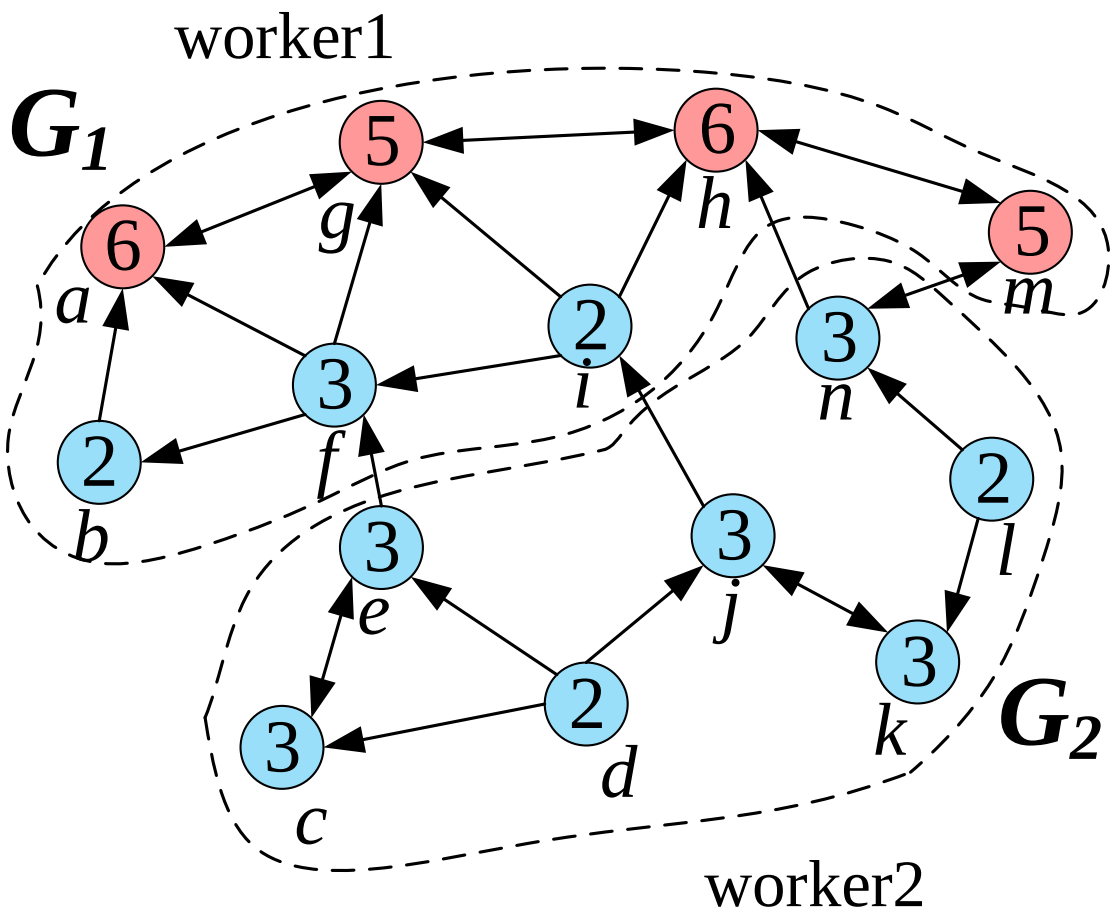


vertex balance

Partition graph into G1 and G2

Worker1 processes G1
Worker2 processes G2

Vertex Balanced Partition

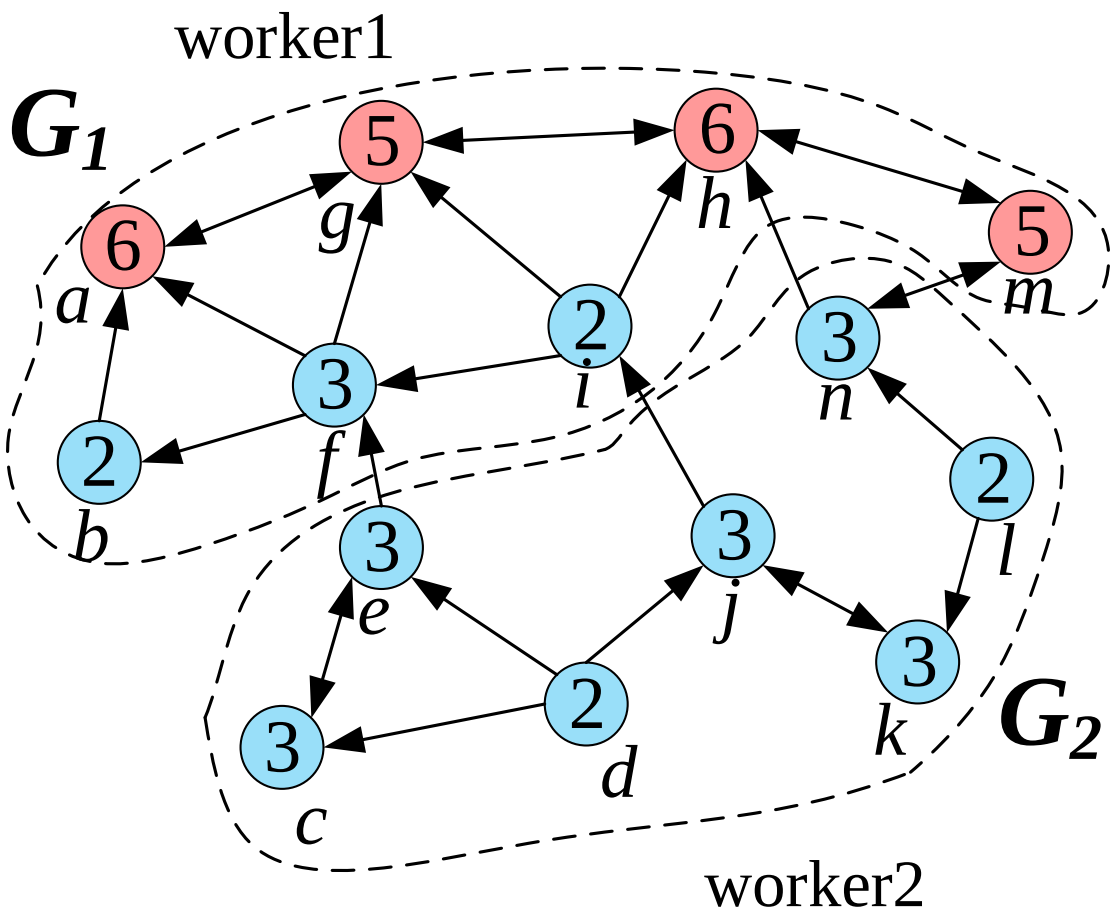


vertex balance

The number of vertices in G1 and G2 is equal.
The sum of hotness in G1 and G2 is not equal.

Prior vertex balanced graph partition methods do not take the vertex hotness into account when partitioning graphs.

Vertex Balanced Partition

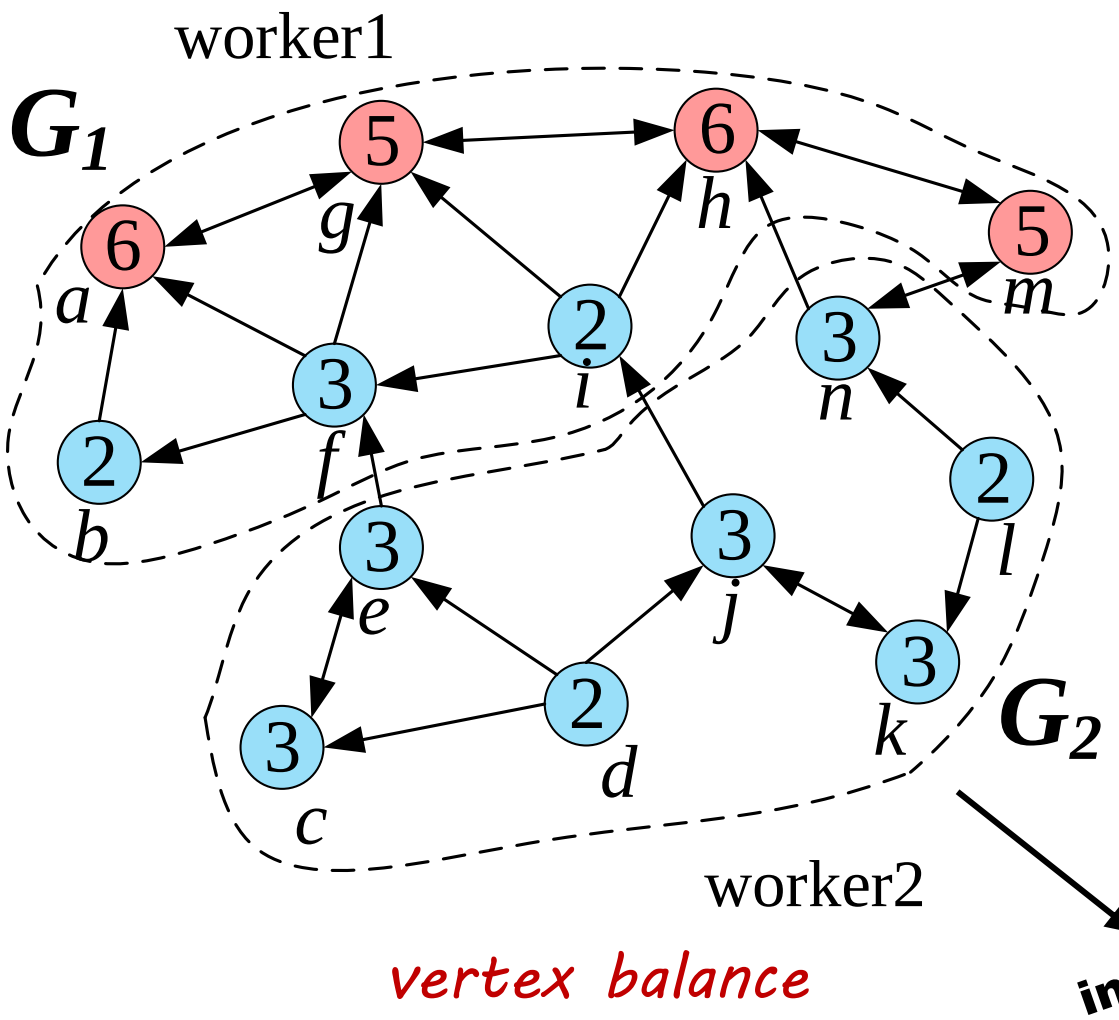


vertex balance

Two workers are running.

There may be a lot of invalid computation in woker2 since vertices in G2 need fewer iterations.

