The History and Future of the Blurring of Stream Processing & OLTP

BERLIN BUZZWORDS 2017 JUNE 11-13

VOLTDB

John Hugg
June 12th, 2017
@johnhugg
jhugg@voltdb.com

Who Am I?

- Engineer #1 at VOLTDB
- Responsible for many poor decisions and even a few good ones.
- jhugg@voltdb.com
- @johnhugg
- http://chat.voltdb.com slack


Beatlejuice App

Minimum Viable Product

- Alexa/Siri-ish device in your house
- Sends discernible words to your servers
- If it hears “beetlejuice” three times, play sound clips from the movie
- Because everyone hates “Word Count”

“Just Use Postgres"

Dumbish Client

Audio

Play Sound

Rich Client Logic

Speech to text Etc…

Add record

Count records

Record Action & Reset Count

PostgreSQL

Running in IaaS or Datacenter

“Just Use Postgres"


```
CREATE TABLE BEETLECOUNT (
    user_id BIGINT NOT NULL,
    spoken_count INTEGER NOT NULL,
    PRIMARY KEY(user_id)
);
```

Storm?

Client

?

Running in IaaS or Datacenter

Why Storm?

- Would I use Storm for pretty much anything now? No.
- It's decently exemplar, well known, and not fancy.
- But there are soooo many other choices.



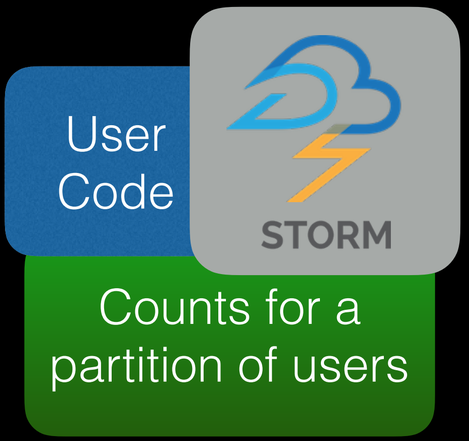

Storm

Client

Audio

Instructions to play sound

Sometimes…



Failure

Postgres Failure

Audio

Play Sound

- Postgres software or hardware stops
- Network fails

Storm

Client

Audio

Instructions to play sound

Sometimes…

User Code

STORM

Counts for a partition of users

User Code

STORM

Counts for a partition of users

Use Idempotence with At-Least-Once delivery for Exactly Once Semantics

Side Effects Ruin Everything

- Playing a sound on the speaker is not a transaction you can roll back. It’s an external action — A side effect.
- Good examples are SMS or a REST-API call.

Does Kafka Fix Failure?

All better?

Audio

Play Sound

kafka

Rich Client Logic

PostgreSQL

- More robust to Postgres failure in some ways
- Side effects can’t be helped much
- Latency hit
- But many good reasons to do this (coming up)

Commonalities: Stream & DB

- In order to ensure delivery, the original source has to be prepared to resend an event until acknowledged.
- Idempotency is the key to exactly-once-semantics.
- It's impossible to guarantee a side effect happens exactly once.
  So we do our best.
- Systems that are flexible/safe/accurate are really really hard.

What makes a “hard” app?

- Scale / Complexity
- Velocity of requirements changes
- Precision
  “exactly three times”
  Chaining precise conditions and actions
  Non-commutative math
- Side Effects
- Partial Control

In the tradition of: **Tabs vs Spaces** and **Emacs vs Vim**

and **Javascript Floating Point vs Anything Resembling Sanity** comes:

Stream vs Database

2017 Edition

Why Stream / Log?

- Can easily Tee (split into two identical streams) to prod and test, or to A/B test.
- Often allows for simpler clients, as richer processing is moved into the streaming system.
- Often easier to understand performance characteristics, especially under load.
- By replaying a logged stream, can often roll-back/forward to any recent state. Need truncating snapshots for this.
- Can sometimes make multi-DC easier and more consistent.
- Horizontal scalability and fault tolerance often easier.

Why Database

- Truncating a stream requires compaction or a snapshot. This is hard to get right for many apps.
- You can query it!
- Secondary indexes, Materialized views, Constraints, Joins, FKs
- The tooling is really mature.
- Typically more appropriate for apps that require lower latency.

