Apache Cassandra Present and Future

Jonathan Ellis

DataStax

History

Google

Bigtable, 2006

amazon.com

Dynamo, 2007

facebook

OSS, 2008

Incubator, 2009

TLP, 2010

1.0, October 2011

DataStax

Why people choose Cassandra

- Multi-master, multi-DC
- Linearly scalable
- Larger-than-memory datasets
- High performance
- Full durability
- Integrated caching
- Tuneable consistency

Cassandra users

- Financial
- Social Media
- Advertising
- Entertainment
- Energy
- E-tail
- Health care
- Government

Road to 1.0

- Storage engine improvements
- Specialized replication modes
- Performance and other real-world considerations
- Building an ecosystem

Compaction

Size-Tiered

Leveled

Differences in Cassandra's leveling

- ColumnFamily instead of key/value
- Multithreading (experimental)
- Optional throttling (16MB/s by default)
- Per-sstable bloom filter for row keys
- Larger data files (5MB by default)
- Does not block writes if compaction falls behind

Column ("secondary") indexing

```
cqlsh> CREATE INDEX state_idx ON users(state);
cqlsh> INSERT INTO users (uname, state, birth_date)
       VALUES ('bsanderson', 'UT', 1975)
```

users

	state	birth_date
bsanderson	UT	1975
prothfuss	WI	1973
htayler	UT	1968

state_idx

UT	bsanderson	htayler
WI	prothfuss	

Querying

```
cqlsh> SELECT * FROM users
       WHERE state='UT' AND birth_date > 1970
       ORDER BY tokenize(key);
```

More sophisticated indexing?

- Want to:
  - Support more operators
  - Support user-defined ordering
  - Support high-cardinality values
- But:
  - Scatter/gather scales poorly
  - If it's not node local, we can't guarantee atomicity
  - If we can't guarantee atomicity, doublechecking across nodes is a huge penalty

