Brief Announcement: Practical Summation via Gossip

Wesley W. Terpstra Christof Leng* Alejandro P. Buchmann

Technische Universitat Darmstadt
D-64283 Darmstadt, Germany
{terpstra,cleng,buchmann}@dvs1.informatik.tu-darmstadt.de

ABSTRACT

Distributed summation is computed in asymptotically minimal rounds by Kempe et al's Push-Sum algorithm. Unfortunately it has minor problems in practise, resolved here.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms

Algorithms, Experimentation, Performance

Keywords

Peer-to-Peer, Gossip, Sums, Simulation

1. INTRODUCTION

Although peer-to-peer systems strive to interact solely with a few neighbours, there are many useful metrics that capture global state. Examples include network size, churn rate, and average bandwidth. We are primarily concerned with computing sums across all peers in an expander graph (many peer-to-peer networks). Approaches to computing aggregate functions efficiently and in a fully decentralized manner are a hot research topic [2–5]. Competing solutions can be compared by convergence speed and message count.

Computing the maximum (or minimum) is relatively simple and well-understood: each peer gossips about the largest value it currently knows. Whenever a neighbour informs a peer of a larger value, the peer updates its value. Averages can be computed with so-called mass conservation, explained well in [3]. Kempe et al's Push-Sum is asymptotically optimal with respect to the convergence speed. While sums are slightly harder, both recent approaches from PODC 2006 [4, 5] do not build on the optimal mass conservation, but start anew, with sub-optimal convergence speed.

There are valid concerns about the Push-Sum algorithm raised in [5], but it is not the case that the Push-Sum algorithm requires a complete graph. It also operates over expander graphs [3]. The main Push-Sum problem we resolve here is that sum computation and termination co-ordination require a designated leader. The remaining problem, that Push-Sum as stated is synchronous, is only a problem with its proof. Kempe conjectures synchrony is unnecessary in practise, a claim we validate here by simulation. Similar asynchronous algorithms [2] have been shown correct.

*Supported by the DFG Research Group 733, Quality in Peer-to-Peer Systems (QuaP2P).

PODC'07, August 12–15, 2007, Portland, Oregon, USA.
ACM 978-1-59593-616-5/07/0008.

Listing 1: The Push-Sum Algorithm

```
currentEstimate =
  my_water / my_fish;
Periodically:
  my_fish /= my_degree+1;
  my_water /= my_degree+1;
  sendToAllNeighbours(my_fish, my_water);
Receive(fish, water):
  my_fish  += fish;
  my_water += water;
```

2. THE PUSH-SUM ALGORITHM

We explain the intuition behind the Push-Sum algorithm by analogy. By letting loose a school of fish into a lake, and assuming they spread out uniformly, one can measure the lake's volume by counting the fish in a cubic metre of water. To compute a sum, every peer initially manages some lake region (its volume is the variable to sum). When gossipping, peer v mixes its lake region with its neighbours and itself; everyone gets $\frac{1}{\deg(v)+1}$ of this water and fish. Thus the fish swim/diffuse throughout the lake until they are uniformly distributed. Listing 1 implements this asynchronously.

Kempe et al proved that for an expander graph with n peers, after $O(\log n + \log \frac{1}{\epsilon})$ rounds every peer's ratio of water to fish is within ϵ relative error of the sum. In our opinion, their key insight was to introduce a unit of measure (the fish) which also diffuses throughout the network. This allows some peers to possess more water and fish (due to, for example, higher degree), yet still compute the same ratio.

3. MAKING IT WORK IN PRACTISE

The main problem with Push-Sum is the need for a designated leader. One of the leader's responsibilities is to release the school of fish into the lake. We avoid this problem. Every peer releases 1.0 fish with distinct fish size; a unique size can be a random number or the IP:Port of the peer. Bigger fish eat smaller fish (just as larger numbers dominate when computing a maximum) until only the largest fish remains. Thus the steady state contains a total of exactly 1.0 fish.

The other responsibility of a Push-Sum leader is deciding when to finalize a measurement. For most uses of aggregate statistics, one wants to recompute the statistic at regular intervals. Therefore, we include a cyclically rotating 16-bit measurement identifier. Whenever a peer receives a message pertaining to a new measurement, it finalizes its current estimate and initializes a new value. Any peer can decide to finalize the calculation with relative error ϵ once it has participated in $\Theta(\log n + \log \frac{1}{\epsilon})$ rounds. Note that this requires an estimate of n; we are not alone in this regard [4, 5]. It is possible to calculate n iteratively using Push-Sum itself.

Message loss can be an additional problem with Push-Sum in practise. If fish or water go missing, mass conservation is not observed and the correctness proof fails. The simple solution we take is to send incremental sums in each message. So instead of sending w_i we send $\sum_{j \leq i} w_j$. The receiving peer can now easily derive water lost in any messages prior to the currently received message. This requires keeping some additional state for each neighbour a peer has.

