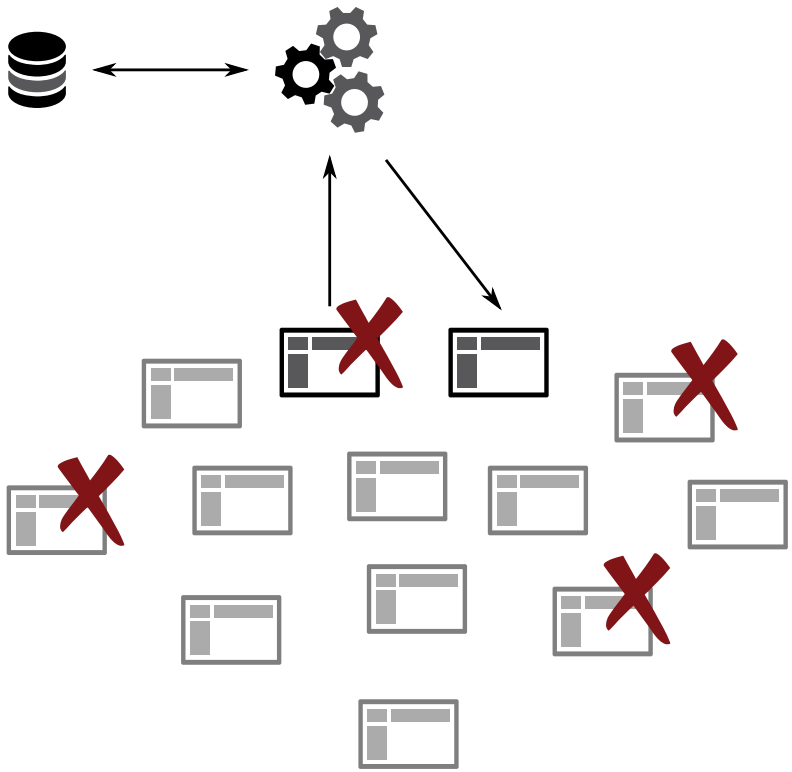
Keeping Distributed Systems in Sync

International PHP Conference