Vertex Balanced Partition

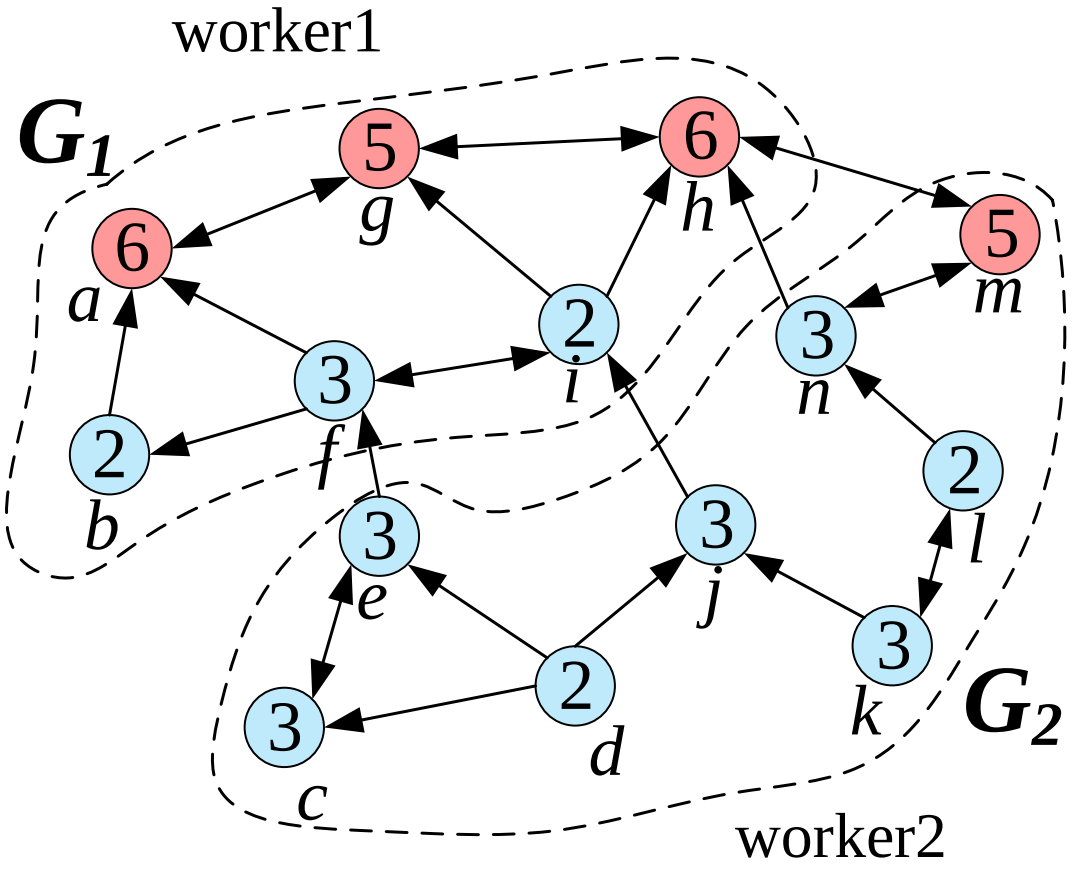


Two workers are running.

There may be a lot of invalid computation in woker2 since vertices in G2 need fewer iterations.

Hotness Balanced Partition

Sum of hotness of G2 and G1 is equal



hotness balance

Hotness Balanced Partition

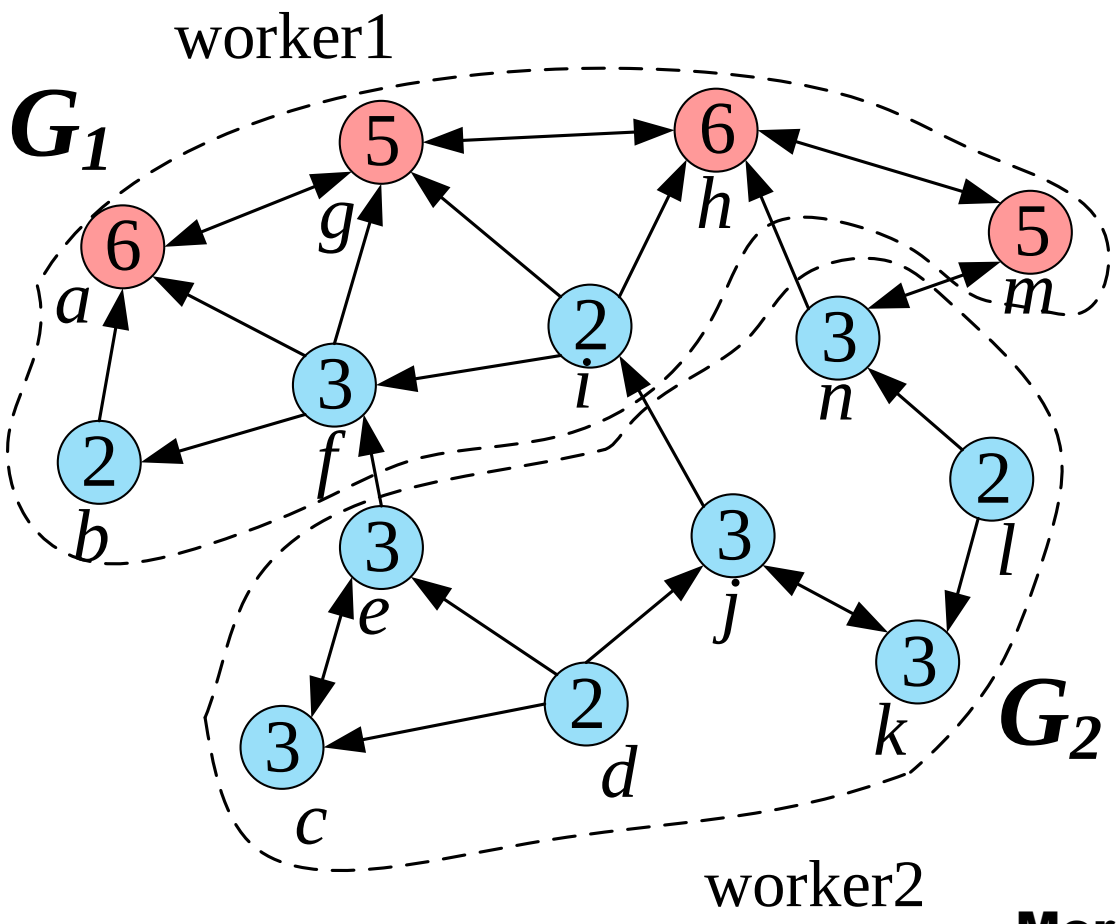
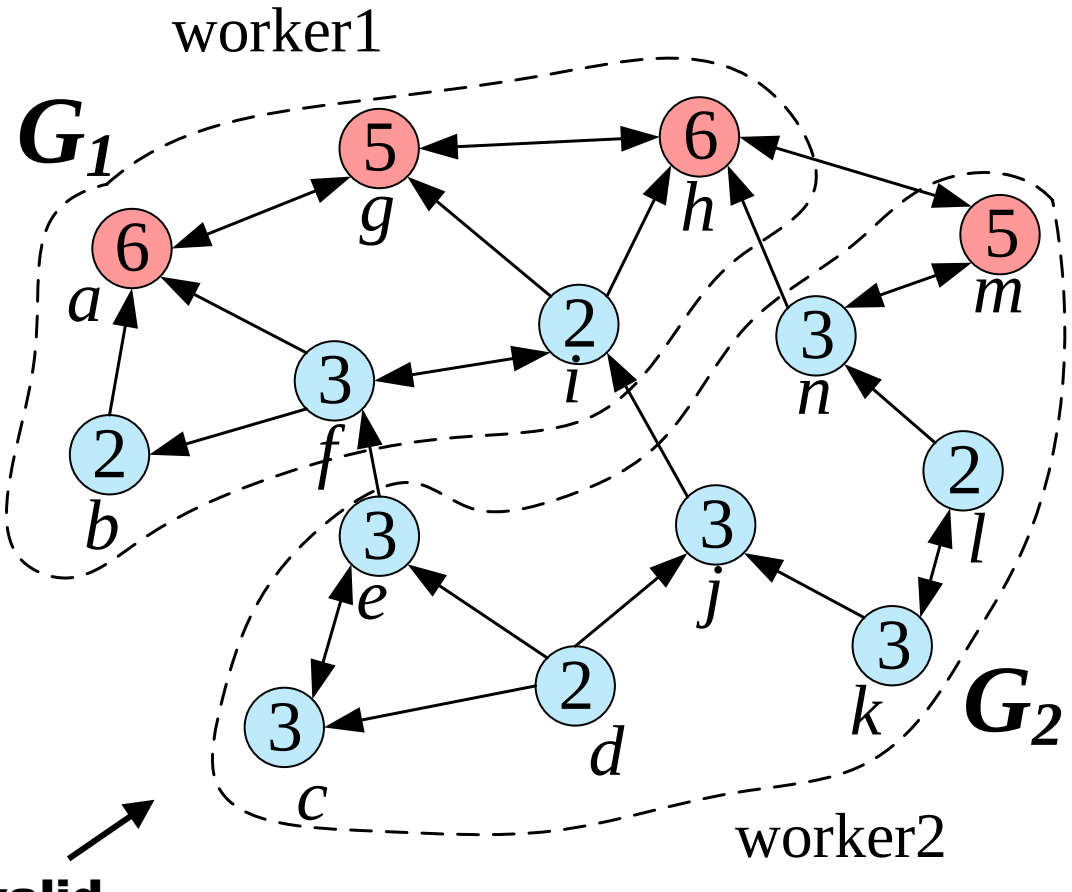