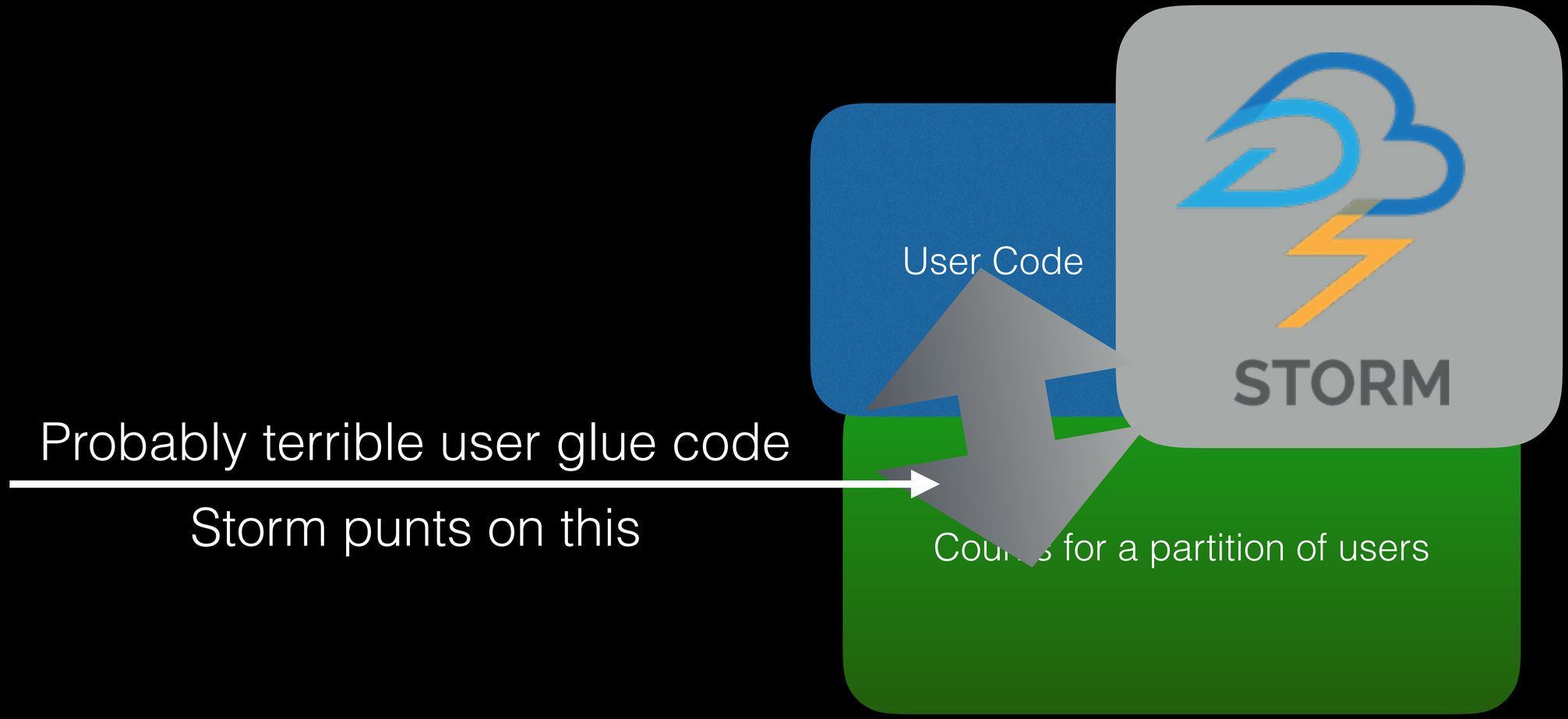
Use A Streaming System with a DB?

- Sometimes you can get a lot of the benefits of both
- More integration points and more ways to fail.