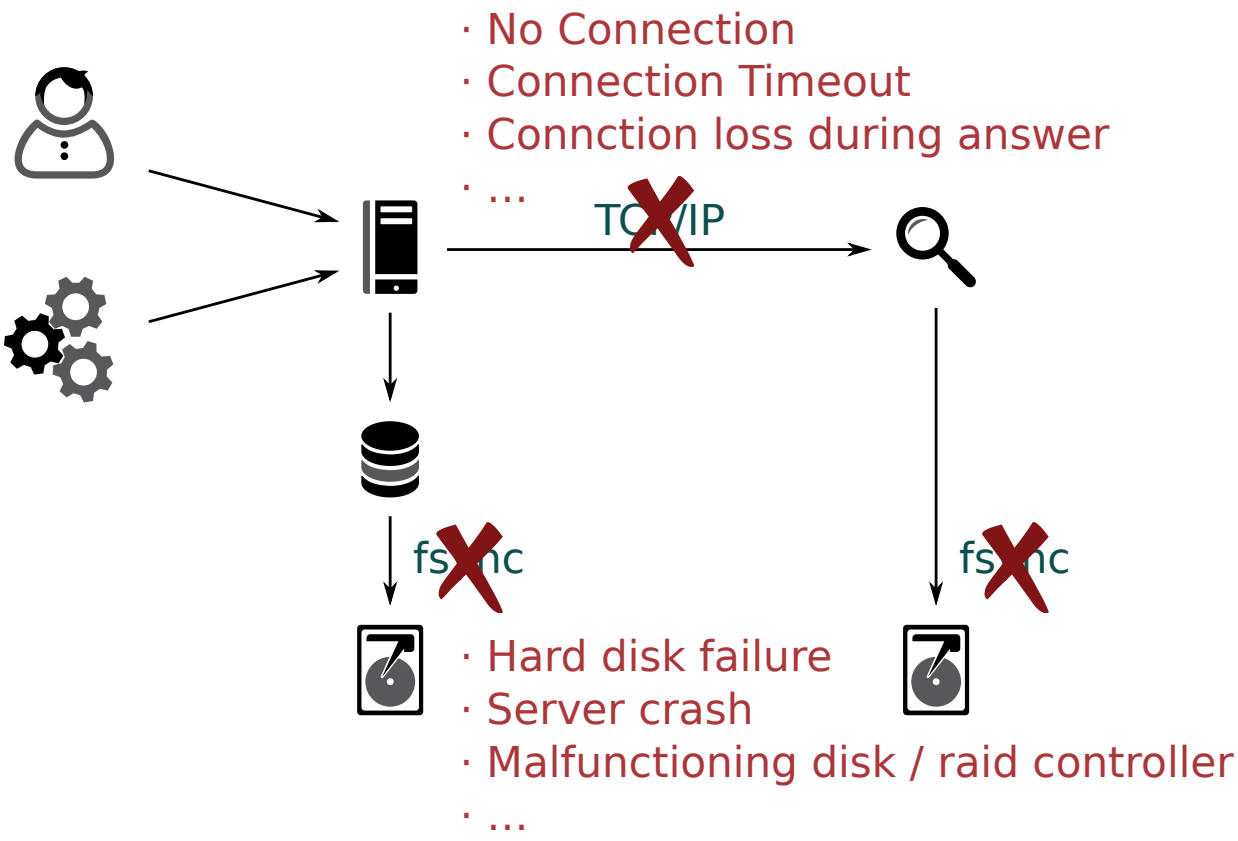
Kore Nordmann / @koredn / <kore@qafoo.com>

