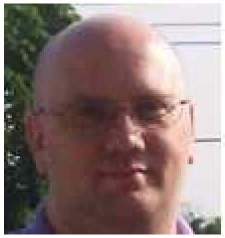
BlueRunner: Building an Email Service in the Cloud

Jun Rao
IBM Almaden Research Center

Apache Cassandra Committer

The Team

User Systems and Experience Research



Jerald Schoudt

Stefan Nusser

Jan Pieper

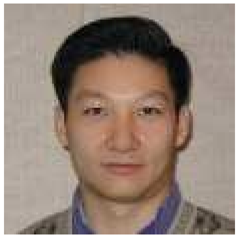
Hernan Badenes

Julian Cerruti

Information Management Research

Sandeep Tata

Bo Shekita



Jun Rao

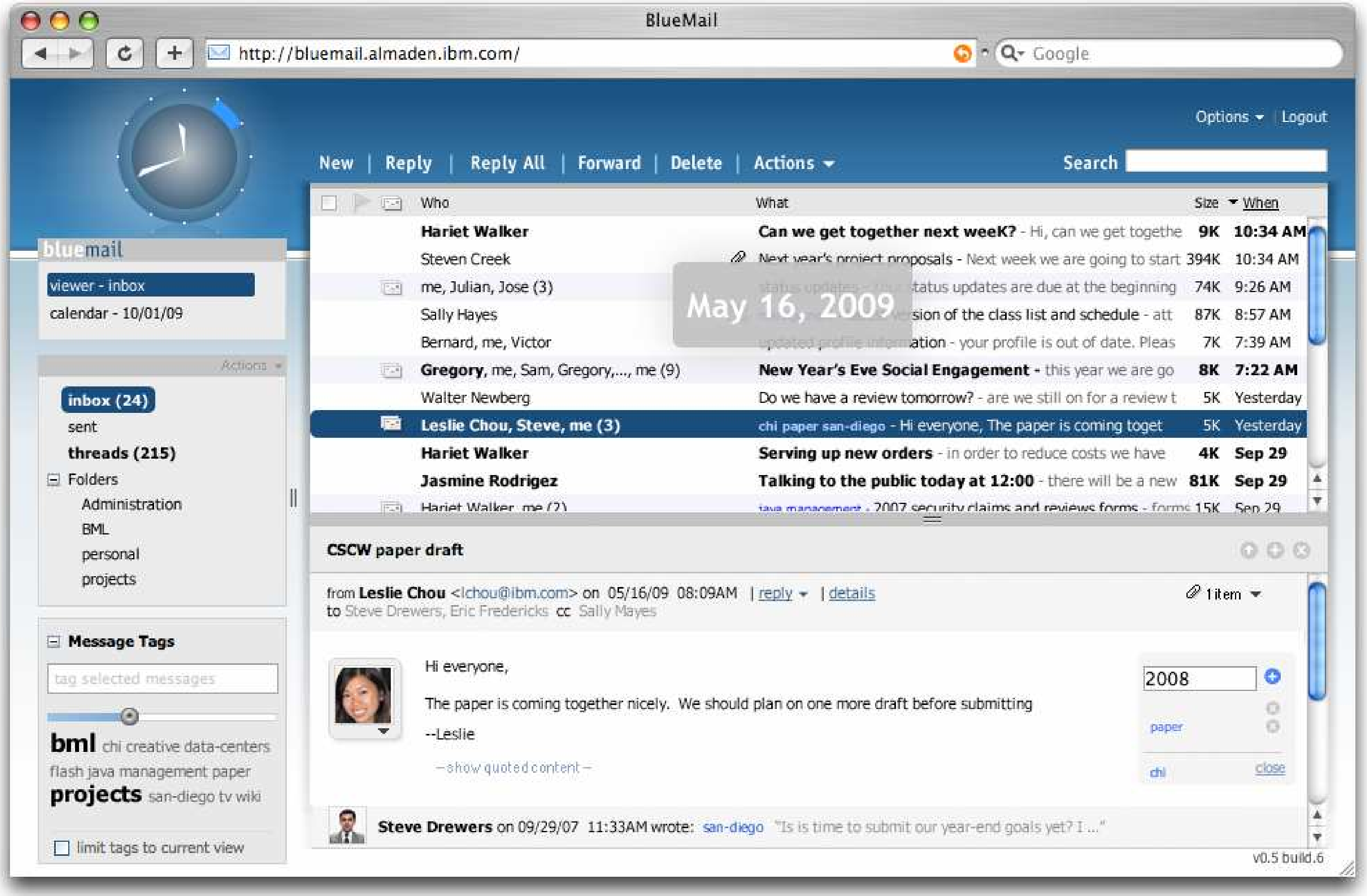
Outline

- BlueRunner overview
- Scalable Row Stores
- BlueRunner design in Cassandra
- Preliminary performance results
- Summary

What’s BlueRunner

- Research prototype for hosted emails at IBM ARC
- Browser-based email client + Cassandra backend
  - 3+ years on the client
  - ~1 year on the Cassandra backend
- Many advanced features in client
  - Scrolling
  - Foldering/Tagging
  - Sorting/Pivoting
  - Threading
  - Orienteering and Usability Improvements
- Backend
  - Thin client; most operations pushed to the backend
  - Designed for large-scale hosted environment
    - 100K mailboxes, each with 100K messages

How does the client look?