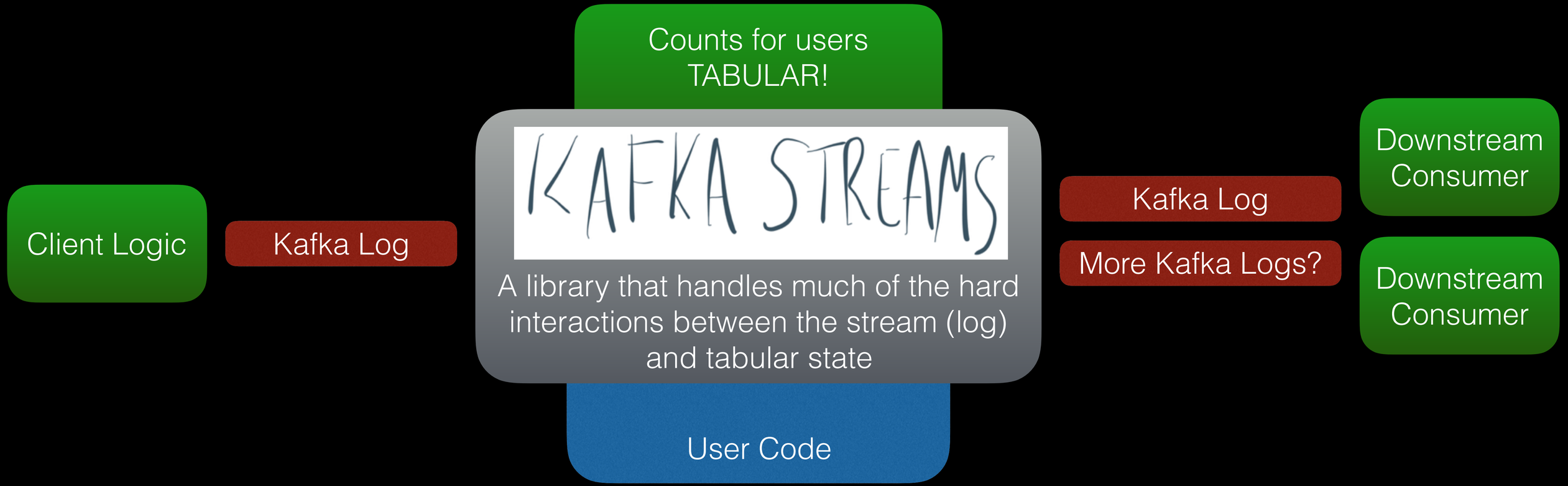
BUT IT'S OK TO WANT IT ALL

You promised blurring...

Add Databaseness to Streaming



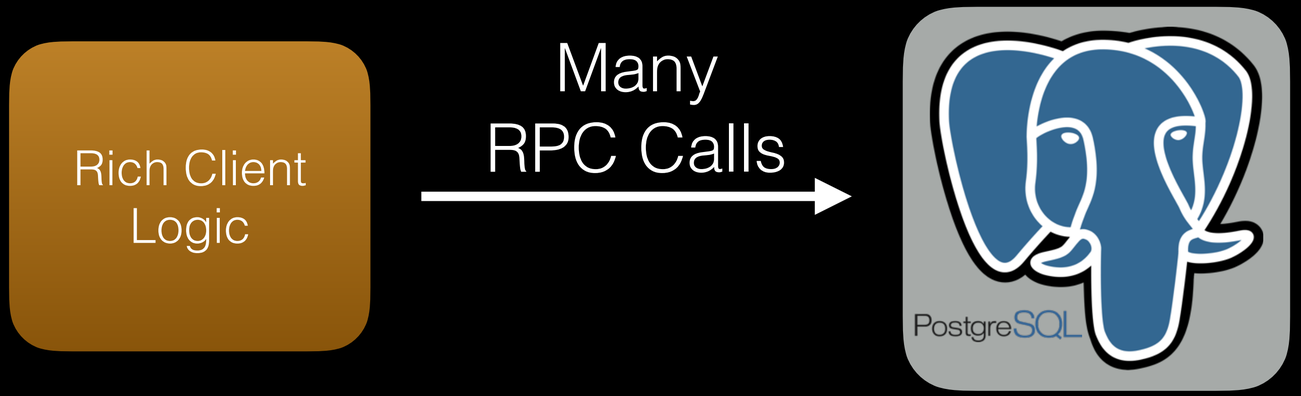
Add Databaseness to Streaming



- Other systems have ways to integrate state with streams, but none seem as ambitious as Kafka Streams