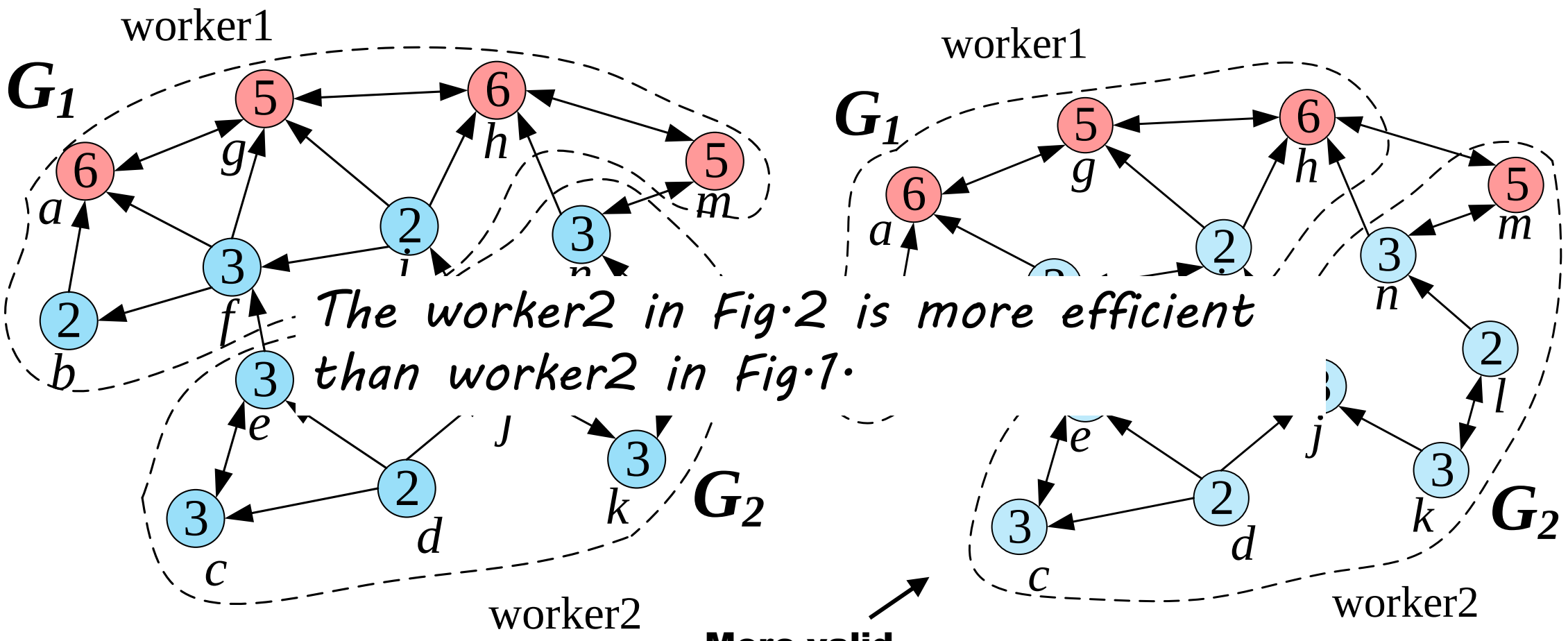
Fig. 1: vertex balance



More valid computation

Fig. 2: hotness balance

Hotness Balanced Partition

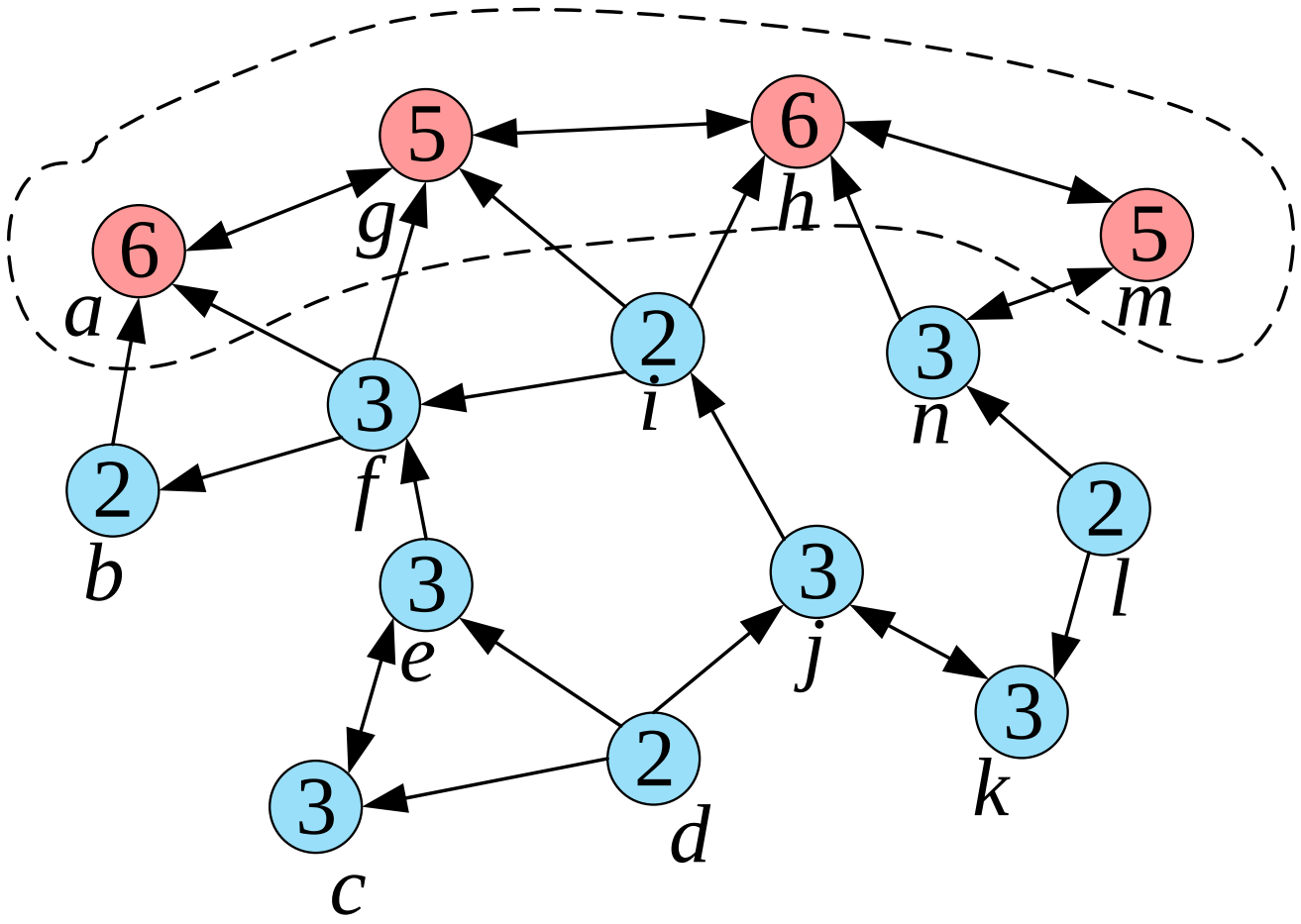


Fig· 1: vertex balance

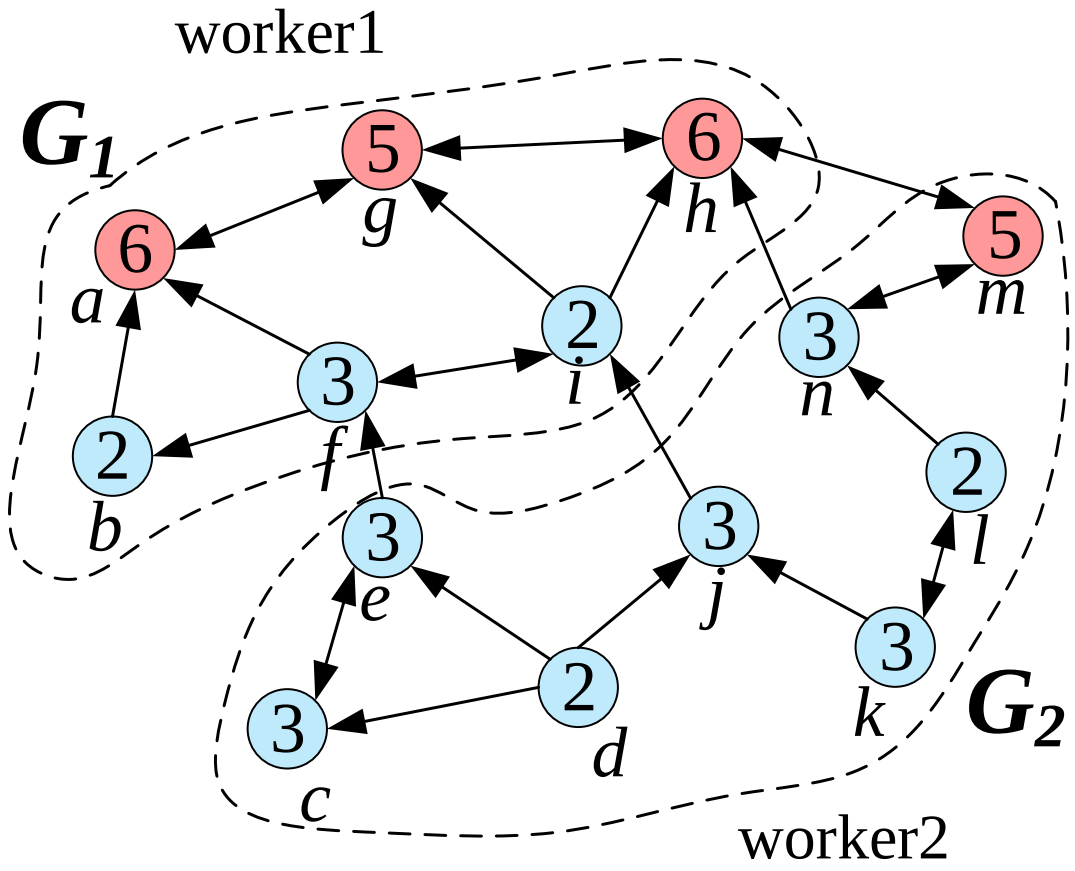
Fig· 2: hotness balance

Priority Scheduling

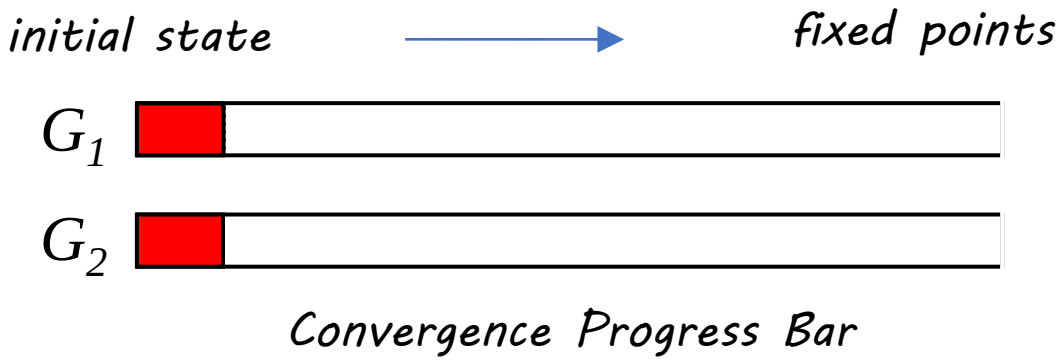
Accelerate graph computing by selecting hot vertices to process preferentially.



