Announcements (Jan. 25)

- Sign up for the first round of presentations
  - Please reply by Sunday
  - First one on sensor network applications on Feb. 20
- Course project handout distributed today
  - Milestone 1: form teams and schedule an appointment with me by March 1 (the earlier the better)
- Readings for next Tuesday: Monitoring of Extreme Values by Silberstein et al. and Contour Map by Xue et al.
  - One review due (you choose which)

Computing aggregates

- SQL aggregates: MIN, MAX, SUM, COUNT, AVG
- More complex: COUNT(DISTINCT ...), median/quantiles, wavelets, samples, ...
- An aggregate function can be implemented with three functions:
  - Generate, $G(x)$: produce a partial state record from input
  - Fuse, $F(r_1, r_2)$: merge two records into one
  - Evaluate, $E(r)$: evaluate result from a partial state record
- E.g., for AVG:
  - $G(x) = \frac{x}{1}$
  - $F\left(\frac{x}{1}, \frac{y}{w}\right) = \frac{x+y}{w+x}$
  - $E\left(\frac{x}{1}\right) = \frac{x}{w}$

Making a tree more robust

- Tree is pretty fragile
  - If one link fails, data from entire subtree is lost
- Turn tree into a DAG?
  - Send $1/k$ of the summary to $k$ parents, for free (broadcast)
  - One link failure only drops $1/k$ data
Aggregation + routing spaghetti

- Variation of the DAG idea: send the whole summary up to $k$ parents?
  - Works for some aggregates (which?)
  - But in general, one item can be counted many times!
- Aggregation scheme is too dependent on routing!
  - Routing tweaks affect correctness of aggregation
  - Can we decouple them?

Order and duplicate insensitivity (ODI)

- Won’t it be nice if aggregation scheme is insensitive to the sequence or duplication of inputs?
  - More precisely, a scheme is ODI-correct if, for any DAG, it produces a result identical to the correct answer produced by a canonical tree
  - This is the property that made MIN/MAX easy

Testing ODI-correctness

- Necessary and sufficient test turns out to be really simple
  - $G(.)$ preserves duplicates; i.e., if $x_1$ and $x_2$ are considered duplicates, then $G(x_1) = G(x_2)$
  - $F(.,.)$ is commutative
  - $F(.,.)$ is associative
  - $F(.,.)$ is same-input idempotent; i.e., $F(r, r) = r$

  - Do MIN/MAX work?
  - Does COUNT work out-of-box?

How to design ODI-correct schemes?

- Let’s do COUNT as an example
  - A little randomness/approximation goes a long way
    - Use synopses—compact, approximate summaries of data—for partial state records
    - Borrow the “almighty” FM-sketch
    - Then turn COUNT into MAX, which is ODI-correct

FM-sketch

- Flajolet and Martin, 1985
- Counts # of distinct elements in a multi-set in one pass
  - Powerful building block for many data stream algorithms
- Start with a bitmap of 0’s
- For each element $x$ in the multi-set, hash it to a positive integer using function $h(x)$
- Turn the $h(x)$-th bit on
  - # of distinct elements $\approx 2^{(\text{position of first 0} - 1) / 0.77351}$
  - Use multiple independent $h$’s to improve accuracy
    - With enough number of $h$’s, can get within a prescribed error with probability higher than a prescribed threshold

FM-sketch (cont’d)

- For FM-sketch to work, need
  - $Pr[h(x) = 1] = 1/2$, $Pr[h(x) = 2] = 1/4$, $Pr[h(x) = 3] = 1/8$, …
  - Easy to simulate with a random binary hash $g(x,i)$
- Intuition: the $i$-th bit will be 1 if there are many more than $2^i$ distinct elements, each trying to set the bit with probability 1/2

Safely 1’s for positions $< \log m$
Safely 0’s for positions $\geq \log m$

Expected position of the first 0 roughly estimates $\log m$
3

Back to COUNT...

- Suppose each node has a unique $id$
- Partial state record: FM sketch with $> \log n$ bits
- $G(id)$: generate FM-sketch with $\{id\}$
- $F(s_1, s_2)$: bitwise-OR the two input sketches
  - OR is like MAX
- $E(s)$: estimate using the position of first 0 in $s$

- How about SUM?
  - Convert to COUNT: for node $id$ with integer value $v$, generate $v$ items $(id, 1), (id, 2), \ldots, (id, v)$

Rings

- Now we can use much more flexible routing structures to help improve communication reliability without double-counting

Snooping tricks

- Implicit acknowledgement
  - Explicit ack too expensive for sensor networks
  - Node $u$ sending to $v$ snoops subsequent transmissions from $v$ to see if $v$ indeed forwards the message for $u$
    - $Why$ doesn't this trick work for TAG SUM?
    - $How$ does it work with synopsis?
- Suppression
  - If my neighbor's transmission subsumes mine, no need to transmit mine
    - Used in TAG
    - $Would$ this trick work in synopsis diffusion?

Another example: uniform sample

- Suppose each node has a unique $id$
- $G(id, v) = \{(id, v, r)\}; r$ is randomly chosen from $[0, 1]$
- Partial state record: a set of no more than $K$ entries of the form $(id, v, r)$
- $F(s_1, s_2)$: up to $K$ distinct entries in $s_1 \cup s_2$ with largest $r$
  - Again, top-$K$ is a simple extension of MAX
- $E(s)$: output all $(id, v)$ entries
  - A random sample because $r$'s are randomly generated

Sensor aggregation problem: solved?

- How large are synopses?
- What are the costs of complex local processing?
- Is snooping completely free?
- MAX is not robust against outliers
  - What if somebody injects an all-1 FM-sketch?
- Everybody still transmits!
  - Can we do better?
- Are we taking advantage of spatio-temporal correlations?
- Can suppression and redundancy really mix?