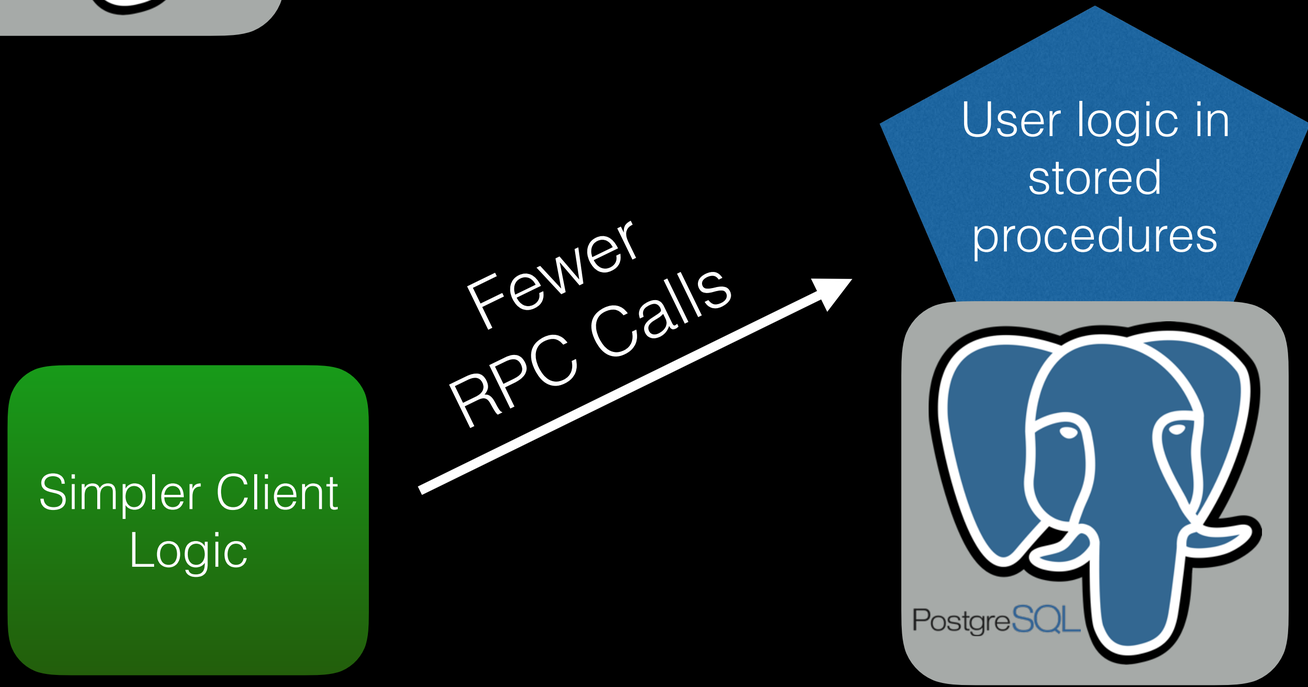

Make a database
that smells streamy

Put Processing in the DB

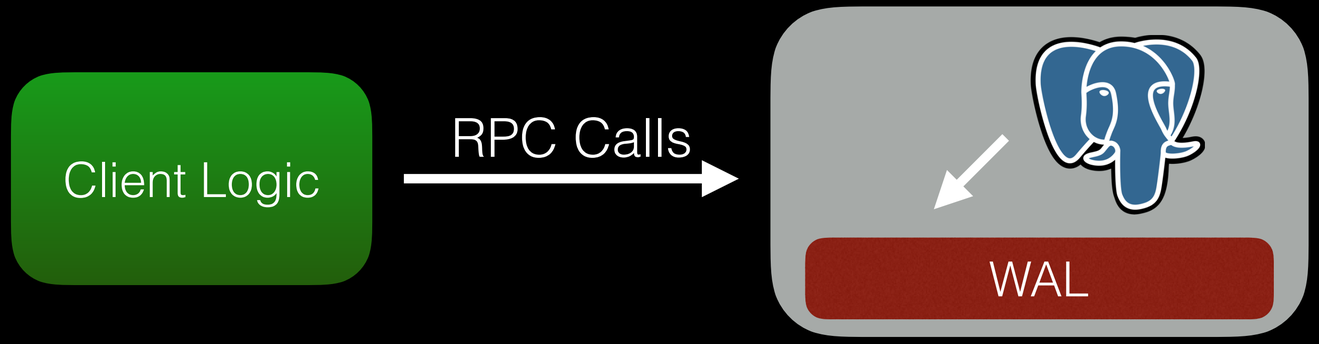


Beyond Postgres:

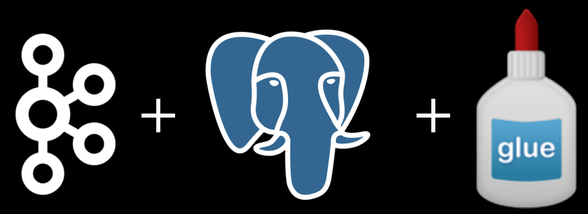
- Better tools for managing user code.
- Better tools for debugging.
- Better monitoring and transparency for user code running in the DB.