Hotness Balanced Partition



hotness balance

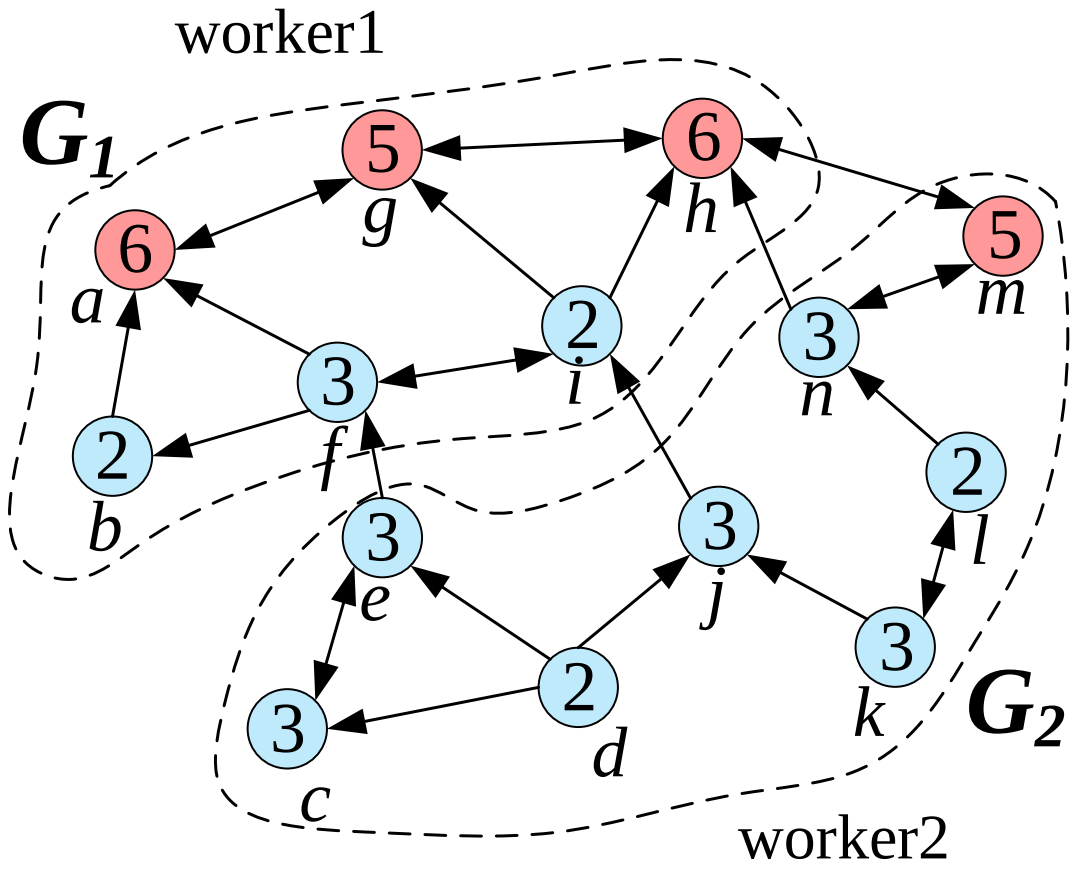


Convergence Progress Bar

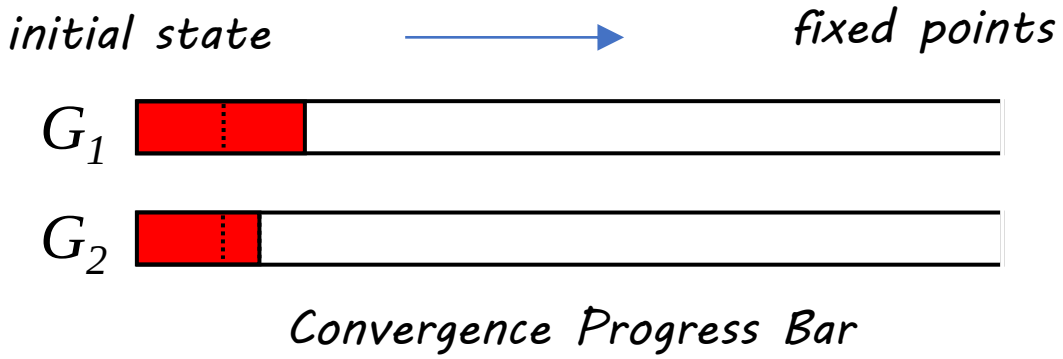
Worker1 process a

Worker2 process m

Hotness Balanced Partition



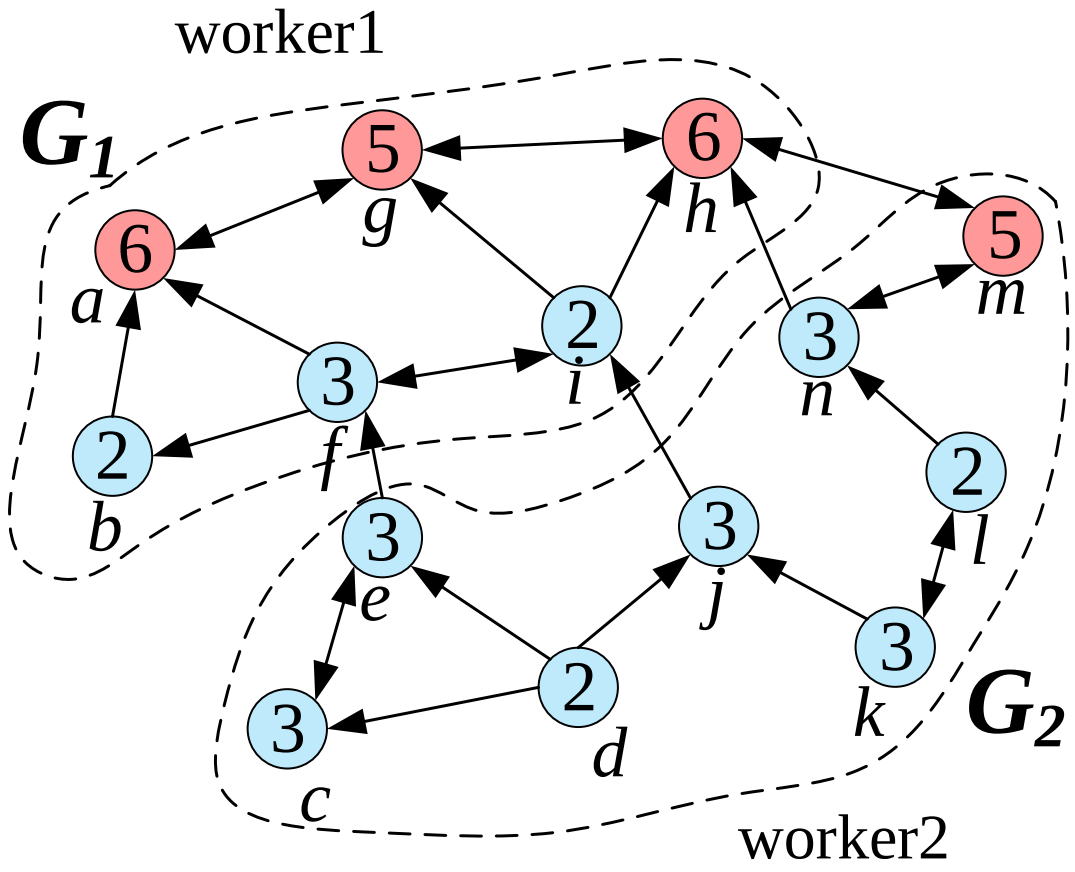
hotness balance



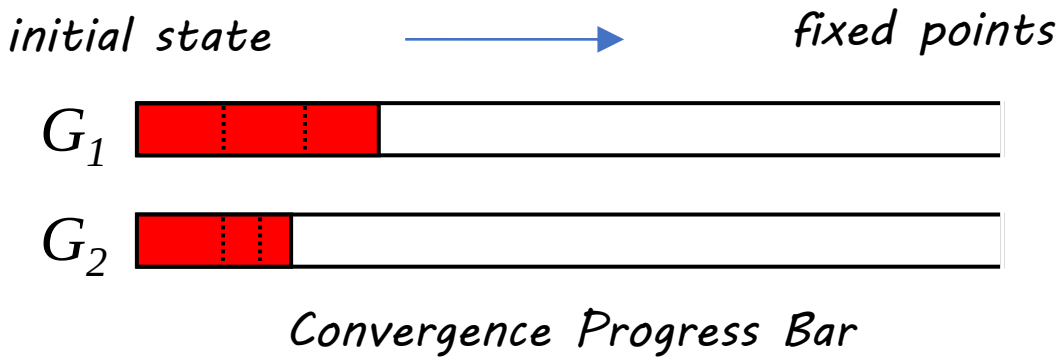
Worker1 process g
Worker2 process n

A hot vertex can make more contributions to approaching the fixed point.

Hotness Balanced Partition



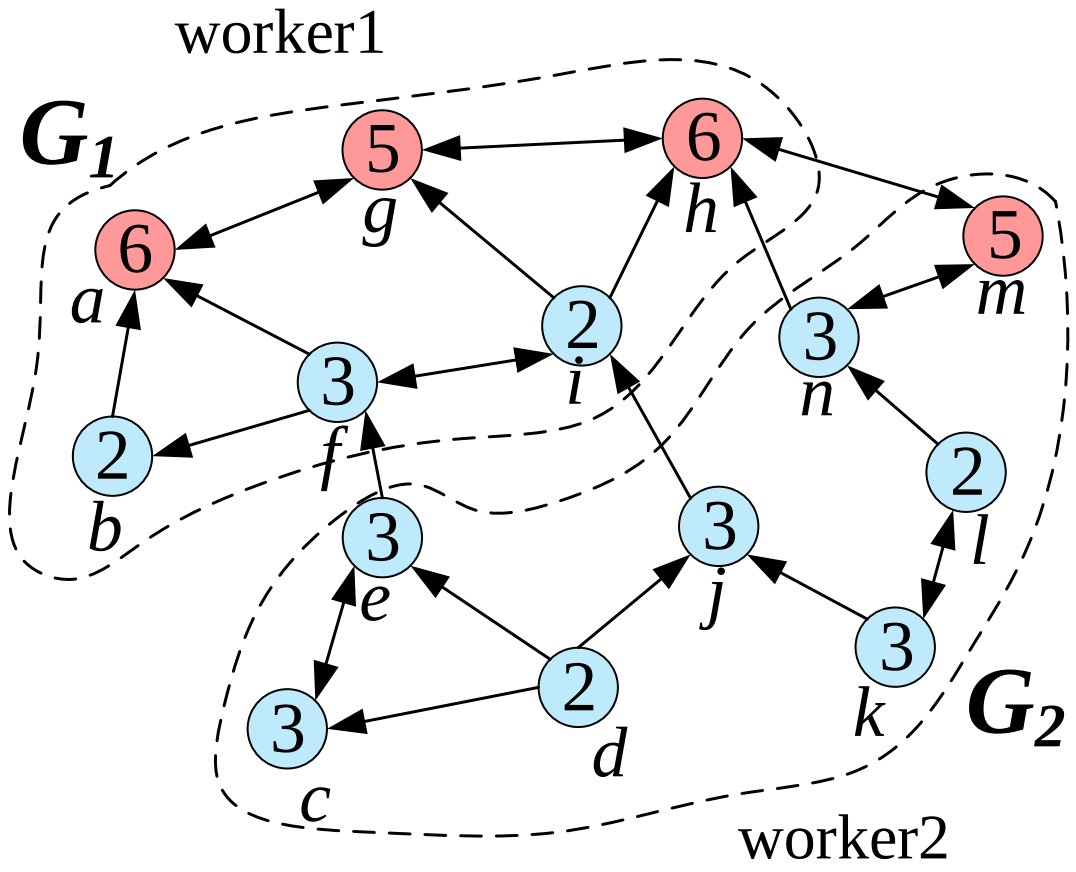
hotness balance



Worker1 process h
Worker2 process l

A hot vertex can make more contributions to approaching the fixed point.

Hotness Balanced Partition



hotness balance

The worker2 is still not as efficient as we expected, because no matter how to schedule, worker2 always processes vertices with low hotness.

The hotness distribution of each partition should be the same as that of the original graph.

Partition Goals

prioritized iterative graph computations

- *Assign the same amount of vertex hotness to workers*
- *Minimize the variance between hotness distributions of each partition and the original graph.*
- *Minimize the communication cost between partitions.*

both synchronous and asynchronous frameworks

Partition Goals

- *Assign the same amount of vertex hotness to workers*
- *Minimize the variance between hotness distributions of each partition and the original graph.*
- *Minimize the communication cost between partitions.*

$$HJS(P||P_i) = \frac{1}{2}\left[\sum_{j=1}^{z} P(\mathcal{H}_j) log\left(\frac{P(\mathcal{H}_j)}{\frac{P(\mathcal{H}_j)+P_i(\mathcal{H}_j)}{2}}\right) + \sum_{j=1}^{z} P_i(\mathcal{H}_j) log\left(\frac{P_i(\mathcal{H}_j)}{\frac{P_i(\mathcal{H}_j)+P(\mathcal{H}_j)}{2}}\right)\right]$$

$$P(\mathcal{H}_j) = \frac{\sum_{v\in\mathcal{H}_j} h_v}{\sum_{v\in V} h_v}$$

HJS distance is to measure the variance between hotness distributions of each partition and the original graph.

Partition Goals

- *Assign the same amount of vertex hotness to partitions*
- *Minimize the HJS distance between each partition and the original graph.*
- *Minimize the communication cost between partitions.*

Partition Goals

- *Assign the same amount of vertex hotness to workers*
- *Minimize the HJS distance between each partition and the original graph.*
- *Minimize the communication cost between partitions.*

When balancing the hotness of each bin partition:

1. The amount of vertex hotness of each partition is equal.
2. *HJS* distance between each partition and the original graph is 0.

Partition Goals

The first two partition goals can be combined into one:

- *Balance the hotness of each bin partition.*
- *Minimize the communication cost between partitions.*

