Epidemic Algorithms for Replicated Database Maintenance

Xerox Palo Alto Research Center

1987

Idea

Problem — mutual consistency of database replicas in the face of updates

Approach — randomized epidemic algorithms to distribute updates

Note — not strong sequential consistency but eventual consistency *(If all updating stops then eventually all replicas will converge to identical values).*

Setup

- Update injected at a single site and propagated to all sites or substituted by a later update
- Considered factors
  - ✓ Time to propagate the update
  - ✓ Network traffic to propagate an update

  (update_size * no_servers)

Epidemiology

- Infective – has an update and wants to share it
- Susceptible – has not yet received an update
- Removed – received an update but no longer wants to share it

Simple epidemic

- ✓ only susceptible or infective
- ✓ eventually infect the entire population in time proportional with to log(n), n=population size

Complex epidemic

- ✓ susceptible, infective and removed
- ✓ may not infect the entire population

Strategies

- Direct Mail
- Anti-entropy
- Rumor mongering

Direct Mail

- Each new update is mailed from the entry site to all others servers

+ easy to implement
+ timely efficient
+ good for small & static servers
- not scale (O(n) msg per update)
- mail may get lost (queue overflow)
- detecting failures => network admin

Anti-entropy

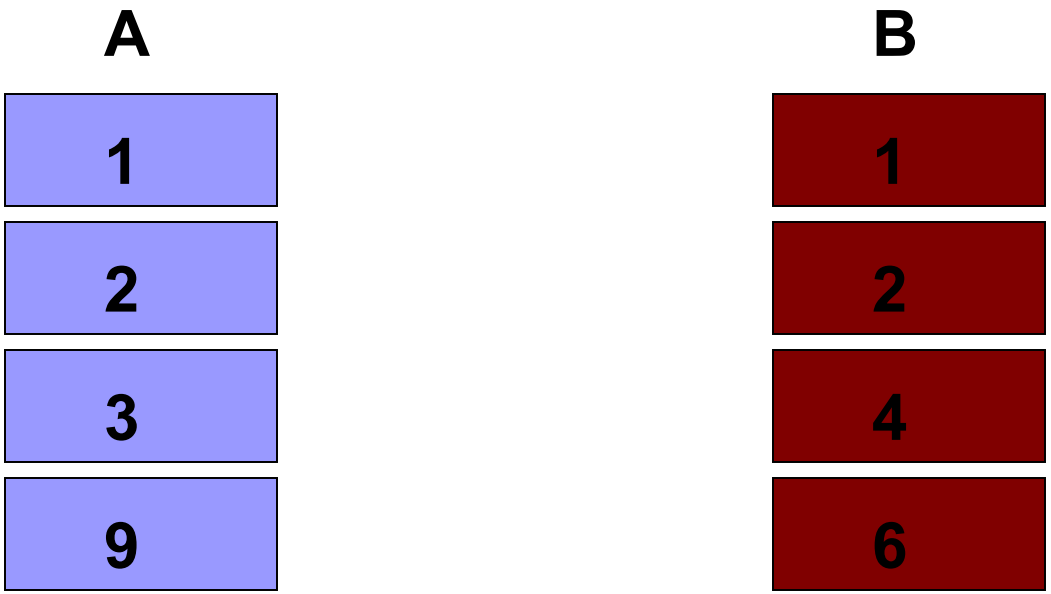
- Server picks random server and resolves differences by exchanging database contents

+ simple epidemic (will affect entire population)
+ reliable
- examine the database contents (cannot be frequent)
- propagate updates slower then direct mail
- too much bandwidth