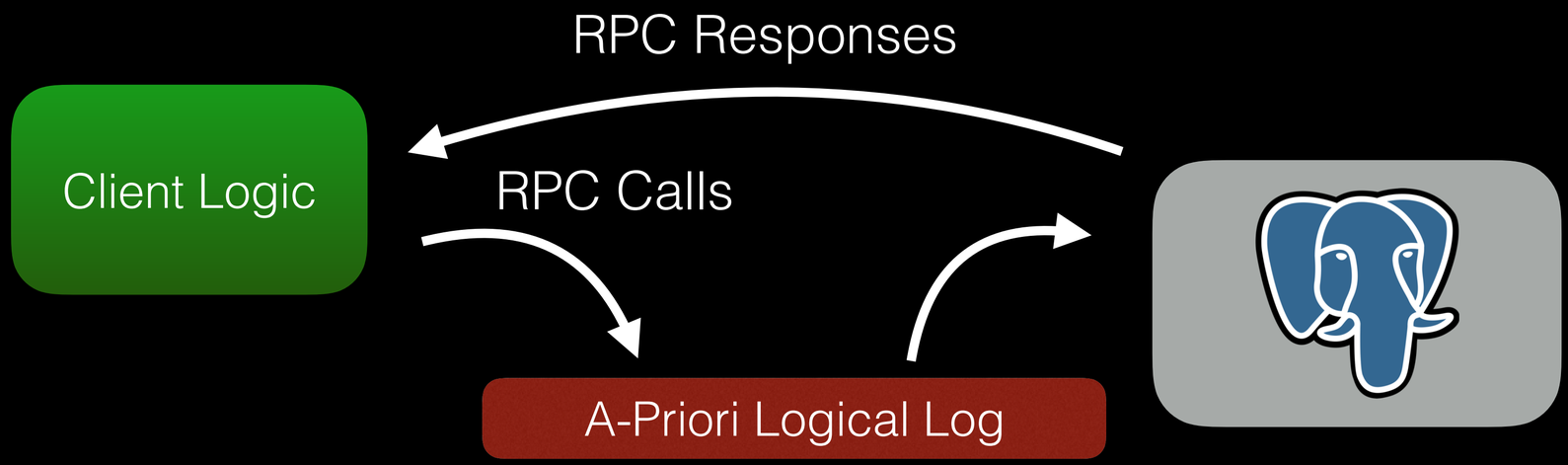
Give the DB a A-Priori-Log



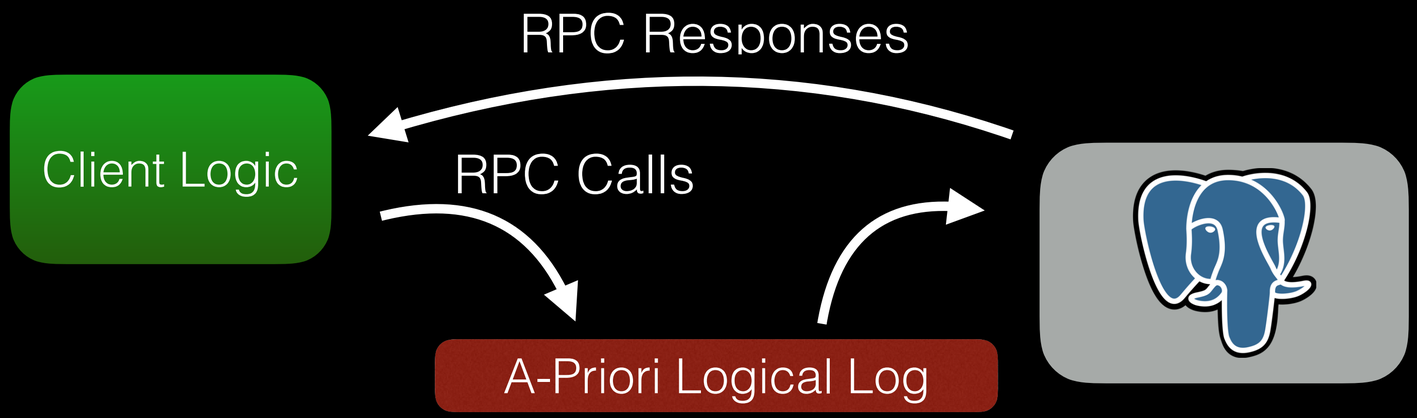
Could do this as

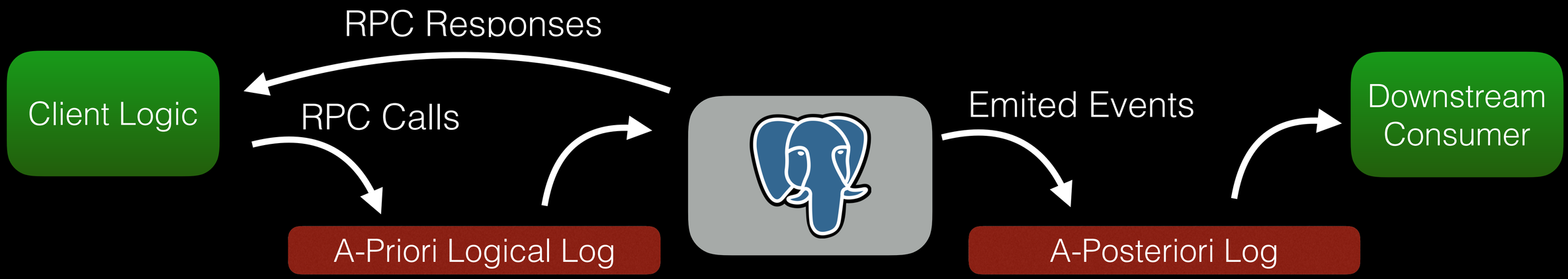


But an integrated product has many advantages.