After solving these problems, there is some additional low-hanging fruit. If IP:Port is used as fish size, a peer far from the graph centre might be selected, every time. A simple way to rotate the fish is to feed the IP:Port through a different bijective function for each measurement number. For example, interpret the IP:Port and measurement number as elements of the the finite field GF(2^{48}) and multiply them.

Another easy improvement is packing multiple statistics into each gossip exchange. The protocol described thus far contains (measurement number, fish size, fish quantity, water) in each message. If one wants to compute several statistics, which is quite typical, it suffices to additionally send only the water for each further statistic; the fish size/quantity and measurement number can be reused.

4. SIMULATION

We use the improved Push-Sum algorithm in the P2P BubbleStorm system [6]. Figure 1 shows the simulated convergence of seven consecutive network size measurements. The network is a 10-regular random graph; such graphs are known to have good expansion with high probability [1]. The scenario starts with one million nodes, of which 50% leave at 28:00 minutes. The network is under constant churn with Poisson arrival rate and exponential peer lifetimes. Each peer gossips with its neighbours every 5 seconds, measured by its own clock, not synchronized with other peers.

Figure 1 plots the maximum and minimum network size estimates of all peers in the system each second; notice that the scale is logarithmic. The standard deviation drops very quickly initially as the dominant fish spreads through-out the network. Thereafter, it continues improving exponentially as predicted ($\log \frac{1}{\epsilon}$ rounds). After sufficient accuracy is achieved, one or more peers finalize their estimate and start a new measurement, resulting in the periodic behaviour.

5. RECOMMENDATIONS

We found that implementing Push-Sum poses no real difficulties, so let's compare it to the more recent competition. Figure 2 summarizes the asymptotic complexity of various algorithms on expander graphs. The hidden constants are small and roughly comparable. All algorithms have been made parallel for fair comparison of message rounds. The parameters are network size (n) and relative error (ϵ).

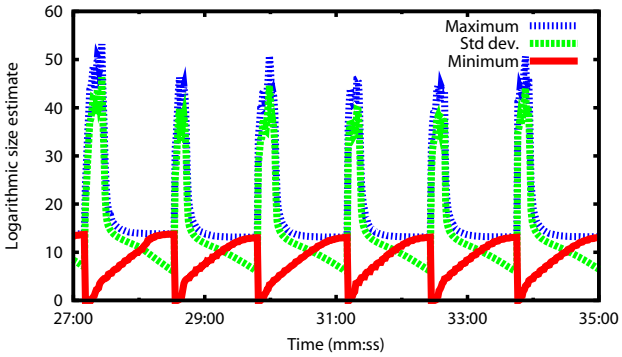

Figure 1: Push-Sum convergence with 1M peers

	Rounds	Messages
Push-Sum [3]	$\log n + \log \frac{1}{\epsilon}$	$n\left(\log n + \log \frac{1}{\epsilon}\right)$
Sample&Collide [4]	$\log n + \frac{1}{\epsilon}\sqrt{n}$	$\frac{1}{\epsilon}\sqrt{n}\log n$
Random Tour [4]	$n + \frac{1}{\epsilon^2}$	$\frac{1}{\epsilon^2} n$
Comp&Spread [5]	$\frac{1}{\epsilon^2}\log^2 n$	$\frac{1}{\epsilon^2} n \log^2 n$

Figure 2: Asymptotic complexity of sum algorithms

For latency (rounds required), Push-Sum wins for all parameters. For total messages, Push-Sum performs the best when high accuracy is required ($\frac{1}{\epsilon}$ large). However, the random walk approaches from [4] can finalize inaccurate measurements with less messages. Practically speaking, it's unclear how much of an advantage this is. Sample&Collide can only measure n, not general sums, and for accuracy beyond $\pm\sqrt{n}$, its message complexity is higher. Random Tour (on realistic network sizes) is worse for accuracy beyond $\pm 20\%$.

Whenever reasonable accuracy or low latency are required, Push-Sum appears to be the appropriate choice. Given its generality and the ease of implementation, current competing approaches seem seldom preferable. Therefore, research into generalizing the Push-Sum algorithm and related theorems for asynchronous environments would likely be fruitful.

6. REFERENCES

[1] B. Bollobás. *Random Graphs*. Cambridge University Press, 2nd edition, 2001.

[2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip Algorithms: Design, Analysis and Applications. In *INFOCOMM*, pages 1653-1664, 2005.

[3] D. Kempe, A. Dobra, and J. Gehrke. Gossip-Based Computation of Aggregate Information. In *FOCS*, pages 482–491, 2003.

[4] L. Massoulié, E. Le Merrer, A.-M. Kermarrec, and A. Ganesh. Peer Counting and Sampling in Overlay Networks: Random Walk Methods. In *PODC*, pages 123-132, 2006.

[5] D. Mosk-Aoyama and D. Shah. Computing Separable Functions via Gossip. In *PODC*, pages 113-122, 2006.

[6] W. W. Terpstra, J. Kangasharju, C. Leng, and A. P. Buchmann. BubbleStorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search. To appear in *SIGCOMM*, 2007.