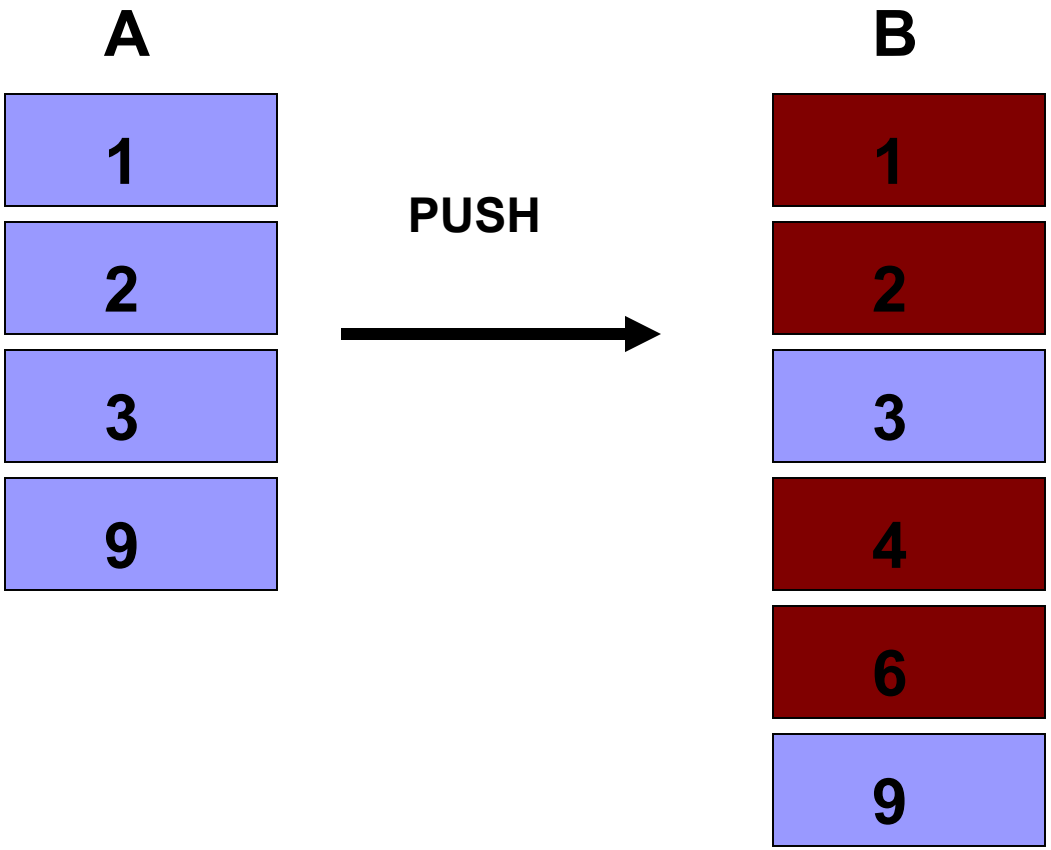
Anti-entropy
Resolve differences

- Push
- Pull
- Push-Pull

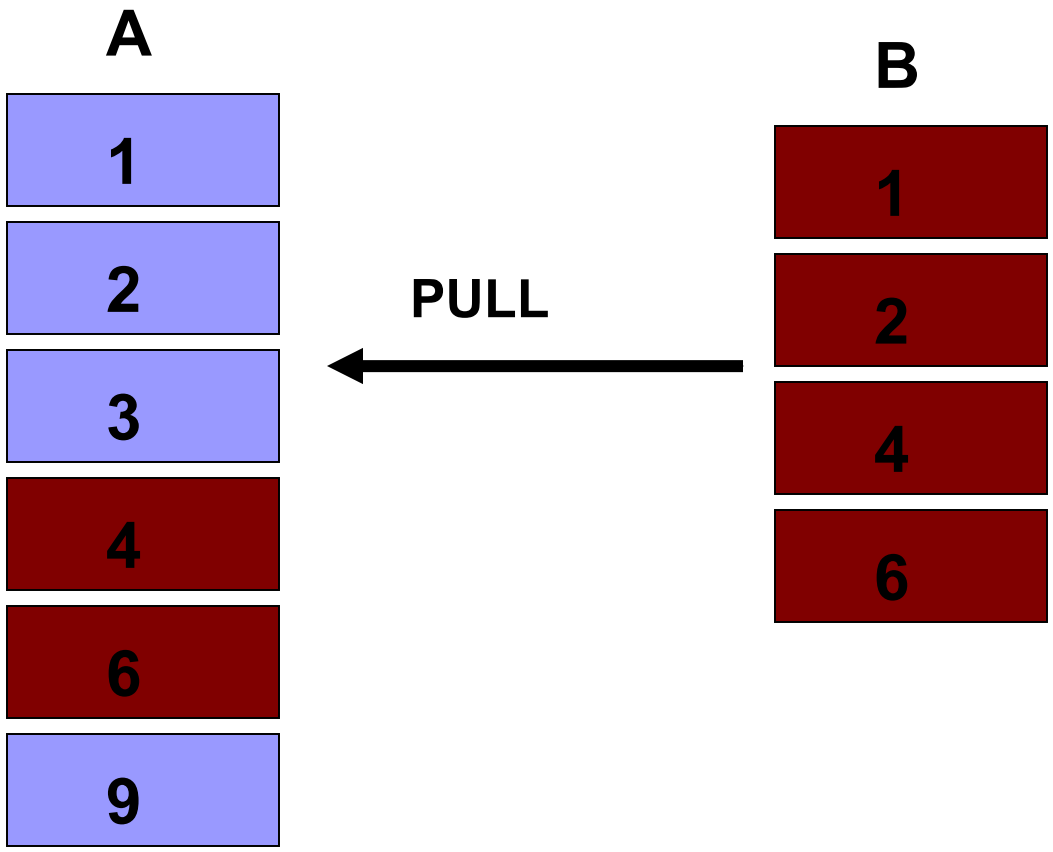
Anti-entropy Example



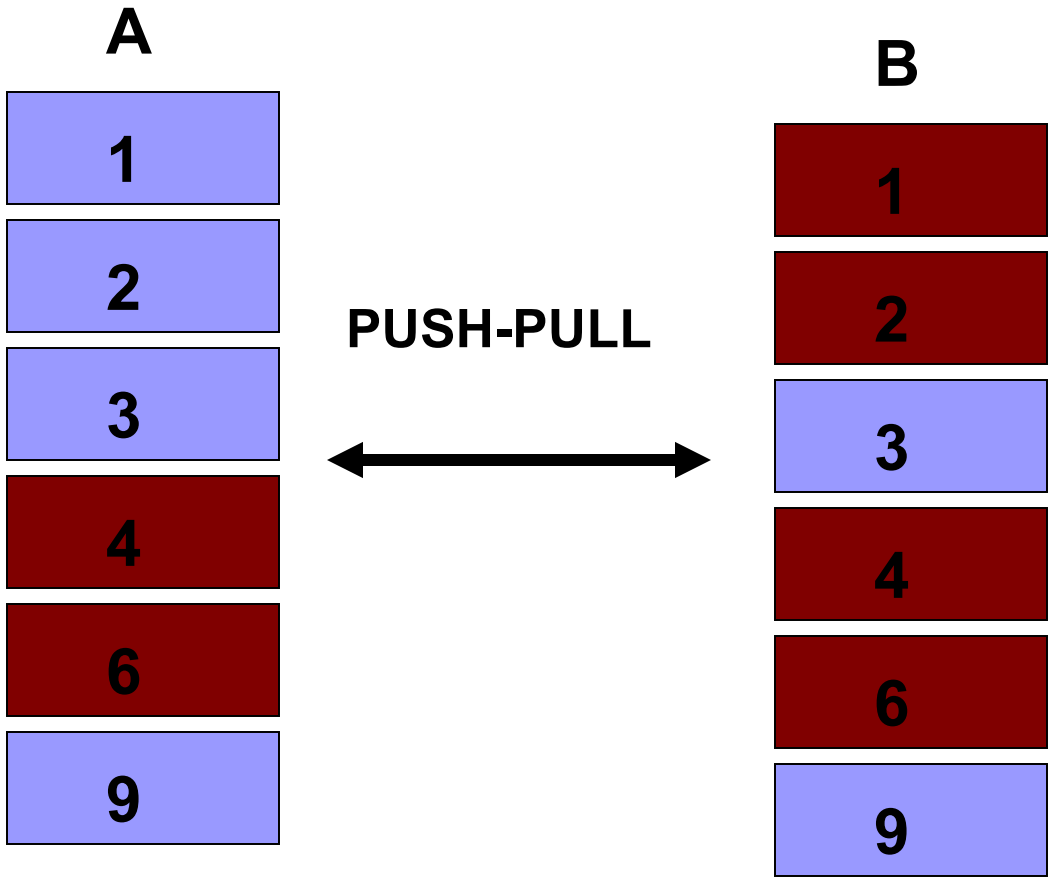
Anti-entropy Push



Anti-entropy
Pull



Anti-entropy Push-Pull



Anti-entropy as Backup

- When anti-entropy is used as backup for other distribution mechanism pull or push-pull is greatly preferable to push (the probability for a site to remain susceptible after a number of cycles converges faster to 0 with pull)

Anti-entropy Optimizations

- compute database content checksum
- exchange list of recent updates, apply the updates and compute checksum
- exchange updates in reverse chronological order until the checksum agree
- what does a recent update mean?

Clearinghouse Servers

- maintains translations from names to machine addresses (similar DNS)
- use direct-mail & anti-entropy once per day, but anti-entropy did not complete because of network load

  => whenever 2 anti-entropy participants disagreed, remailing step

  => breakdowns in all the network services

Rumor mongering

- Initially, all sites are ignorant
- A site gets a new update and becomes “hot rumor”
- While a site is “hot rumor”, it picks another site and sends out the update
- After trying to share a “hot rumor” with too many sites that know it, it stops

+ require fewer resource then anti-entropy

- complex epidemic (an update may not reach all sites)

Rumor mongering Implementation

- Sender keeps a list of infective updates
- Receiver inserts each update in the database and adds new updates to the infective list
- When to remove an update from the infective list?