June 9th, 2015

The Usecase



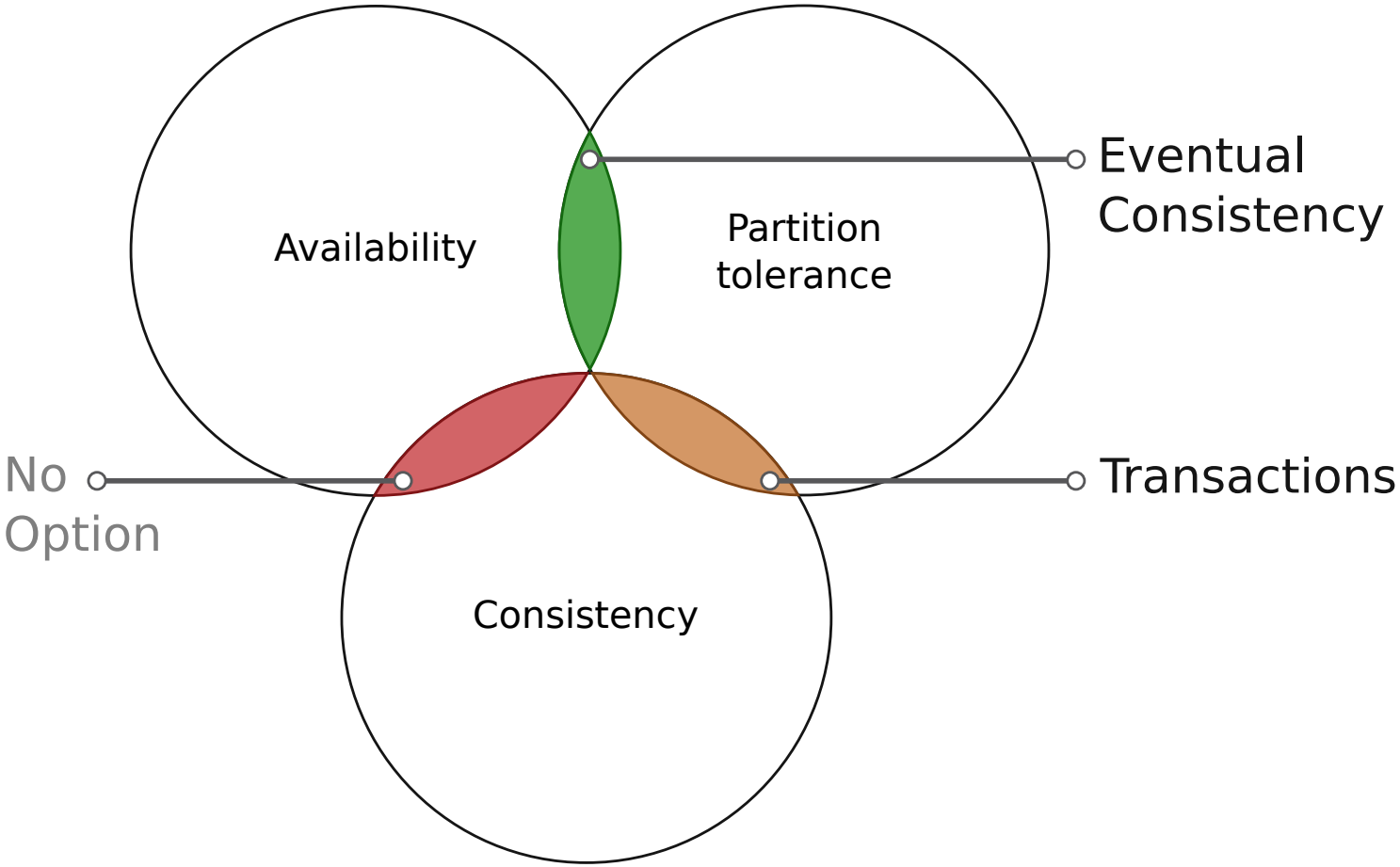
A Simpler Usecase



Solutions

- Transactions
  - Write is only ACK'd if all nodes ACK'd
    - Not possible if nodes do not ACK properly (Solr, MongoDB, ElasticSearch, . . .)
    - Two / three phase commits take time. . .
  - Rollback and deny writes entirely if one node does not ACK
    - Omitted rollback requires full-sync
    - Requires re-transmitting all data
    - Checking which IDs are transmitted requires iterating all IDs
- Eventual Consistency