Per-Bin Hotness Balanced Partition

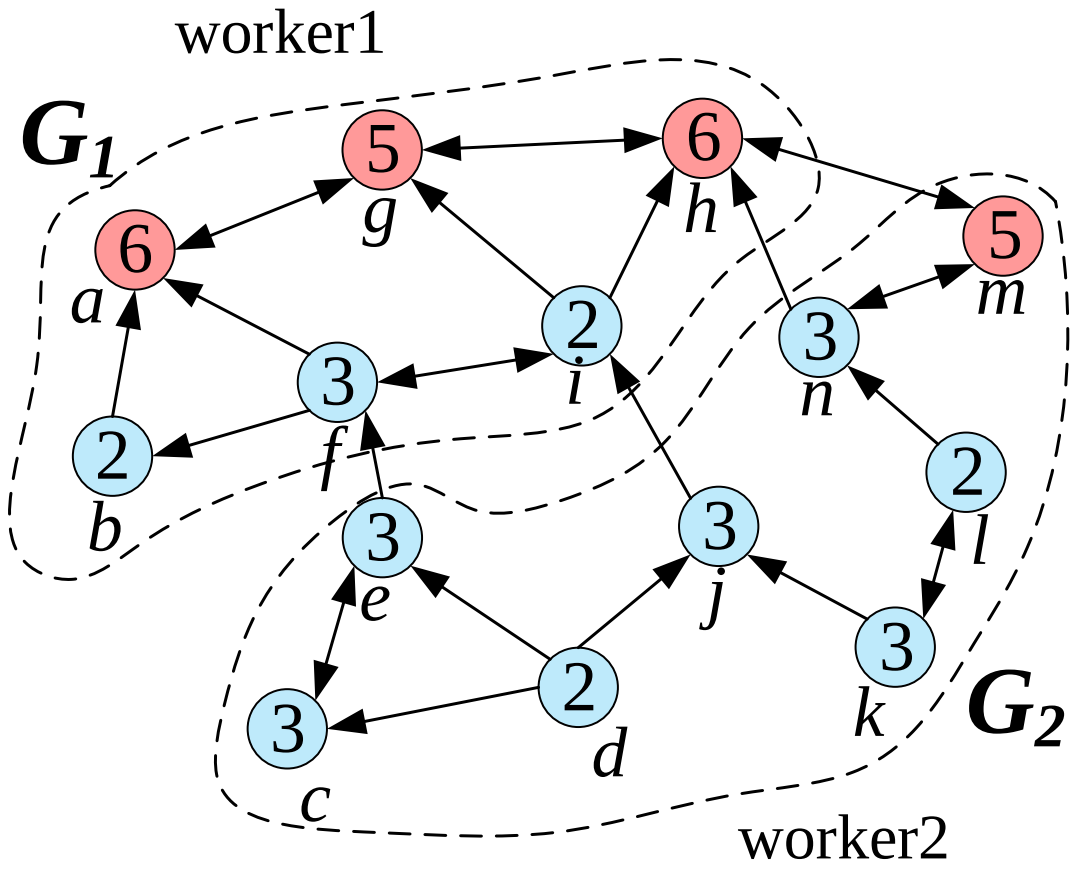


Fig. 1: hotness balance

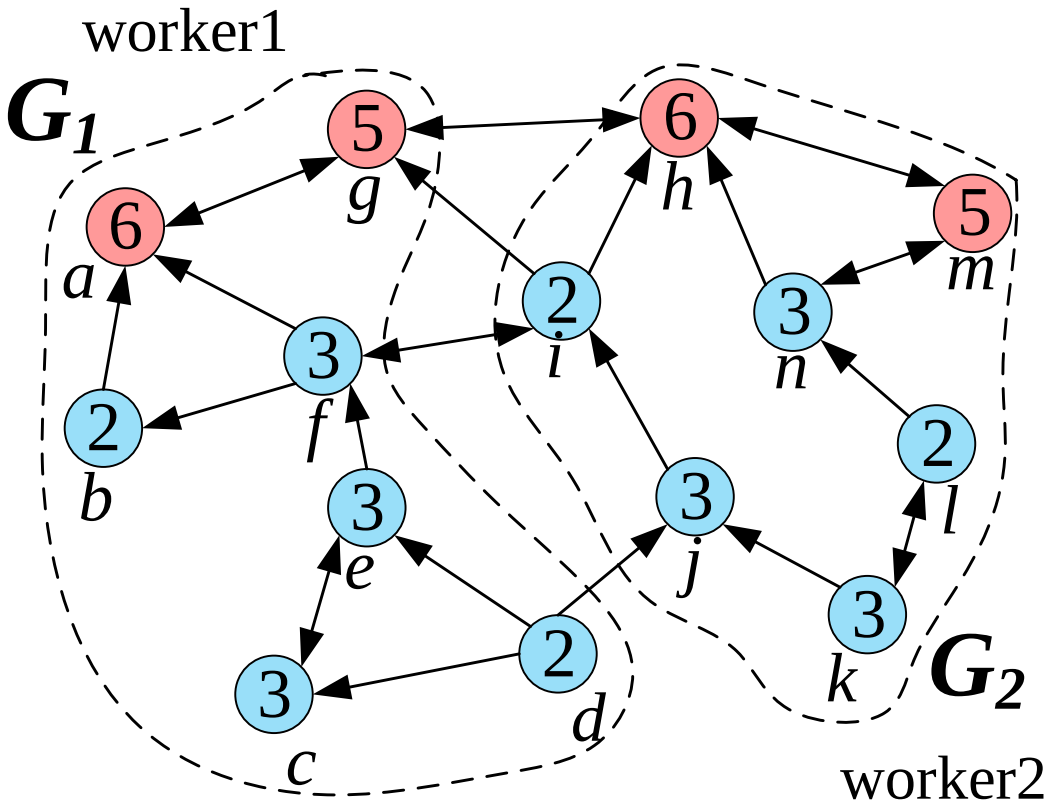
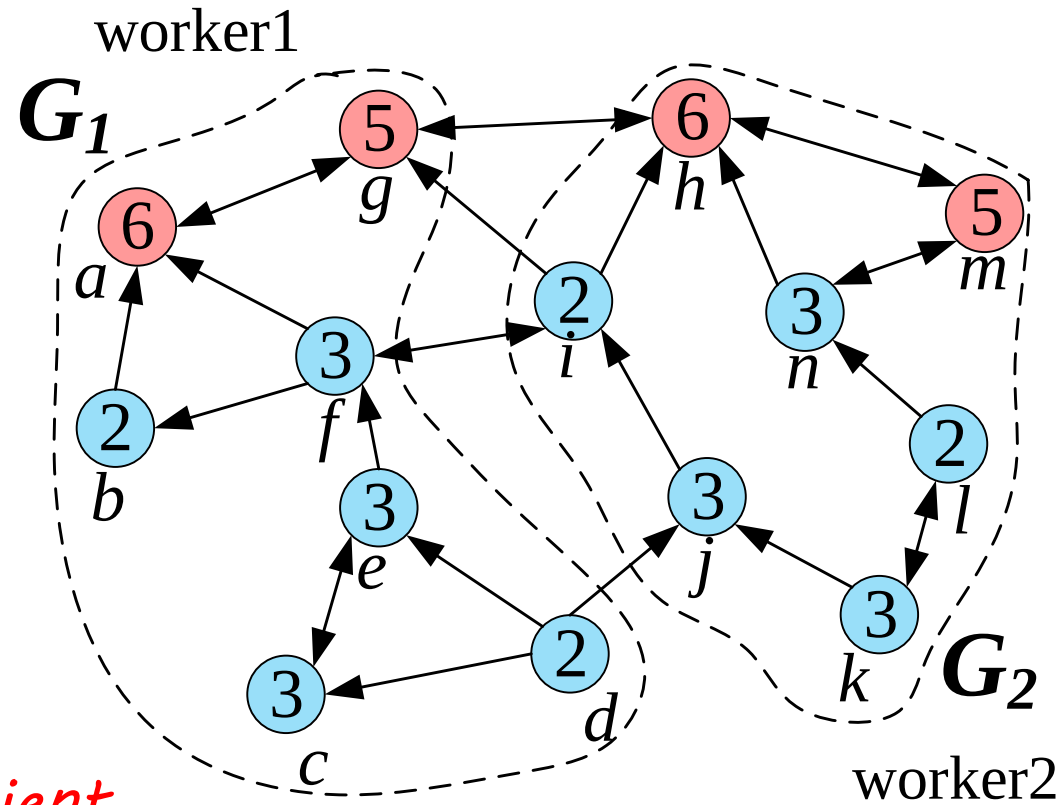


Fig. 2: per-bin hotness balance

Per-Bin Hotness Balanced Partition

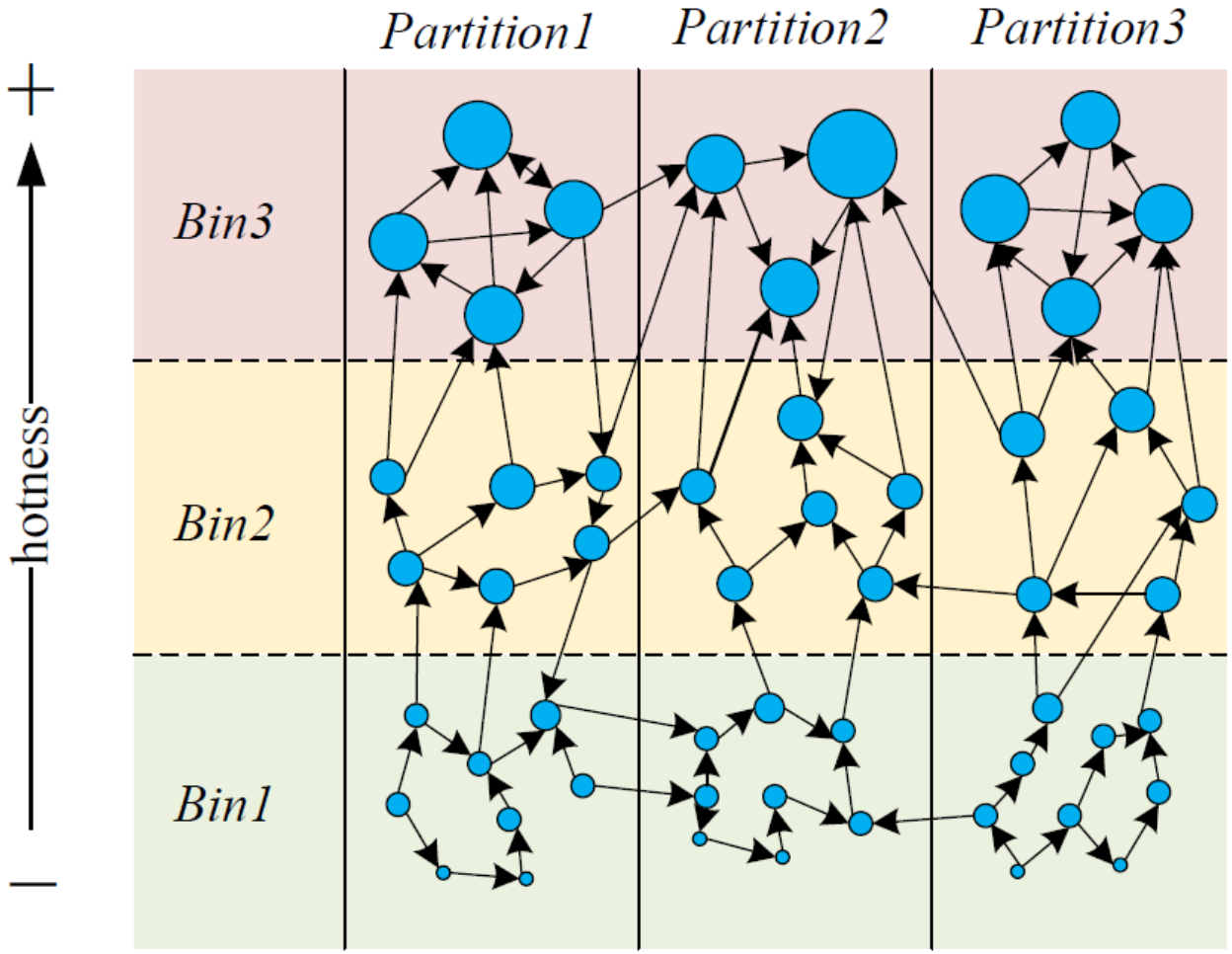
The hotness of each partition is balanced and the distribution of hotness values of each partition is the same as that of the original graph.



Both worker1 and worker2 are efficient

per-bin hotness balance

Per-Bin Hotness Balanced Partition



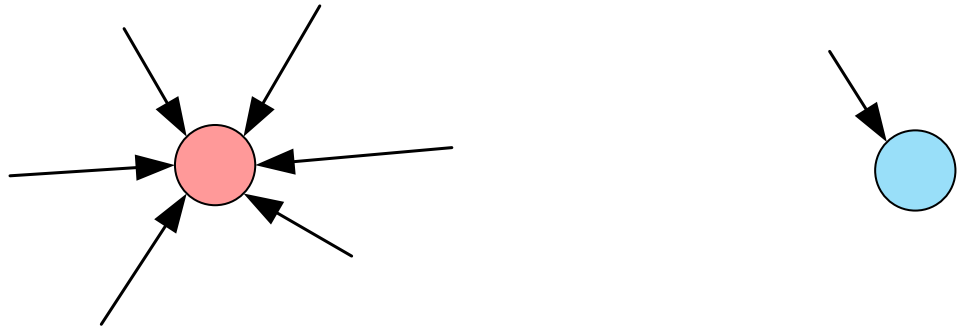
Partition each bin into k hotness balanced partitions.

Maximize the communication between bins in the same partitions

Minimize the communication between bins in different partitions.