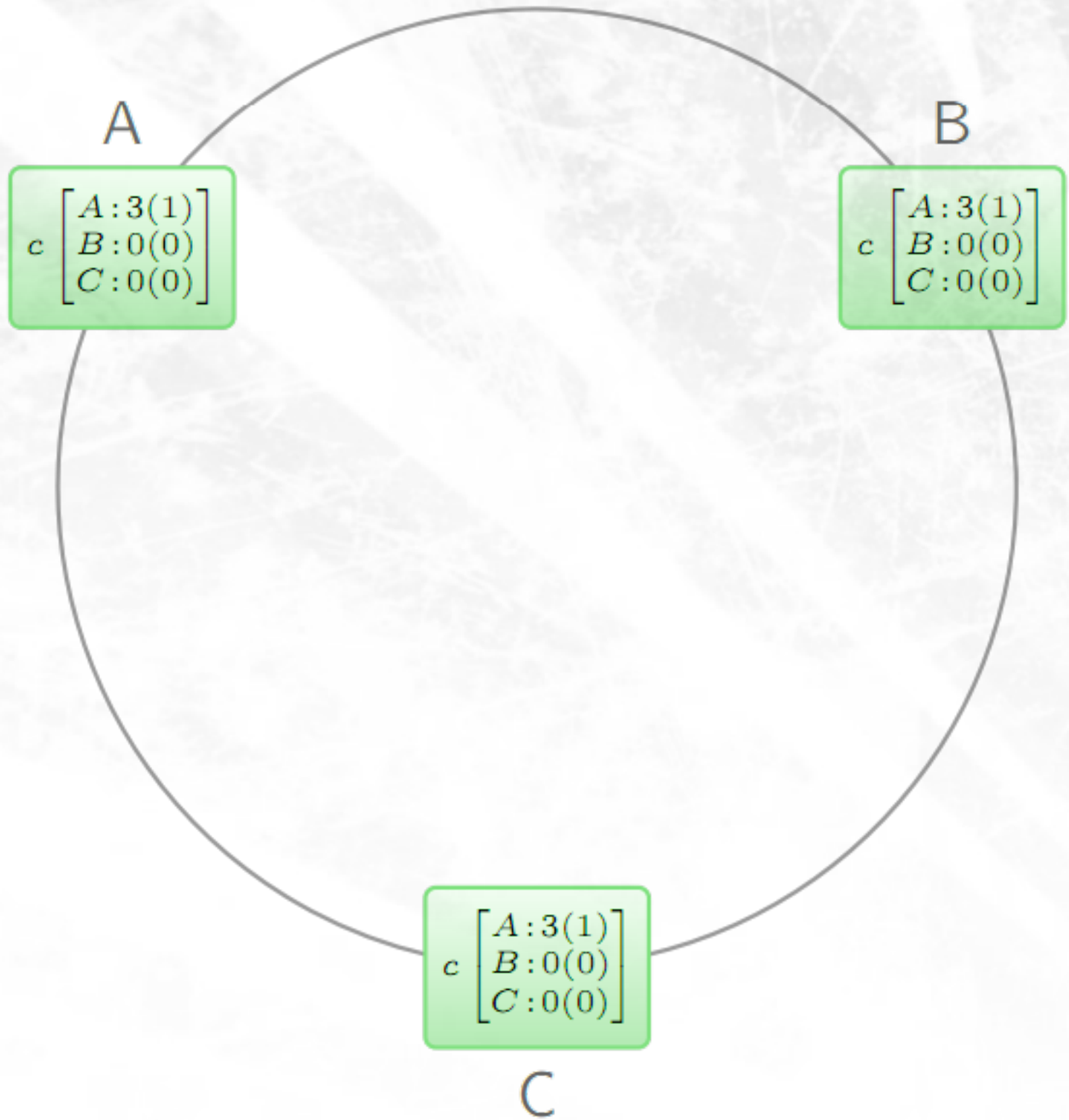
Other storage engine improvements

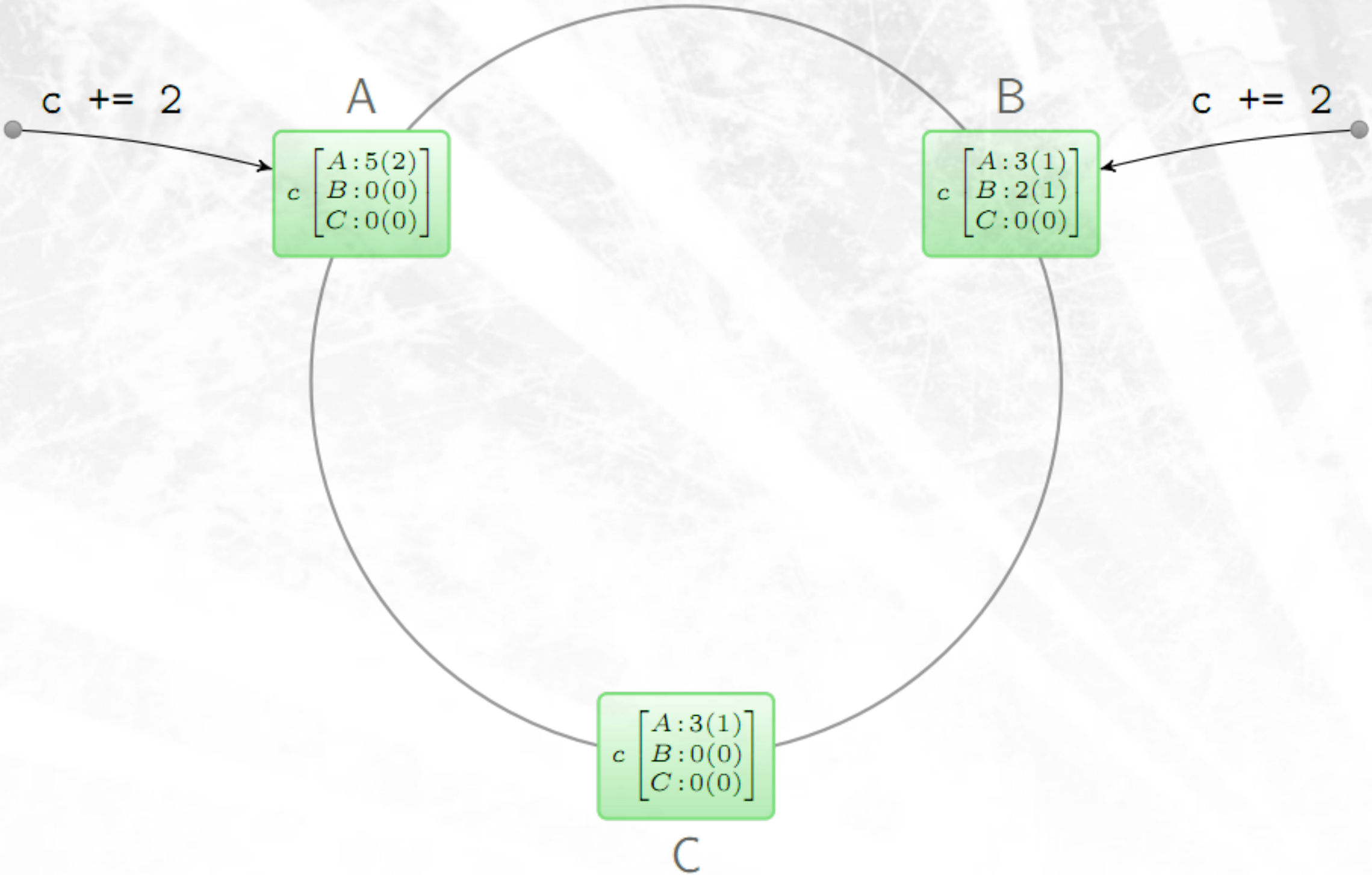
- Compression
- Expiring columns
- Bounded worst-case reads by re-writing fragmented rows