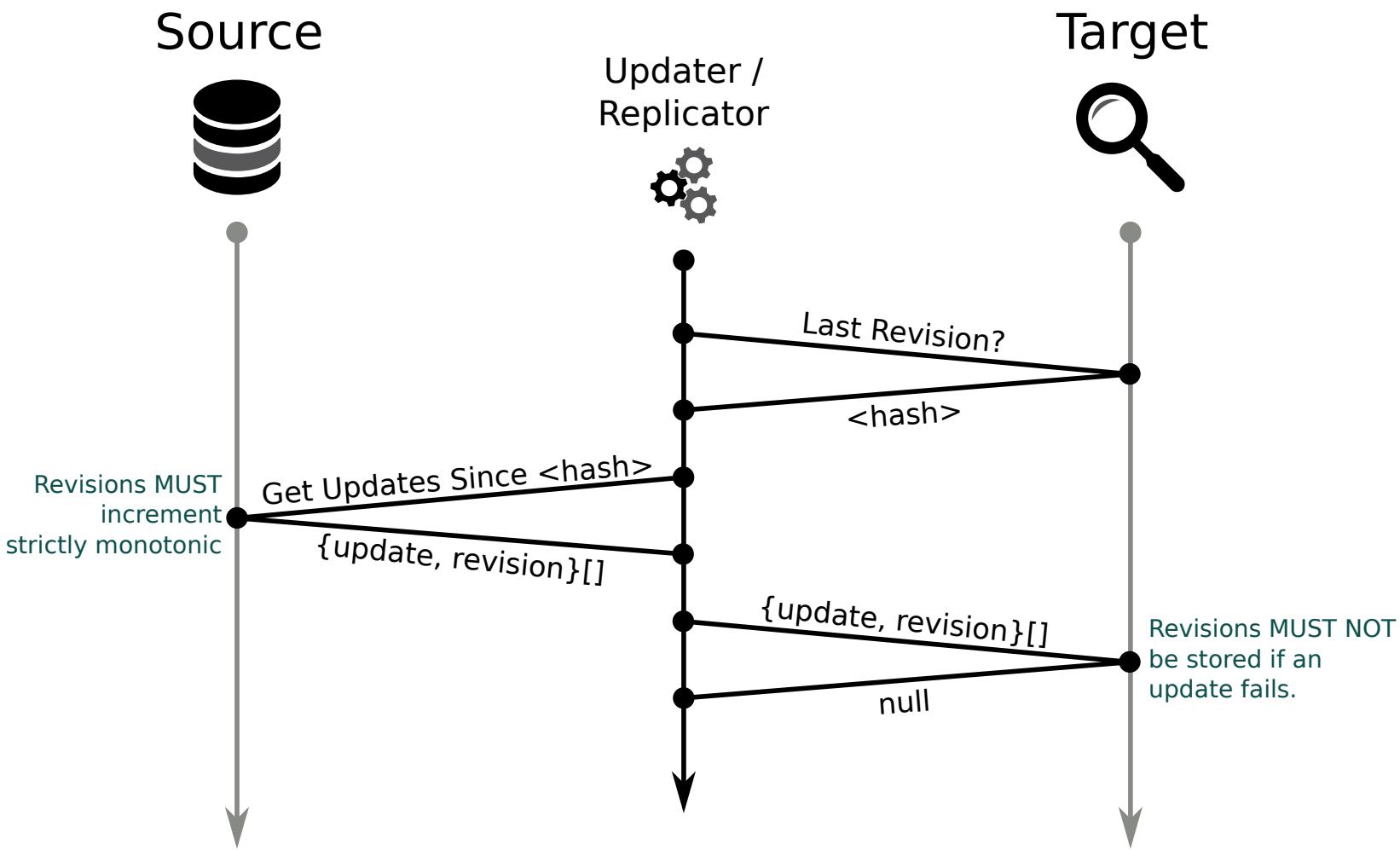
CAP Theorem



Eventual Consistency

Sounds good – but how?

Eventual Consistency



Eventual Consistency

That’s all?

No.

Replicator Implementation

- Implement as a dedicated process (daemon, cronjob, . . .)
- *Can* be implemented in PHP – we also have an Go implementation
- Protocol:
  - We used JSON-RPC and XML-RPC, but does not matter
  - Caching makes no sense and a single endpoint URL simplifies integration
- In a basic implementation it just dispatches RPC messages
  - Sharding, logging, request signing are optional, but sensible

Replicator Implementation

```
1       public function replicate($channel, Endpoint $source, Endpoint $target)
2       {
3           /* @var Result $lastUpdate */
4           $lastUpdate = $target->execute(
5               new Command('lastUpdate', $channel)
6           );
7
8           /* @var Result $updates */
9           $updates = $source->execute(
10              new Command('updates', $channel,
11                  array(
12                      'since' => $lastUpdate->payload['revision'],
13                  )
14              )
15          );
16
17          $target->execute(
18              new Command('replicate', $channel,
19                  $updates->payload
20              )
21          );
22      }
```

Replicator Implementation

- `Endpoint` will encode & send the command to the current endpoint
  - By default `Endpoint\JsonRPC`, but could also be `Endpoint\Solr`
- `Command` – simple data object containing:
  - `$method`
  - `$channel`
  - `$payload`
- `Result` – simple data object containing:
  - `$ok`
  - `$payload`