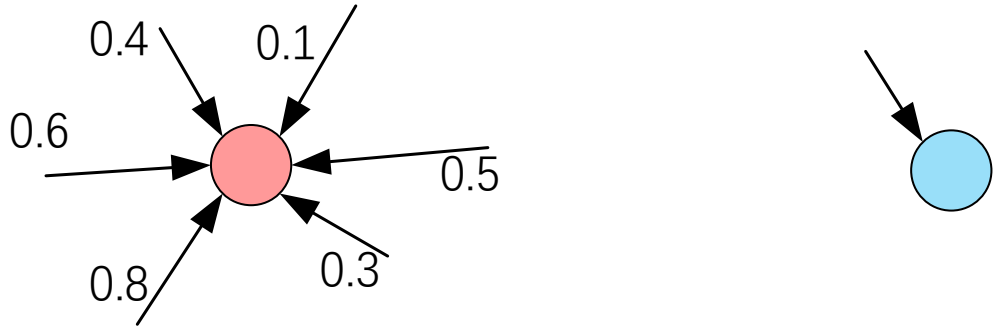
An illustration of Per-Bin hotness balanced partition

Hotness Estimation



The vertex with high indegrees always has higher execution priority and is likely to be hot.

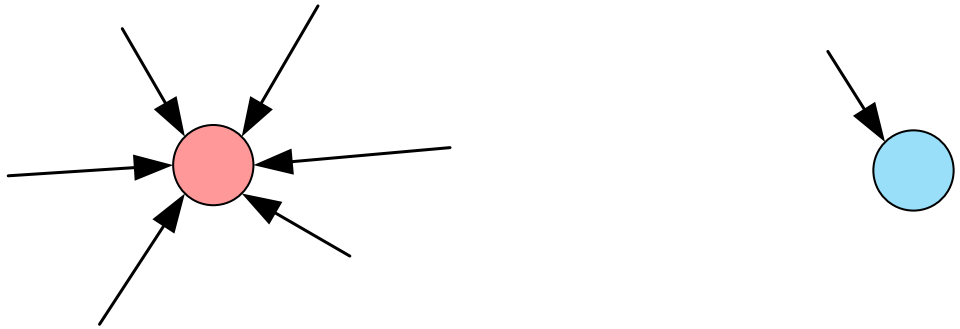
Hotness Estimation



The vertex with high indegrees always has higher execution priority and is likely to be hot.

Weights of edges may affect the hotness value of target vertices.

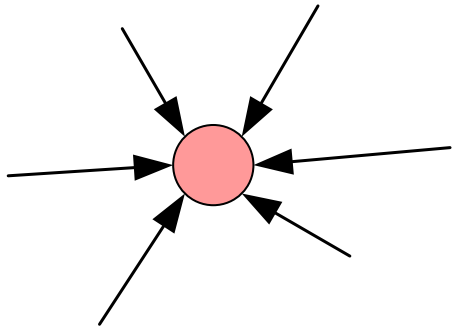
Hotness Estimation

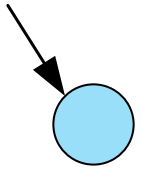


$$h_v = \sum_{u \in IN(v)} \frac{w_{u,v}}{\sum_{w \in OUT(u)} w_{u,w}}$$

the hotness of vertex v

Hotness Estimation





$$h_v = \sum_{u \in IN(v)} \frac{w_{u,v}}{\sum_{w \in OUT(u)} w_{u,w}}$$

the hotness of vertex v

$$com_{u,v} = hu$$

the communication of edge $e_{u,v}$

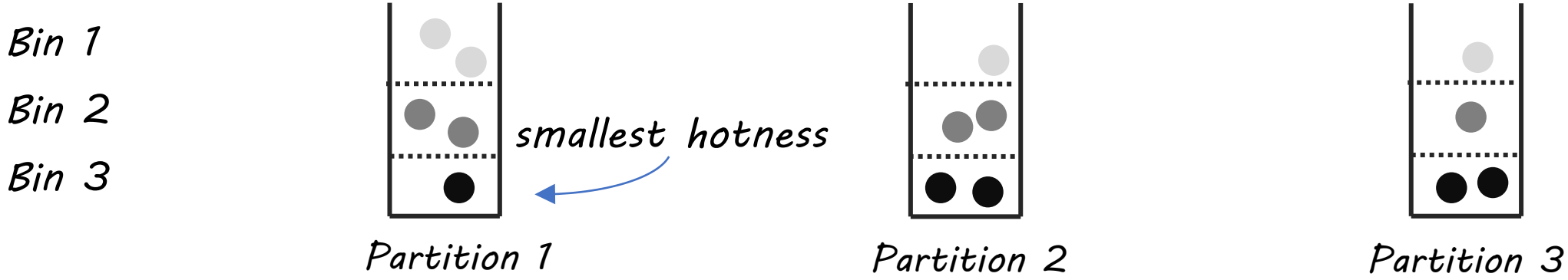
Streamed Per-Bin Hotness Balanced Partition

We assign each vertex to the partition with low hotness on the corresponding bin and low communication cost (edge information is omitted).

belong to bin3

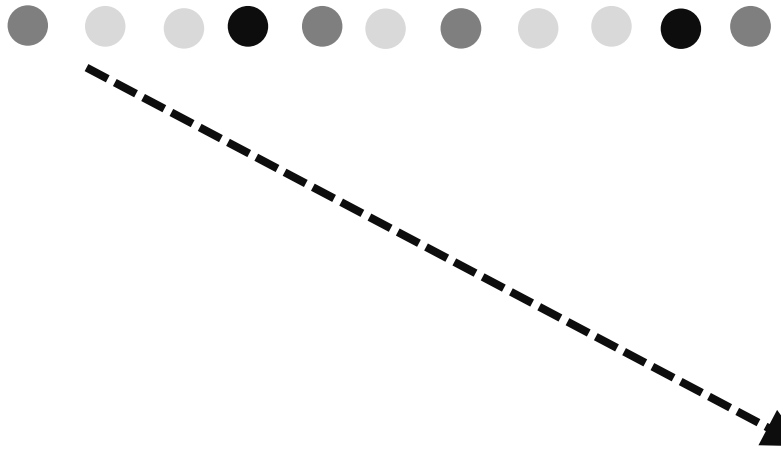
the hotness values are separated into 3 bins

The first partition's third bin has smallest hotness, so this vertex is assigned into the first partition.



Streamed Per-Bin Hotness Balanced Partition

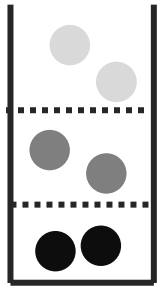
the hotness values are separated into 3 bins



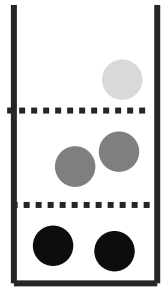
Bin 1

Bin 2

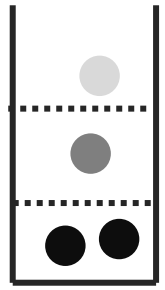
Bin 3



Partition 1



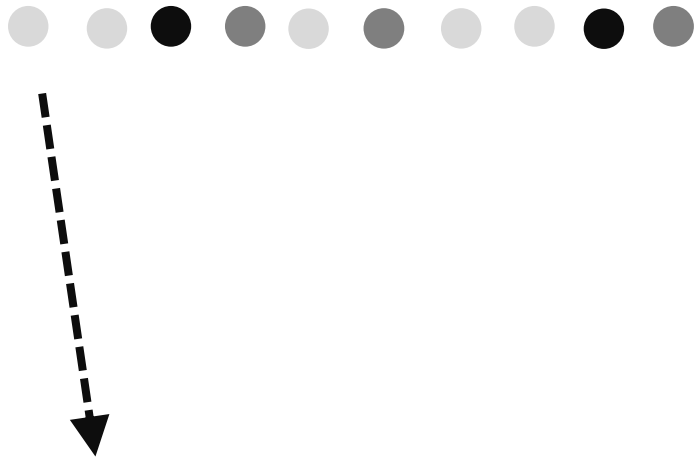
Partition 2



Partition 3

Streamed Per-Bin Hotness Balanced Partition

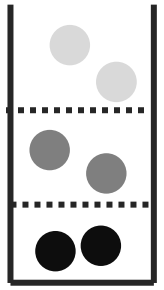
the hotness values are separated into 3 bins



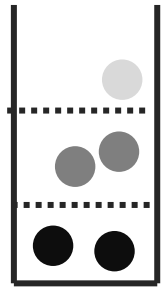
Bin 1

Bin 2

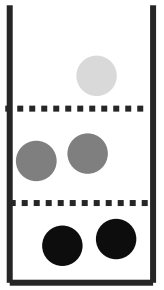
Bin 3



Partition 1



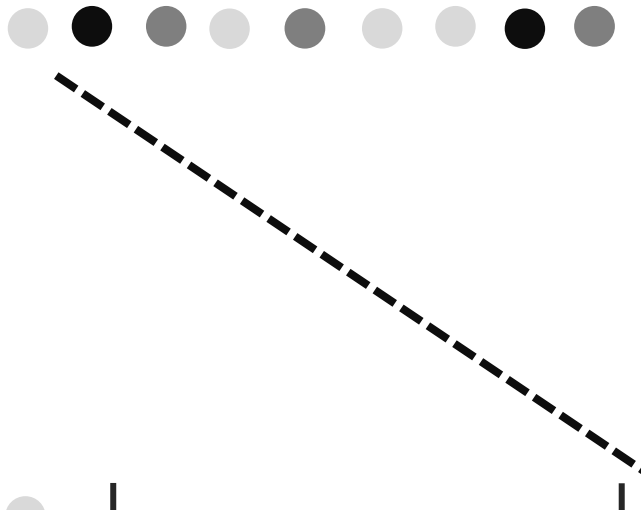
Partition 2



Partition 3

Streamed Per-Bin Hotness Balanced Partition

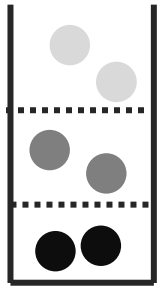
the hotness values are separated into 3 bins



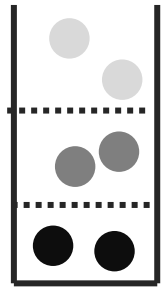
Bin 1

Bin 2

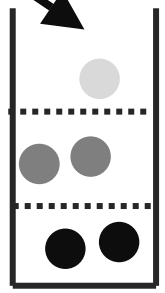
Bin 3



Partition 1



Partition 2



Partition 3

Streamed Per-Bin Hotness Balanced Partition

$$i = \underset{i:\{\boldsymbol{h}_{ji} \leq \tau \frac{\sum_{v \in \mathcal{H}_j} h_v}{k}\}}{\arg\min} \alpha \cdot \left((\boldsymbol{h}_{ji} + h_v)^\gamma - \boldsymbol{h}_{ji}^\gamma \right) + (1-\alpha) \cdot \left(\sum_{u \notin V_i} com_{u,v} + \sum_{w \notin V_i} com_{v,w} \right)$$

where $0 \leq \alpha \leq 1$, $\tau > 1, \gamma > 1$, and h_{ji} is the sum of hotness values of i-th partition in the j-th bin.