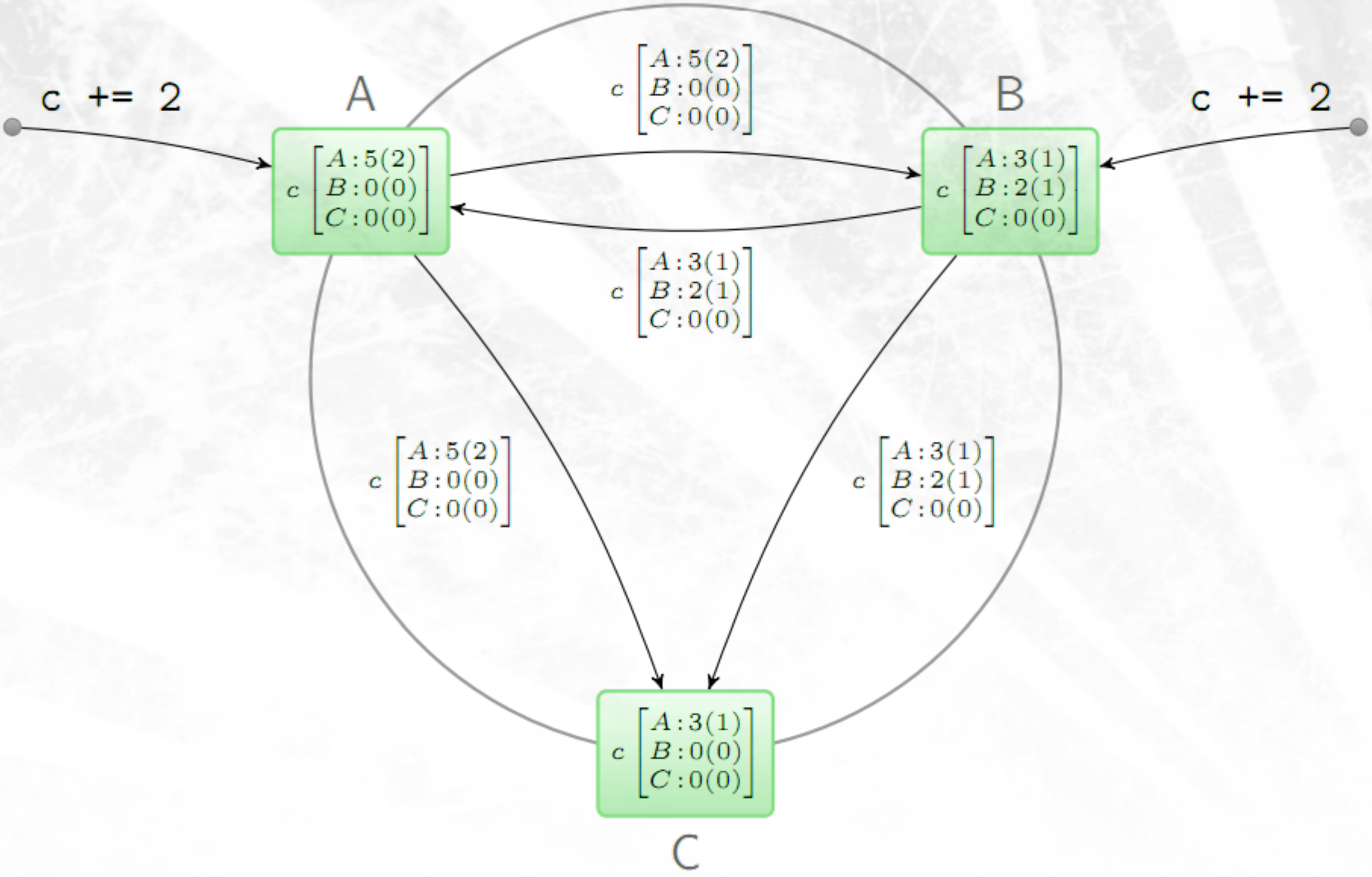
Eventually-consistent counters

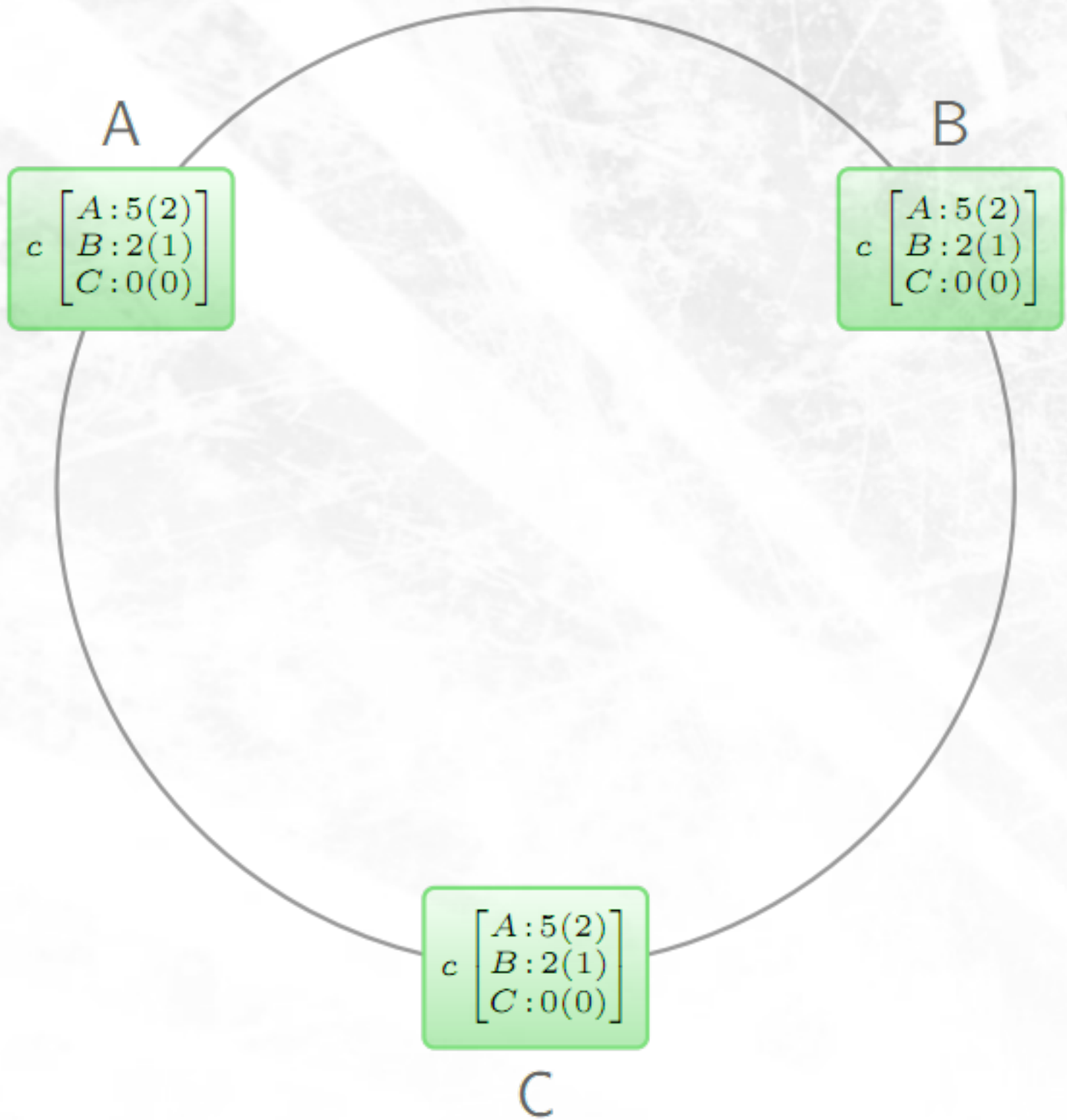
- Counter is partitioned by replica; each replica is "master" of its own partition