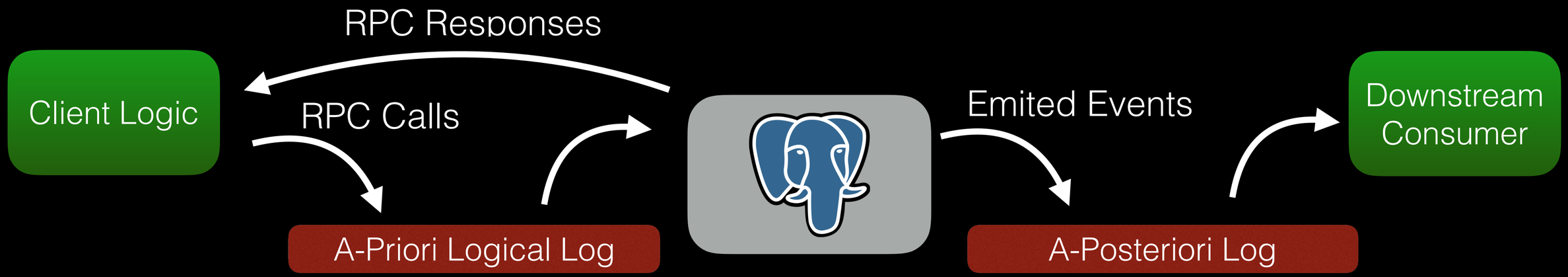


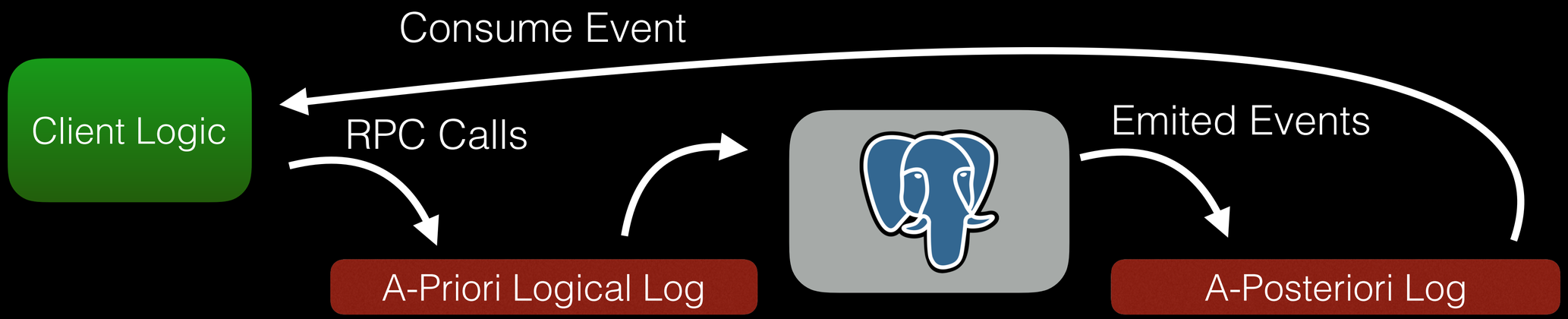
Give the DB an A-Posteriori Log



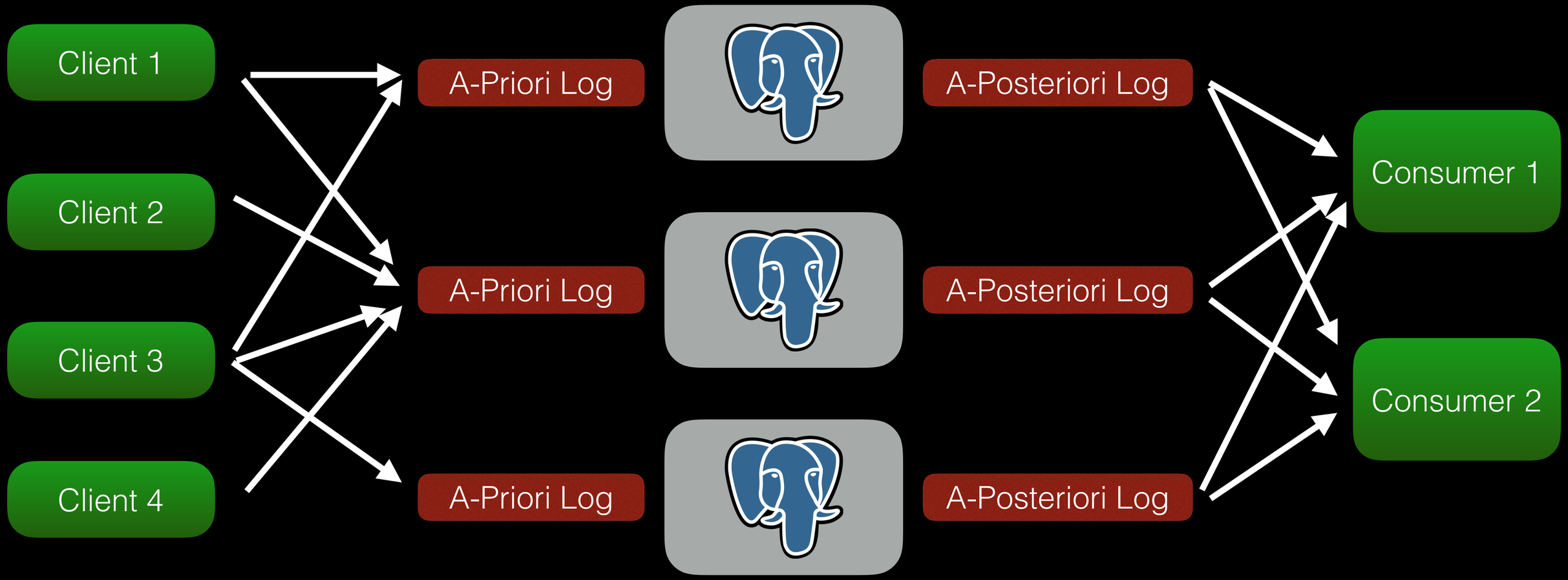


Give the DB an A-Posteriori Log

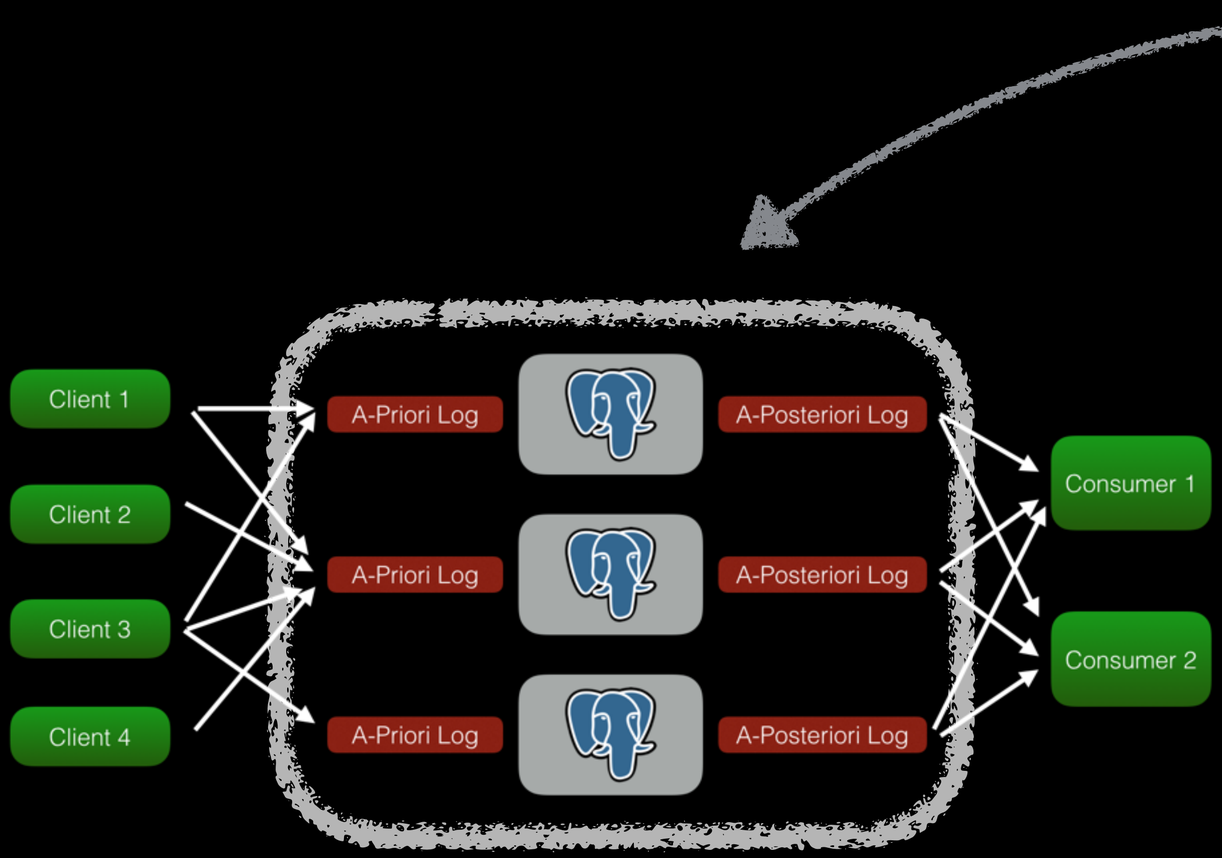




Horizontally Partition the DB

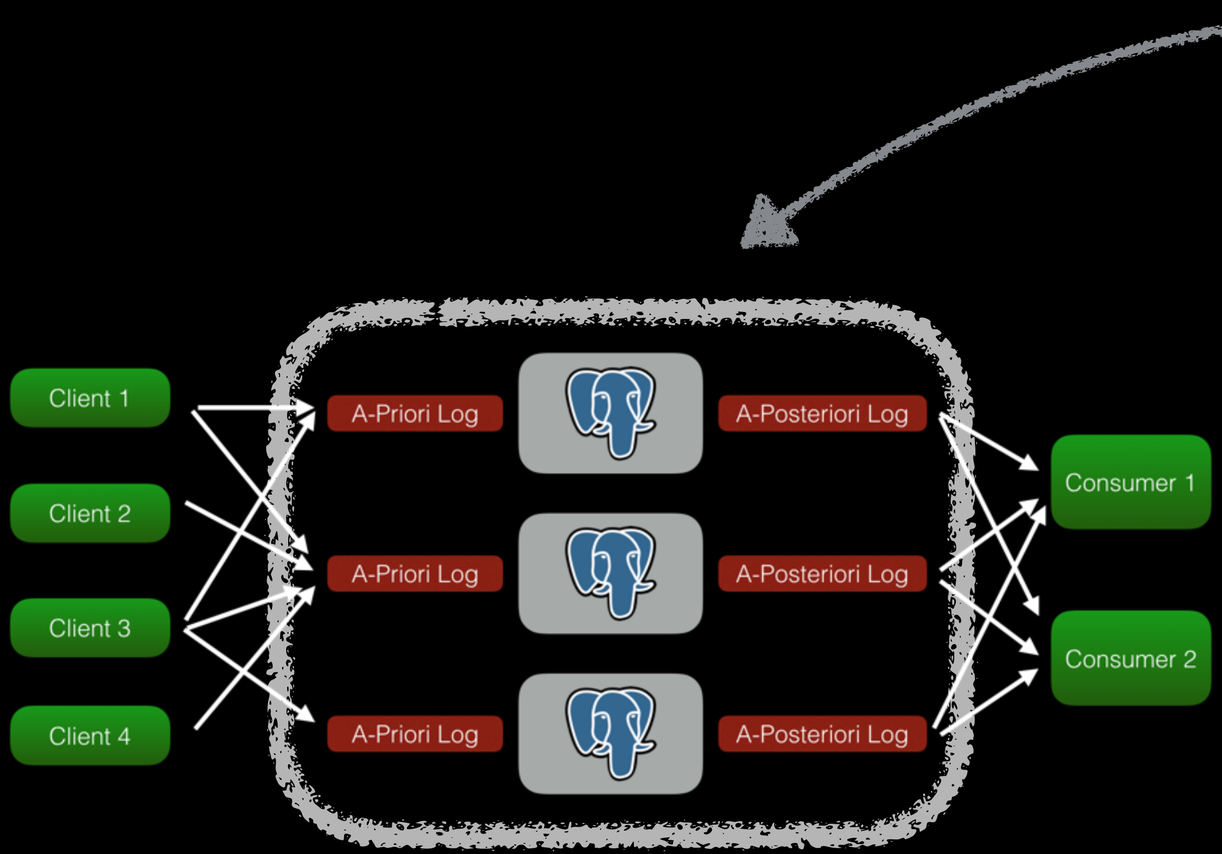


Throw out  but keep 



- Needs to present as a single, managed entity
- Global stats
- Global reads without extra work
- SQL, JDBC, ODBC, etc…
- Global writes? Maybe…

Throw out  but keep 



Because Latency


Where is this going?

We're building a lot of this at VOLTDB

- Horizonitally partitioned, but acts as a single system
- Per partition ordered input and ordered output
  All based on an a-priori logical log
- Debuggable Java stored procedures that can use 3rd party libs
- Ability to emit events into an a-posteriori log
- Native support for secondary indexes, ranking, materialized views, transactions, cross-partition operations, JDBC/ODBC, etc…

Conclusion

- Note: VoltDB is a not as general as we would like yet.
- The future of operations and OLTP is going to be a mix of *streams*, *logs* and *state.*
  We are getting good at these things individually.
- Let's build systems that tackle integration and the in-between problems.
  There's an opportunity here.

VOLTDB

Thank You!

@johnhugg
@voltdb

jhugg
@voltdb.com

chat.voltdb.com

- Please ask me questions now or later.
- Feedback on what was interesting, helpful, confusing, boring is ALWAYS welcome.
- <u>Happy to talk about:</u>
  - Data management
  - Systems software dev
  - Distributed systems
  - Japanese preschools