Streamed Per-Bin Hotness Balanced Partition

imbalance cost

$$i = \underset{i:\{\boldsymbol{h}_{ji} \le \tau \frac{\sum_{v \in \mathcal{H}_j} h_v}{k}\}}{\arg\min} \alpha \cdot \left((\boldsymbol{h}_{ji} + h_v)^\gamma - \boldsymbol{h}_{ji}^\gamma \right) + (1-\alpha) \cdot \left(\sum_{u \notin V_i} com_{u,v} + \sum_{w \notin V_i} com_{v,w} \right)$$

Communication cost

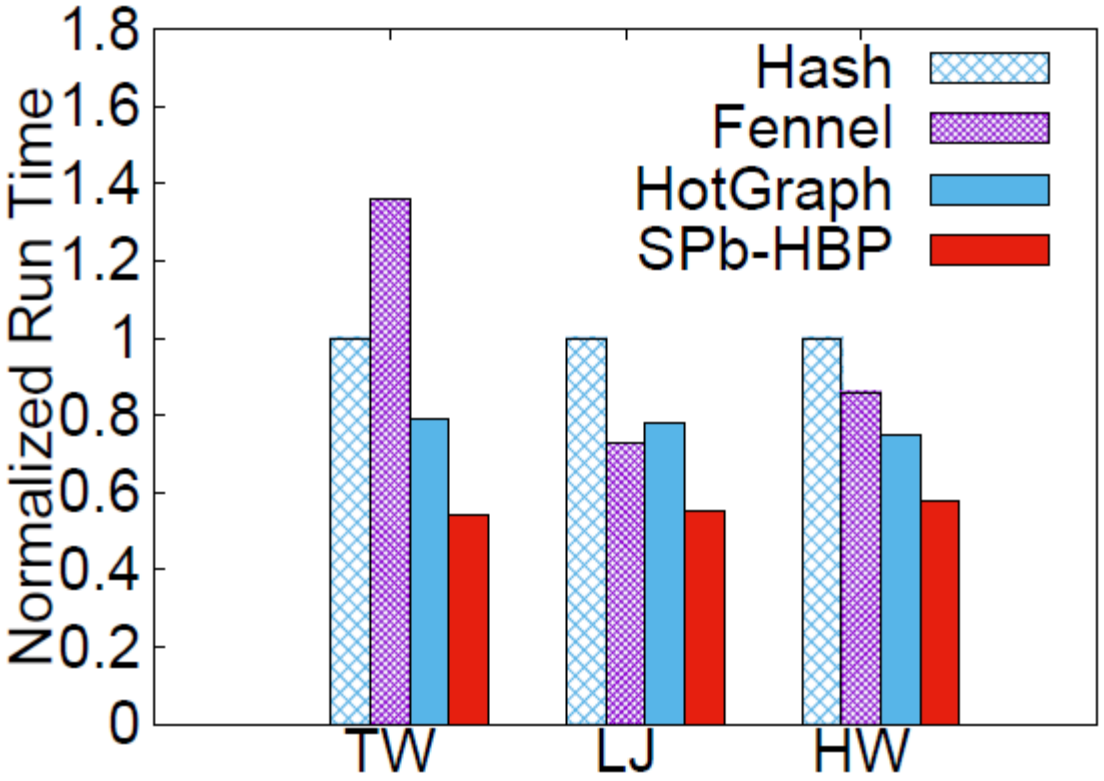
where $0 \le \alpha \le 1$, $\tau > 1, \gamma > 1$, and h_{ji} is the sum of hotness values of i-th partition in the j-th bin.

Experiment: Preparation

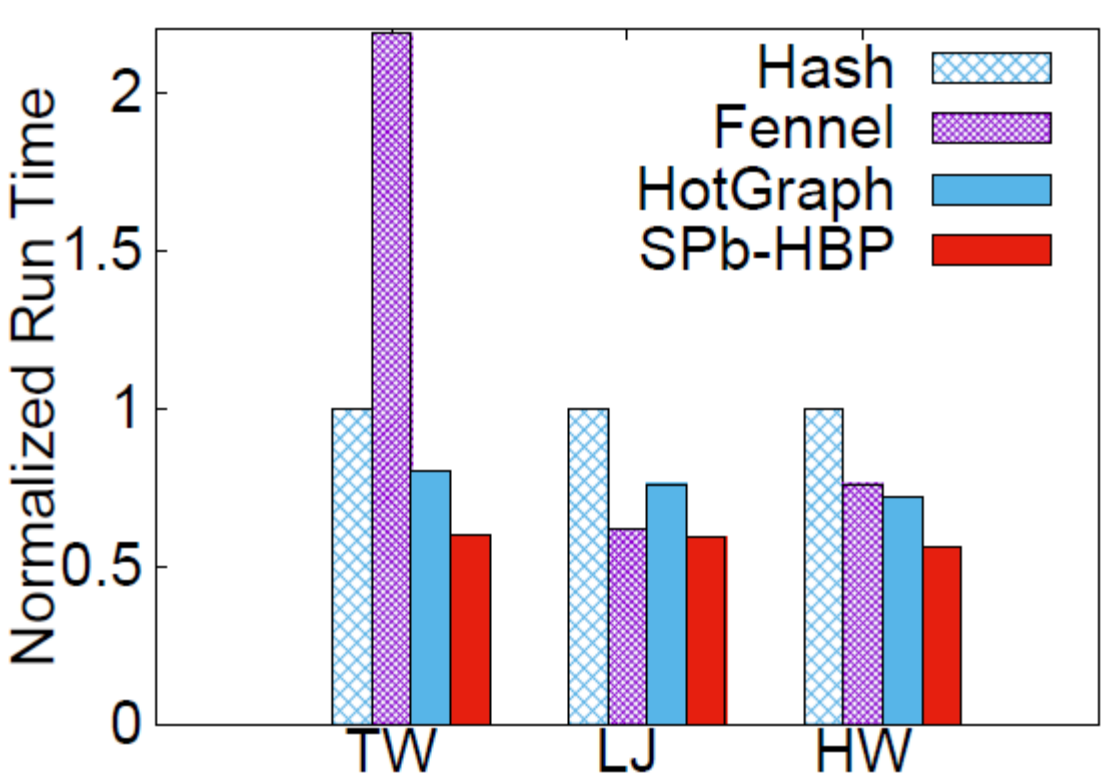
Platform	Alibaba Cloud
System	Maiter
Cluster	1 master (ecs.cs.large)
	4 slaves (ecs.cs.large)
Algorithm	PageRank, PHP
Competitors	Hash, Fennel[1], HotGraph[2]
Data sets	Twitter (TW), LiveJournal (LJ) Hollywood (HW)

1. Tsourakakis, Charalampos, et al. "Fennel: Streaming graph partitioning for massive scale graphs." Proceedings of the WSDM 2014.
2. Zhang, Yu, et al. "HotGraph: Efficient asynchronous processing for real-world graphs." *IEEE TOC 2016*.