$$c: \begin{bmatrix} A:24\ (2) \\ B:42\ (3) \\ C:17\ (1) \end{bmatrix} \iff c = 83$$


A
B
C
$c\begin{bmatrix}A:3(1)\\B:0(0)\\C:0(0)\end{bmatrix}$
$c\begin{bmatrix}A:3(1)\\B:0(0)\\C:0(0)\end{bmatrix}$
$c\begin{bmatrix}A:3(1)\\B:0(0)\\C:0(0)\end{bmatrix}$


c += 2
A
$c \begin{bmatrix} A:5(2) \\ B:0(0) \\ C:0(0) \end{bmatrix}$
B
$c \begin{bmatrix} A:3(1) \\ B:2(1) \\ C:0(0) \end{bmatrix}$
c += 2
$c \begin{bmatrix} A:3(1) \\ B:0(0) \\ C:0(0) \end{bmatrix}$
C


c += 2
A
$c\begin{bmatrix}A:5(2)\\B:0(0)\\C:0(0)\end{bmatrix}$
$c\begin{bmatrix}A:5(2)\\B:0(0)\\C:0(0)\end{bmatrix}$
$c\begin{bmatrix}A:3(1)\\B:2(1)\\C:0(0)\end{bmatrix}$
B
c += 2
$c\begin{bmatrix}A:3(1)\\B:2(1)\\C:0(0)\end{bmatrix}$
$c\begin{bmatrix}A:5(2)\\B:0(0)\\C:0(0)\end{bmatrix}$
$c\begin{bmatrix}A:3(1)\\B:2(1)\\C:0(0)\end{bmatrix}$
$c\begin{bmatrix}A:3(1)\\B:0(0)\\C:0(0)\end{bmatrix}$
C


A
$c\begin{bmatrix}A:5(2)\\B:2(1)\\C:0(0)\end{bmatrix}$
B
$c\begin{bmatrix}A:5(2)\\B:2(1)\\C:0(0)\end{bmatrix}$
$c\begin{bmatrix}A:5(2)\\B:2(1)\\C:0(0)\end{bmatrix}$
C