  It losses interest to spread the rumor with probabili $s = e^{-(k+1)(1-s)}$.
- Probability of remaining infected after the epidemic finishes

Criteria for good epidemic

- Low Residue - remaining susceptibles when epidemic finishes
- Low Traffic - total update traffic/no of sites
- Low Delay - average time and last time to propagate an update

Variations in rumor mongering

- Blind vs Feedback (receiver knows the update)
- Coin vs Counter (losses interest after k unnecessary contacts)
- Push (numerous independent updates) vs Pull (quiescent database)
- Minimization (increase the smaller counter)
- Connection limit
- Hunting (connection rejected => hunt site)

Feedback Counter vs Blind Prob.

Counter k	Residue s	Traffic m	Convergence t_{ave}	t_{last}
1	0.176	1.74	11.0	16.8
2	0.037	3.30	12.1	16.9
3	0.011	4.53	12.5	17.4
4	0.0036	5.64	12.7	17.5
5	0.0012	6.68	12.8	17.7

Feedback and Counter (k)

Counter k	Residue s	Traffic m	Convergence t_{ave}	t_{last}
1	0.960	0.04	19	38
2	0.205	1.59	17	33
3	0.060	2.82	15	32
4	0.021	3.91	14.1	32
5	0.008	4.95	13.8	32

Blind and Probability (1/k)

Rumor mongering with Anti-entropy

- Rumor mongering spreads updates fast with low traffic network, but not complete
- Run anti-entropy infrequently to make sure every server gets the update
- Rumor mongering vs Direct Mail reduces the cost of redistribution
- Reduce average link traffic by a factor of 4 and on some links by a factor of 30

Death certificates

- Deleted items – death certificates with timestamp spreaded like updates
- When to delete a death certificate?
- If kept for a fixed threshold, obsolete items can be resurrected

Dormant death certificates

- Use 2-thresholds t1,t2 – most sites delete it after t1 and retention sites delete after t2
- Dormant death certificate encounters obsolete update => the death certificate is awakened
- Activate a death certificate :

  timestamp=current time => may cancel a legitimate update
- Use 2 timestamps:

  original - to cancel an item

  activation - when a death certificate is considered dormant

Spatial distribution

- Network is not uniform, certain links are overloaded
  - 80 conversations / transatlantic links
  - 6 average conversations per link
- Favor nearby neighbors

Spatial distribution Implementation

- Each site s sorts the list of sites by distance from it
- Select anti-entropy partners from the sorted list according to a function f(i), i=index in the sorted list
- f(i)=i^a, a = parameter for tuning spatial distribution

Spatial distribution Results

Table 5. Simulation results for anti-entropy, connection limit 1.

Spatial Distribution	t_{last}	t_{ave}	Compare Traffic		Update Traffic	
			Average	Bushey	Average	Bushey
uniform	11.00	6.97	3.71	47.54	5.83	75.17
$a = 1.2$	16.89	9.92	1.14	6.39	2.69	18.03
$a = 1.4$	17.34	10.15	1.08	4.68	2.55	13.68
$a = 1.6$	19.06	11.06	0.94	2.90	2.32	10.20
$a = 1.8$	21.46	12.37	0.82	1.68	2.12	7.03
$a = 2.0$	24.64	14.14	0.72	0.94	1.94	4.85

Table 6. Simulation results for *push-pull* rumor mongering.

Spatial Dist	k	t_{last}	t_{ave}	Compare Traffic		Update Traffic	
				Avg	Bushey	Avg	Bushey
uniform	4	7.83	5.32	8.87	114.0	5.84	75.87
$a = 1.2$	6	10.14	6.33	3.20	18.0	2.60	17.25
$a = 1.4$	5	10.27	6.31	2.86	13.0	2.49	14.05
$a = 1.6$	8	11.24	6.90	2.94	9.80	2.27	10.54
$a = 1.8$	7	12.04	7.24	2.40	5.91	2.08	7.69
$a = 2.0$	6	13.09	7.74	1.99	3.44	1.90	5.94

Conclusions

- Complex deterministic algorithms => randomized algorithms
- Higher performance in achieving consistency
- Reduced network traffic
- No guarantees from the underlying communication system