Experiments: Runtime Comparison



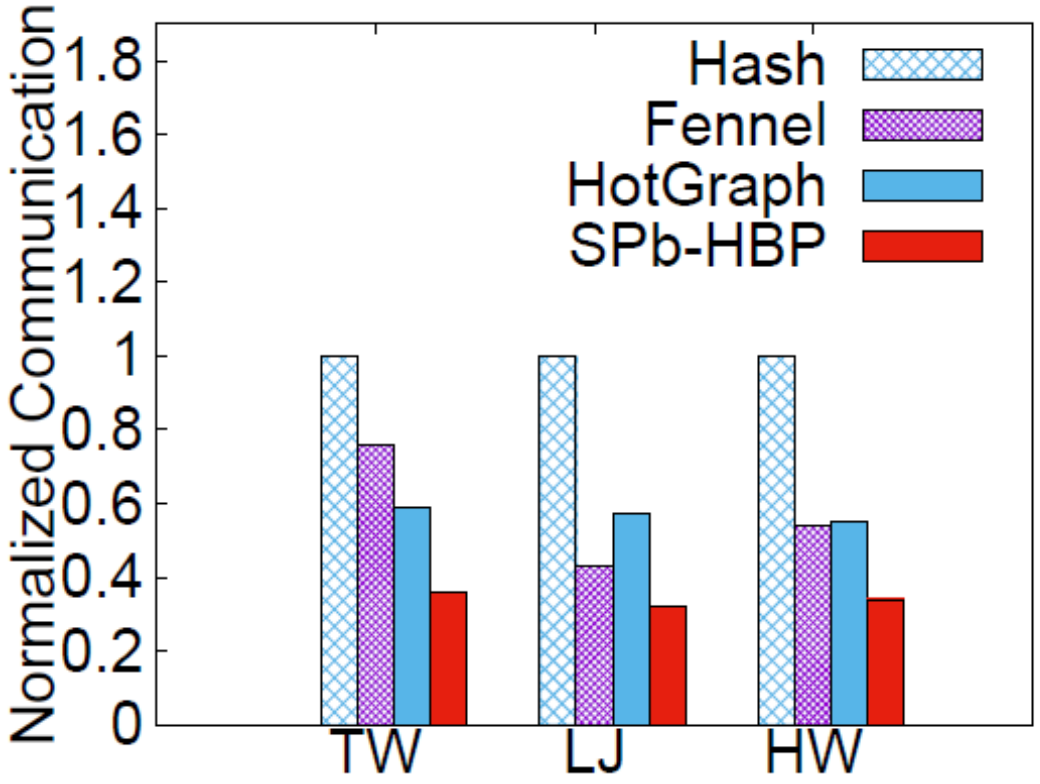
PageRank



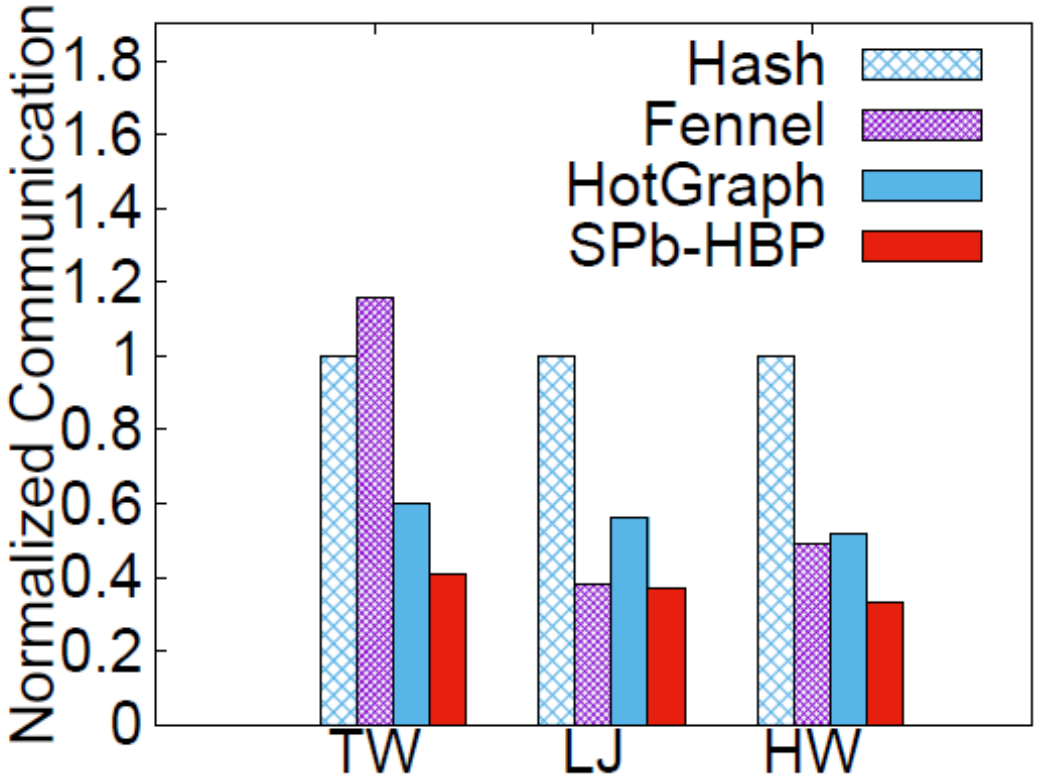
PHP

SPb-HBP is our stream-based per-bin hotness balanced partition

Experiments: Communication Cost Comparison

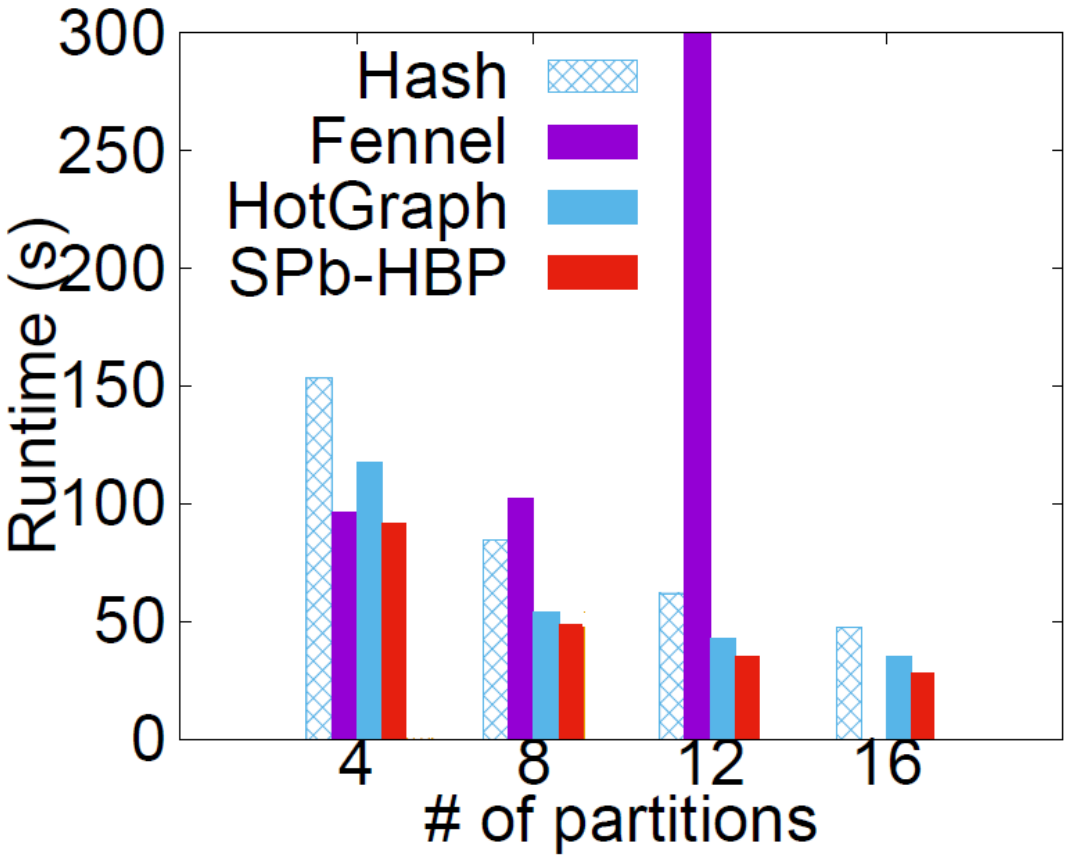


PageRank

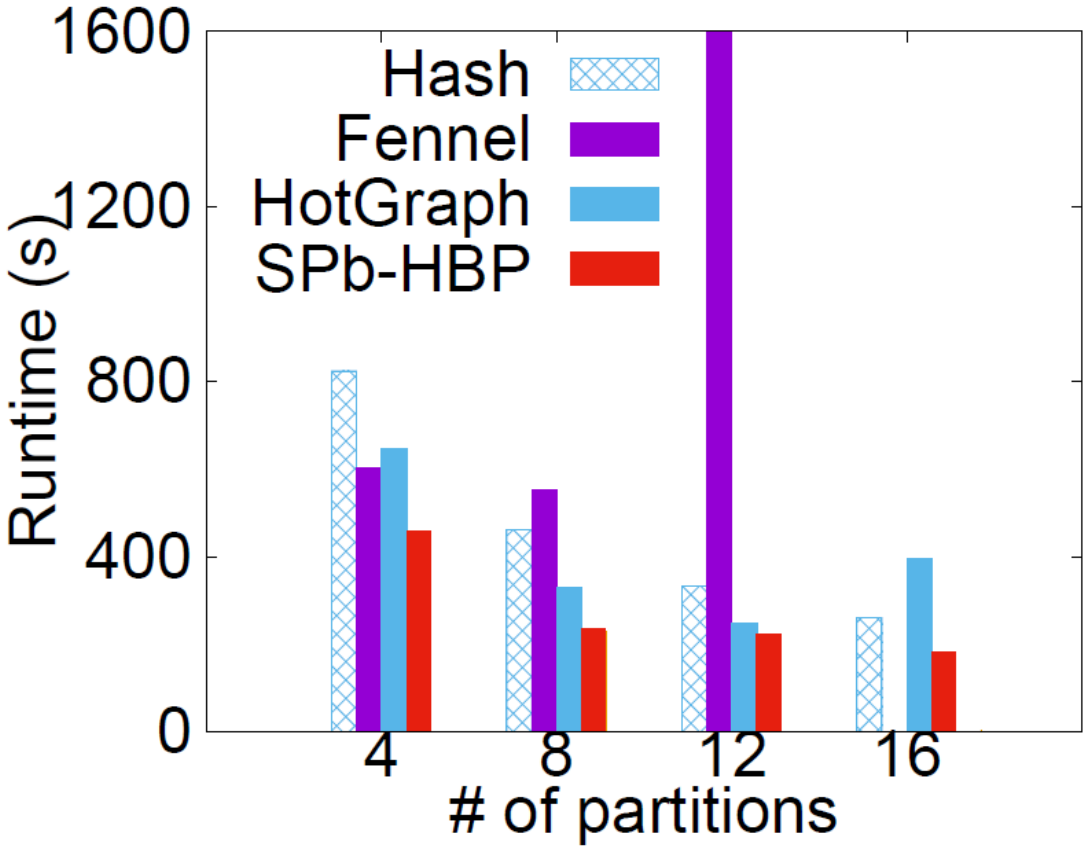


PHP

Experiments: Scalability

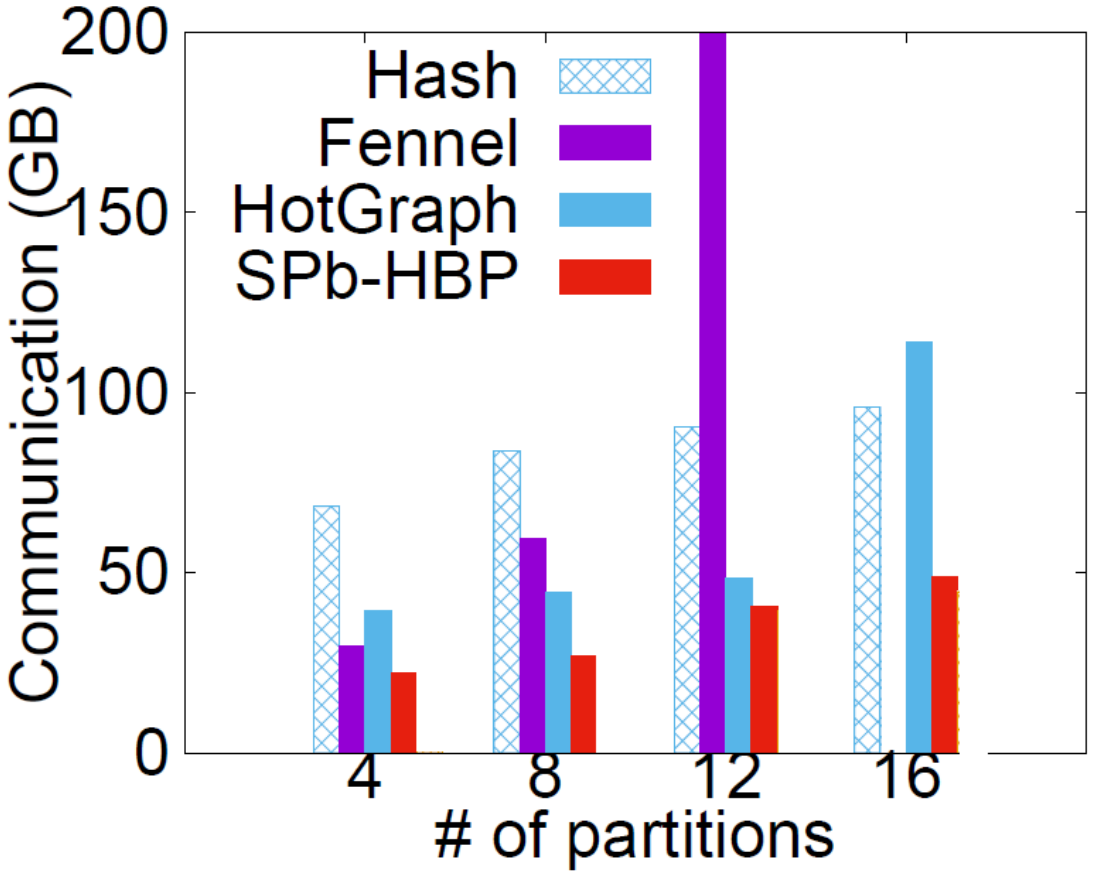


PageRank

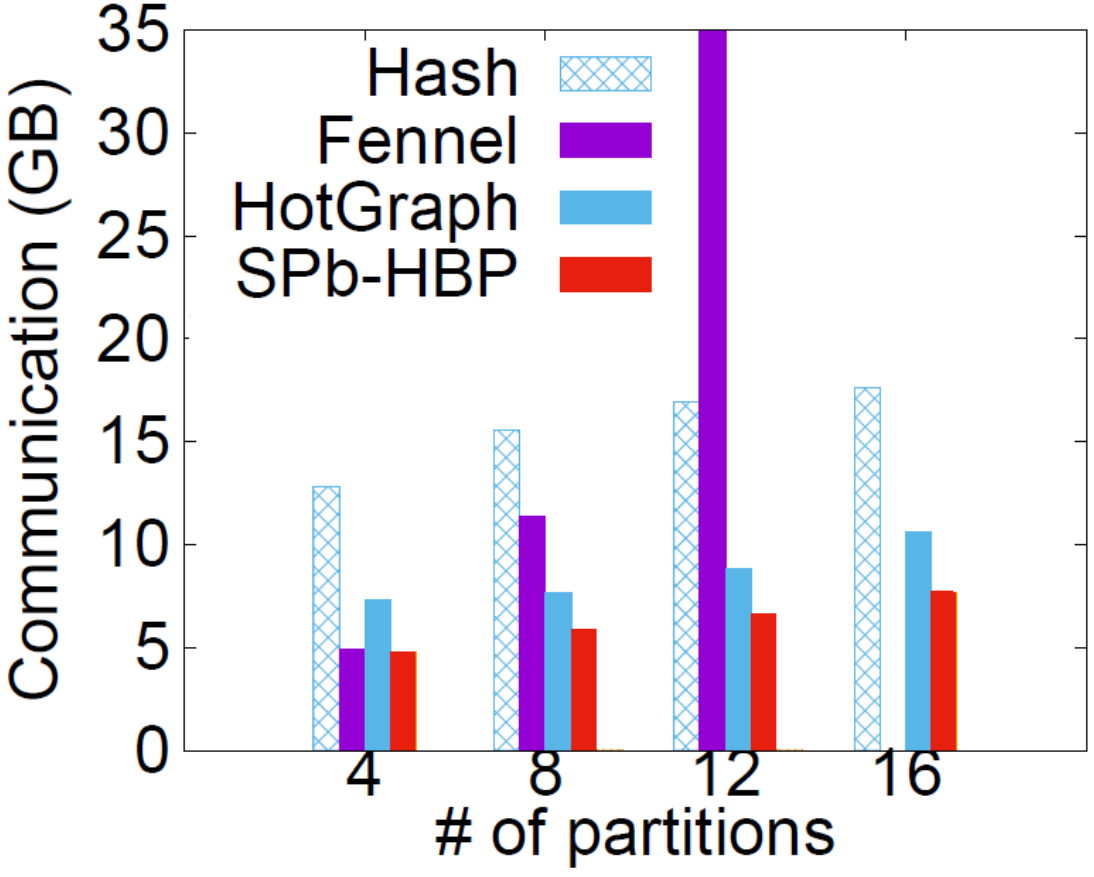


PHP

Experiments: Scalability



PageRank



PHP

Conclusion

- The analysis of the existing graph partition methods finds that they are not suitable for asynchronous graph processing systems.
- A new graph partition idea, hotness balanced partition, is proposed.
- A stream-based per-bin hotness balanced partition algorithm is proposed.

Thanks

Any questions please email
gongsf@stumail·neu·edu·cn