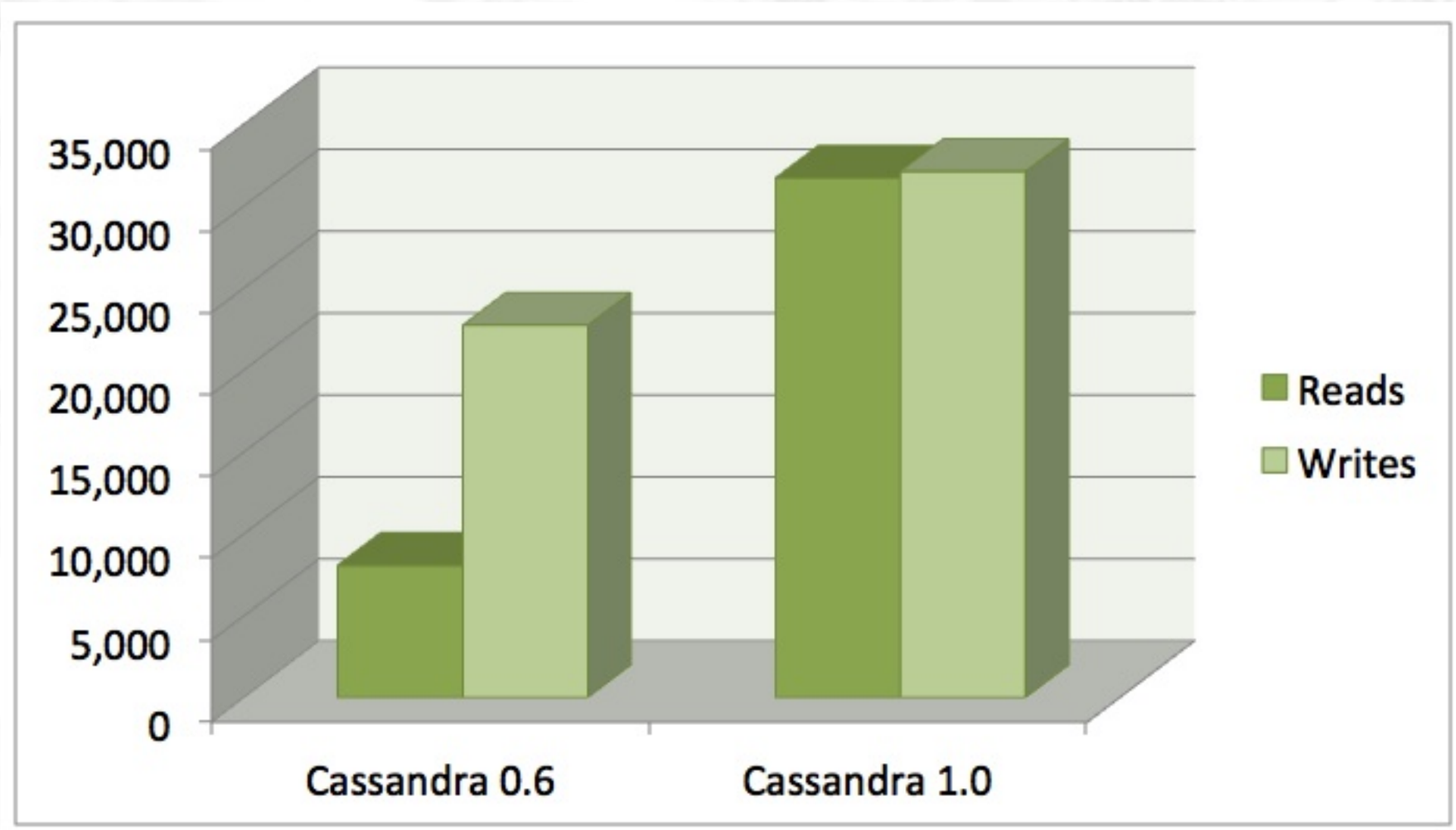
The interesting parts

- Tuneable consistency
- Avoiding contention on local increments
  - Store increments, not full value; merge on read + compaction
- Renewing counter id to deal with data loss
- Interaction with tombstones

(What about version vectors?)

- Version vectors allow detecting conflict, but do not give enough information to resolve it except in special cases
  - In the counters case, we'd need to hold onto all previous versions until we can be sure no new conflict with them can occur
  - Jeff Darcy has a longer discussion at http://pl.atyp.us/wordpress/?p=2601

