Lecture 13:

Gossip Algorithms

Rik Sarkar

Gossip

- People talk to other people : information spreads
- Everyone has a piece of information
- Each node talks to another node at random
- Can we find an aggregate using this sort of information propagation?
  - Aggregates like sum, avg..

Gossip

- Advantages:
  - Fully distributed. No sink.
  - Fault tolerant : failure of many nodes do not affect the protocol
  - Can adapt to changing values : keep gossiping
- Issues:
  - Iterative method, converges to true value, never quite gets to the value
  - May take time to converge
  - May cost communication

Paper

Gossip-Based Computation of Aggregate Information, David Kempe, Alin Dobra, Johannes Gehrke, *FOCS 03*

A basic protocol: push sum

- Each node starts with a sum s_i and a weight w_i
- Initially, s_i = value at I and w_i= 1 (contribution from one value)
- At each round, send (s_i/2, w_i/2) to a random node, and set its own values also to (s_i/2, w_i/2)
- So, at each round, node I sends half of its value to someone

A basic protocol: push sum

- At each round, it also gets values and weights from some other nodes

Algorithm 1 Protocol Push-Sum

1: Let $\{(\hat{s}_r, \hat{w}_r)\}$ be all pairs sent to i in round $t-1$
2: Let $s_{t,i} := \sum_r \hat{s}_r$, $w_{t,i} := \sum_r \hat{w}_r$
3: Choose a target $f_t(i)$ uniformly at random
4: Send the pair $(\frac{1}{2}s_{t,i}, \frac{1}{2}w_{t,i})$ to $f_t(i)$ and i (yourself)
5: $\frac{s_{t,i}}{w_{t,i}}$ is the estimate of the average in step t

- Hope:
  - Mass conservation: sum of all s_i is always total sum, sum of all w_i is always n
  - Eventually, everyone will have 1/n fraction of everyone else's value
  - Eventually, everyone will have 1/n fraction of everyone else's weight
  - So, value will be the average, weight will be = 1
- To compute sum:
  - One node (query node) has w=1, everyone else has w = 0

Diffusion

- The value at each node is diffusing in the network until it is uniformly distributed
- Uniform distribution is a steady state
- How fast are the values diffusing? How long will it take for all the values to be at almost uniform distribution?

Contribution vectors

- Vector $V_{t,i}$: the fractional contribution of each other node to the weight at i
- So, $V_{t,i,j}$: Contribution of j to the weight at i after round t

- Error at a node:

$$\Delta_{i,t} = \max_j \left| \frac{v_{t,i,j}}{\|\mathbf{v}_{t,i}\|_1} - \frac{1}{n} \right|$$

w_i

How fast does the protocol converge?

- Think of a potential function:

$$\Phi_t = \sum_{i,j} (v_{t,i,j} - \frac{w_{t,i}}{n})^2$$

- We show that this potential drops *fast*

- This potential reduces by at least half in each round.

Lemma 2.3 *The conditional expectation of* Φ_{t+1} *is* $\mathrm{E}[\Phi_{t+1} \mid \Phi_t = \phi] = (\frac{1}{2} - \frac{1}{2n})\phi$.

- How many rounds will it take to drop to ε ?

Number of rounds

- Since the initial potential can be at most n
- To get Φ_t to some constant value will require O(log n) round

- To get the value to below ε will require O(log $\frac{1}{\varepsilon}$) rounds
- Thus a total of O(log n + log $\frac{1}{\varepsilon}$) rounds in expectation

- We want this to happen with a good probability at least 1- δ
- Rounds: O(log n + log $\frac{1}{\varepsilon}$ + log $\frac{1}{\delta}$)

How fault tolerant is it?

- Suppose each message ha a chance of failing, how fast does it converge?

Theorem 2.4 *If $\mu < 1$ is an upper bound on the probability of message loss in each round resp. the fraction of failed nodes, then the diffusion speed T' in the presence of failures satisfies $T'(\delta, n, \varepsilon) \leq \frac{2}{(1-\mu)^2} T(\delta, n, \varepsilon)$.*

Summary

- Completely decentralized
- Excellent fault tolerance
- Iterative & probabilistic method, not perfect
- Fast in some scenarios, may be slow in others
- Good for computing sums
- Can be extended to compute many types of linear functions, random sampling, quantiles etc