Distributed Computing

16. GraphX

Graph-Based Analytics

- When MapReduce showed up, several engines for efficient graph-based analytics were introduced
  - Many machine learning & network analysis applications
  - Exploit the same cluster, but have ad-hoc optimizations
- GraphX brings the same concepts to Spark, in a quite small package
  - First version: 2,500 lines of code
- Reference: paper at USENIX OSDI ‘14

Example: PageRank

- Evaluate **how important a node is in a graph** (centrality)
- Can we see which page is more important in the Web graph (pages and links)?
- Idea: a **random walker** starts from a random page and
  - With probability d=0.85, clicks a link at random
  - With probability (1-d) 0.15, goes to another random page
- The score of a page is the **probability the random walker gets there**

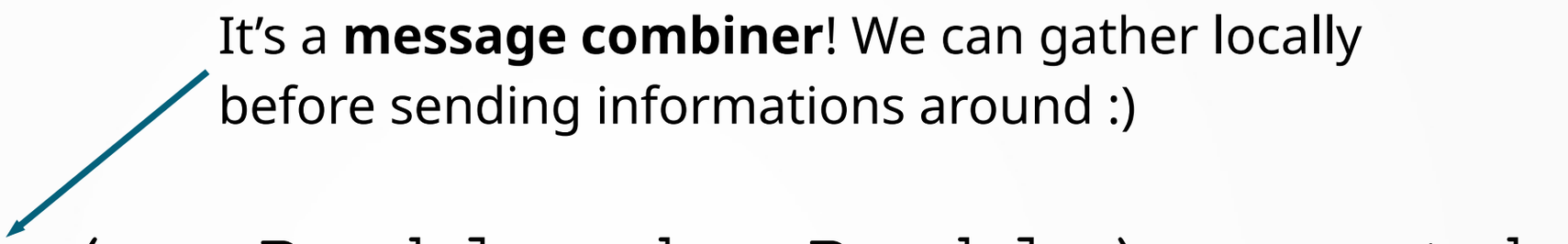
Single-Machine PageRank

- `rank = [1/n, 1/n, … ]  # array of length n`
- `For m iterations:`
  - `new_rank = [(1-d)/n, (1-d)/n, …]  # length n`
  - `For each node x with t neighbors`
    - `For each neighbor y of x`
      - `new_rank[y] += d * rank[x] / t`
  - `rank = new_rank`

Pregel: Think Like a Vertex

```
def PageRank(v: Id, msgs: List[Double]) {
  // Compute the message sum
  var msgSum = 0
  for (m <- msgs) { msgSum += m }
  // Update the PageRank
  PR(v) = 0.15 + 0.85 * msgSum
  // Broadcast messages with new PR
  for (j <- OutNbrs(v)) {
    msg = PR(v) / NumLinks(v)
    send_msg(to=j, msg)
  }
  // Check for termination
  if (converged(PR(v))) voteToHalt(v)
}
```

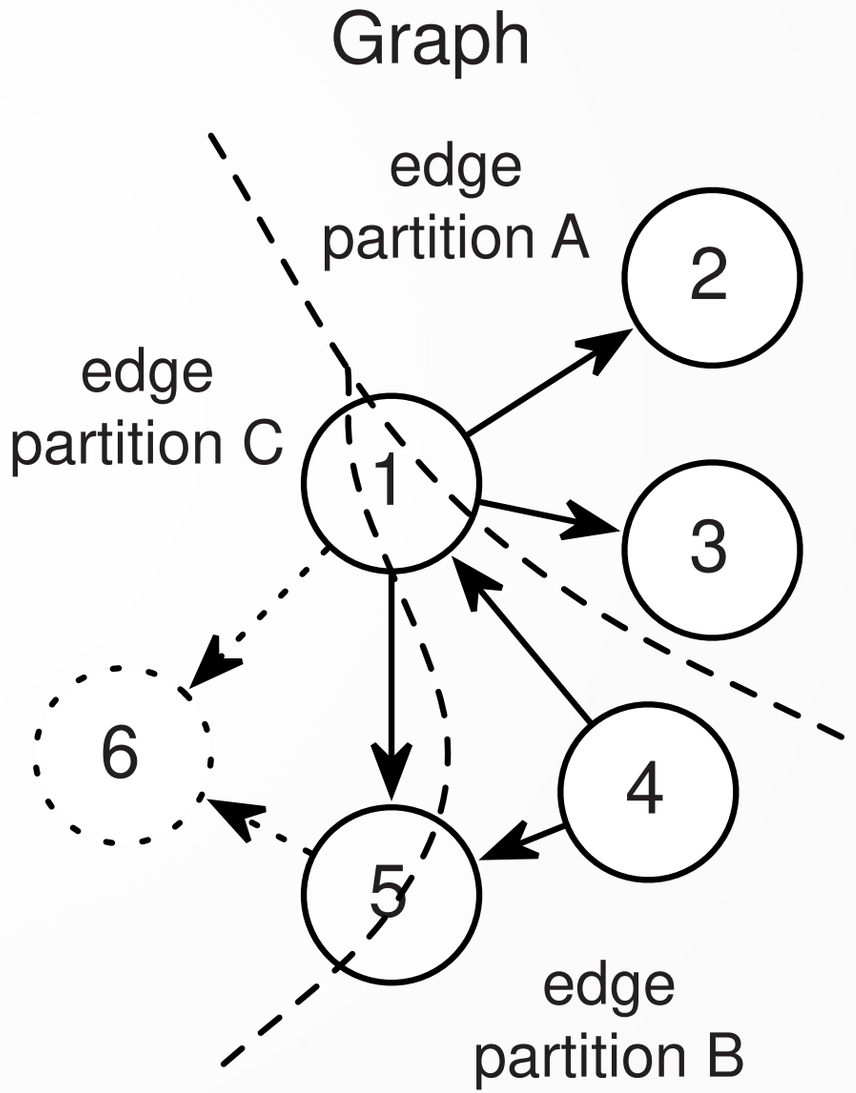
GAS: Gather, Apply, Scatter




```
def Gather(a: Double, b: Double) = a + b
def Apply(v, msgSum) {
  PR(v) = 0.15 + 0.85 * msgSum
  if (converged(PR(v))) voteToHalt(v)
}
def Scatter(v, j) = PR(v) / NumLinks(v)
```

Partitioning

- The largest thing in a graph is **the edge list**
- Hence, divide **by edges**
  - Some nodes will be "replicated" in different partitions (**mirror nodes**)
  - Partition to minimize the number of them
  - Copying a node's data (e.g., their PageRank value) is cheaper than sending messages along edges in different machines



Inactive Nodes

- In many iterative algorithms some nodes “turn off”
- E.g., a node already reached convergence
- We can mark that to stop sending updates to them and accelerate iterations