Performance



A single four-core machine; one million inserts + one million updates

DataStax

Dealing with the JVM

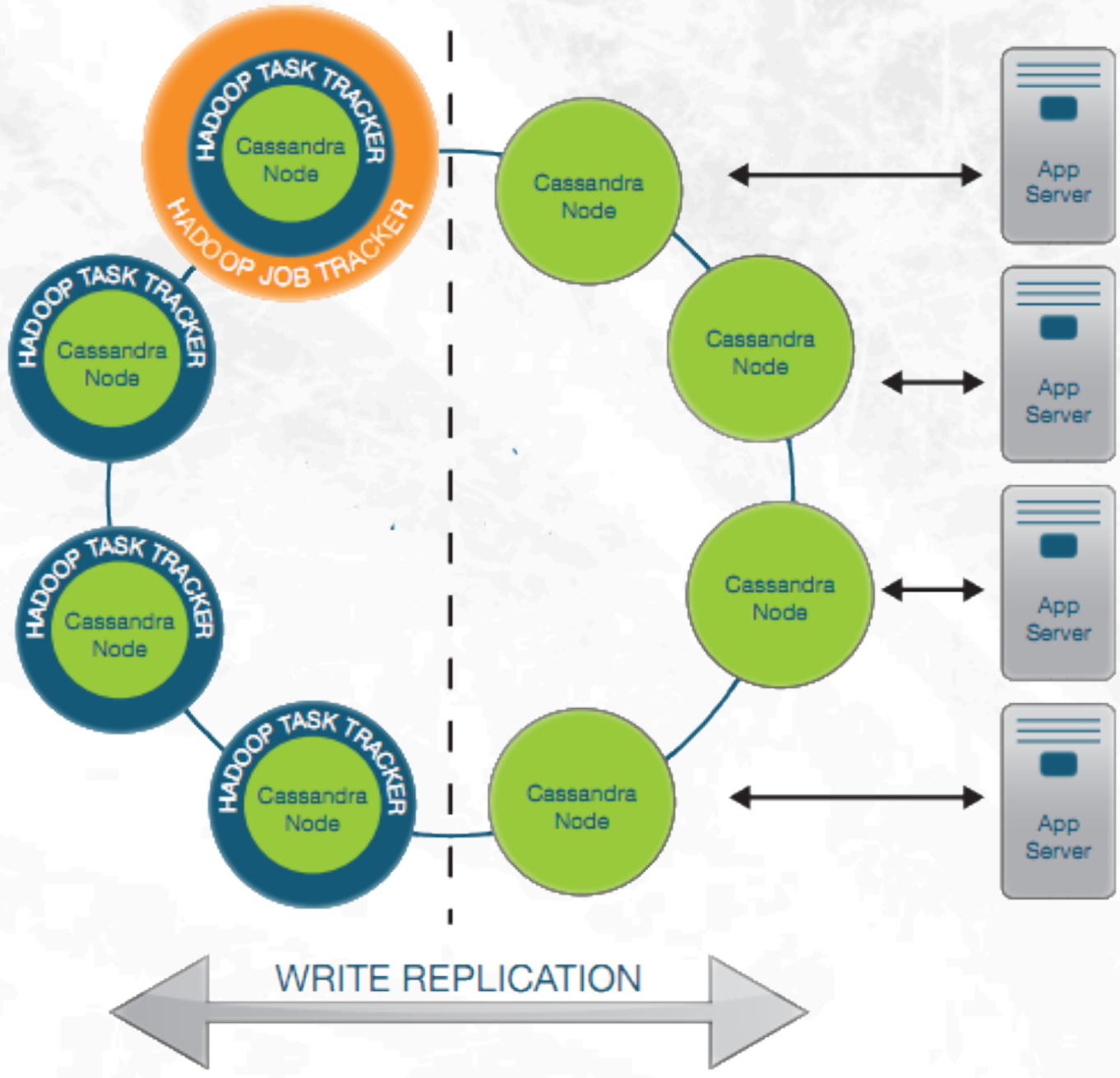
- JNA
  - `mlockall()`
  - `posix_fadvise()`
  - `link()`
- Memory
  - Move cache off-heap
  - In-heap arena allocation for memtables, bloom filters
  - Move compaction to a separate process?

DataStax

The Cassandra ecosystem

- Replication into Cassandra
  - Gigaspaces
  - Drizzle
- Solandra: Cassandra + Solr search
- DataStax Enterprise: Cassandra + Hadoop analytics

DataStax Enterprise



DataStax

Operations

- "Vanilla" Hadoop
  - 8+ services to setup, monitor, backup, and recover
    (NameNode, SecondaryNameNode, DataNode, JobTracker, TaskTracker, Zookeeper, Metastore, ...)
  - Single points of failure
  - Can't separate online and offline processing

- DataStax Enterprise
  - Single, simplified component
  - Peer to peer
  - JobTracker failover
  - No additional cassandra config

What's next

- **Ease Of Use**
- CQL: "Native" transport, prepared statements
- Triggers
- Entity groups
- Smarter range queries enabling Hive predicate push-down
- Blue sky: streaming / CEP

Questions?

- jbellis@datastax.com

- DataStax is hiring!
  - ~15 engineers (35 employees), want to double in 2012
    - Austin, Burlingame, NYC, France, Japan, Belarus
  - 100+ customers
  - $11M Series B funding