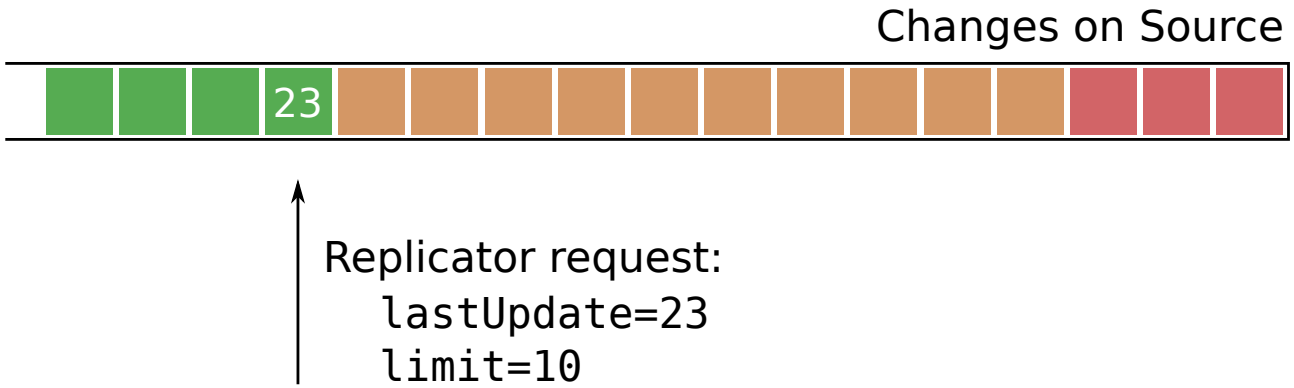
Data Modelling on Source

- Store denormalized “updates”
  - Revision (globally strictly monotonic)
  - Store full data
  - Keep deletes
- Maintaining referential integrity is hard – but not impossible

Replicator Scaling

- With large replication batches:
  - Use limit to reduce batch size
  - Use compaction (vacuum)
- With many targets:
  - Run separate processes / threads per target / target group
  - Reduce replication rate and volume for erroneous targets

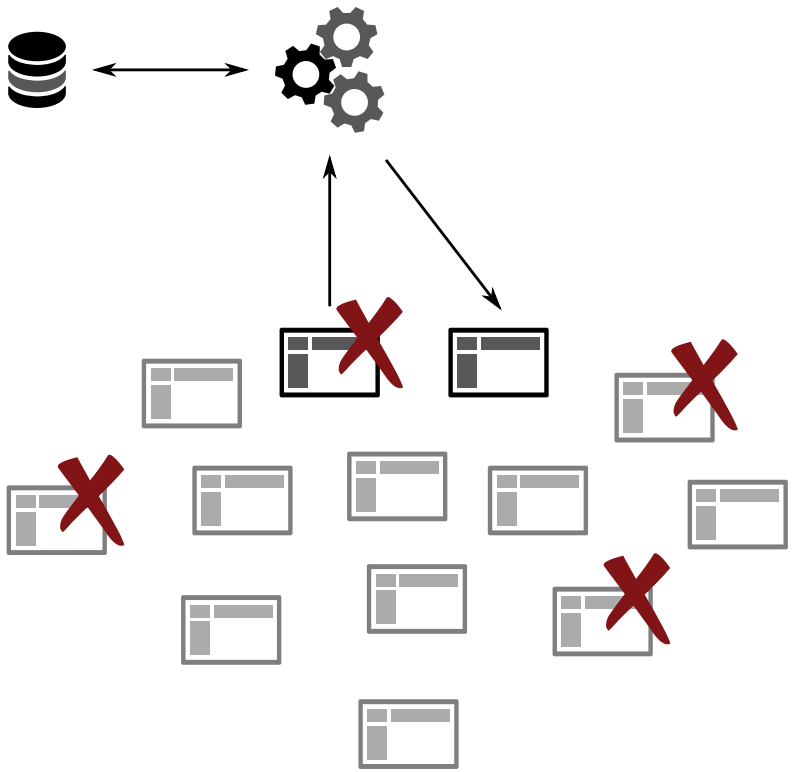
Replication Status



Other Tools

- Same mechanism can be found in:
  - Binary logs (MySQL, . . .)
  - Solr replication
  - CouchDB replication

Scale It



Summary

- Embrace Eventual Consistency
  - Transactional consistency with your search index does not work nor is required
- Implementation is more trivial then continuously checking consistency

THANK YOU

Rent a quality expert
qafoo.com