Who	What	Size	When
Hariet Walker	**Can we get together next weeK?** - Hi, can we get togethe	**9K**	**10:34 AM**
Steven Creek	Next year's project proposals - Next week we are going to start	394K	10:34 AM
me, Julian, Jose (3)	status updates - your status updates are due at the beginning	74K	9:26 AM
Sally Hayes	[illegible] version of the class list and schedule - att	87K	8:57 AM
Bernard, me, Victor	updated profile information - your profile is out of date. Pleas	7K	7:39 AM
Gregory, me, Sam, Gregory,..., me (9)	**New Year's Eve Social Engagement -** this year we are go	**8K**	**7:22 AM**
Walter Newberg	Do we have a review tomorrow? - are we still on for a review t	5K	Yesterday
Leslie Chou, Steve, me (3)	chi paper san-diego - Hi everyone, The paper is coming toget	5K	Yesterday
Hariet Walker	**Serving up new orders** - in order to reduce costs we have	**4K**	**Sep 29**
Jasmine Rodrigez	**Talking to the public today at 12:00** - there will be a new	**81K**	**Sep 29**
Hariet Walker, me (2)	java management - 2007 security claims and reviews forms - forms	15K	Sep 29

CSCW paper draft

from **Leslie Chou** <lchou@ibm.com> on 05/16/09 08:09AM | reply | details
to Steve Drewers, Eric Fredericks cc Sally Mayes

Hi everyone,

The paper is coming together nicely. We should plan on one more draft before submitting

--Leslie

– show quoted content –

Steve Drewers on 09/29/07 11:33AM wrote: san-diego "Is is time to submit our year-end goals yet? I ..."

Why a new backend?

Limitations in Traditional DBs	
Fault-tolerance relies on expensive reliable storage	limiting scalability
Weak elasticity--- hard to grow a cluster incrementally	
No automatic load-balancing	
Rigid relational schema	mismatch for many apps
No versioning support	

The Cloud Landscape for Scalable Backend

LinkedIn
Google App Engine
apache CouchDB relax
facebook
Y!
PNUTS
amazon web services™
Amazon SimpleDB™ BETA
Sun
APACHE
mongoDB
{name: "mongo", type: "db"}

Voldemort

BigTable

Cassandra

Drizzle

HBase

- **Flurry of activity in this space motivated by**
  - RDBMS too rigid/heavy for some apps
  - Existing RDBMS engines missing many key cloud requirements

What is a Scalable Row Store?

- Middle ground btw a DBMS and a file system
  - Much simpler API then SQL
  - Designed to scale

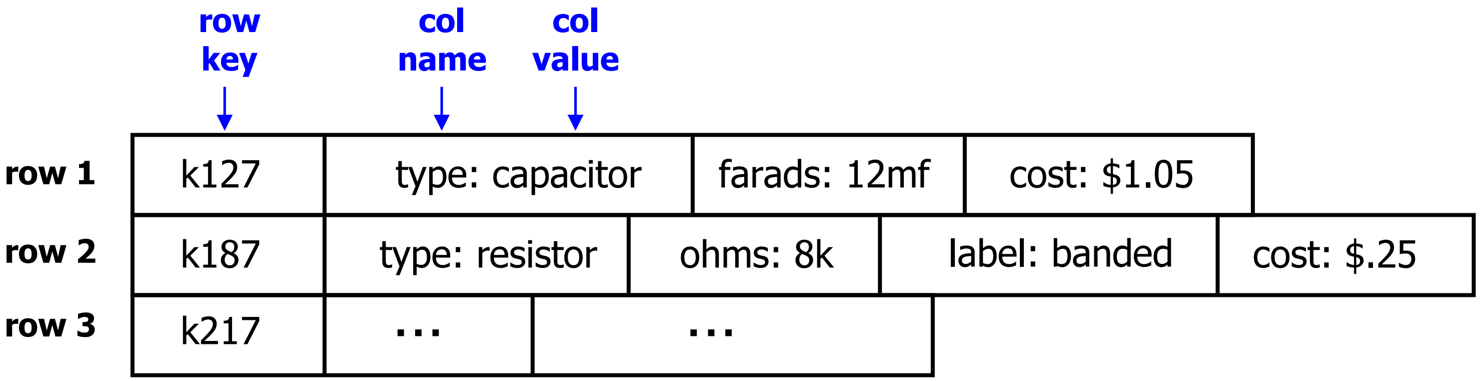
Limitations in Traditional DBs	**Scalable Row Stores**
Fault-tolerance relies on expensive reliable storage	Fault-tolerance done in software; replication on commodity disks
Weak elasticity--- hard to grow a cluster incrementally	Can grow a cluster incrementally and online
No automatic load-balancing	Built-in automatic load-balancing
Rigid relational schema	No strong schema required
No versioning support	Built-in versioning

Cassandra

- Google’s Bigtable data model + Amazon’s Dynamo scalable architecture
- Developed by Facebook in 2007
  - Used in production for a few apps (e.g., inbox search for 200M users)
- Became an Apache Incubator project early 2009
  - active community
  - additional committers from Rackspace and IBM
  - contributors from Digg, Twitter, etc

Cassandra Data Model

- Familiar relational tables, rows, and columns, but more flexible
  - No upfront schema required
  - New columns can be added any time and columns can vary from row to row



row key

col name

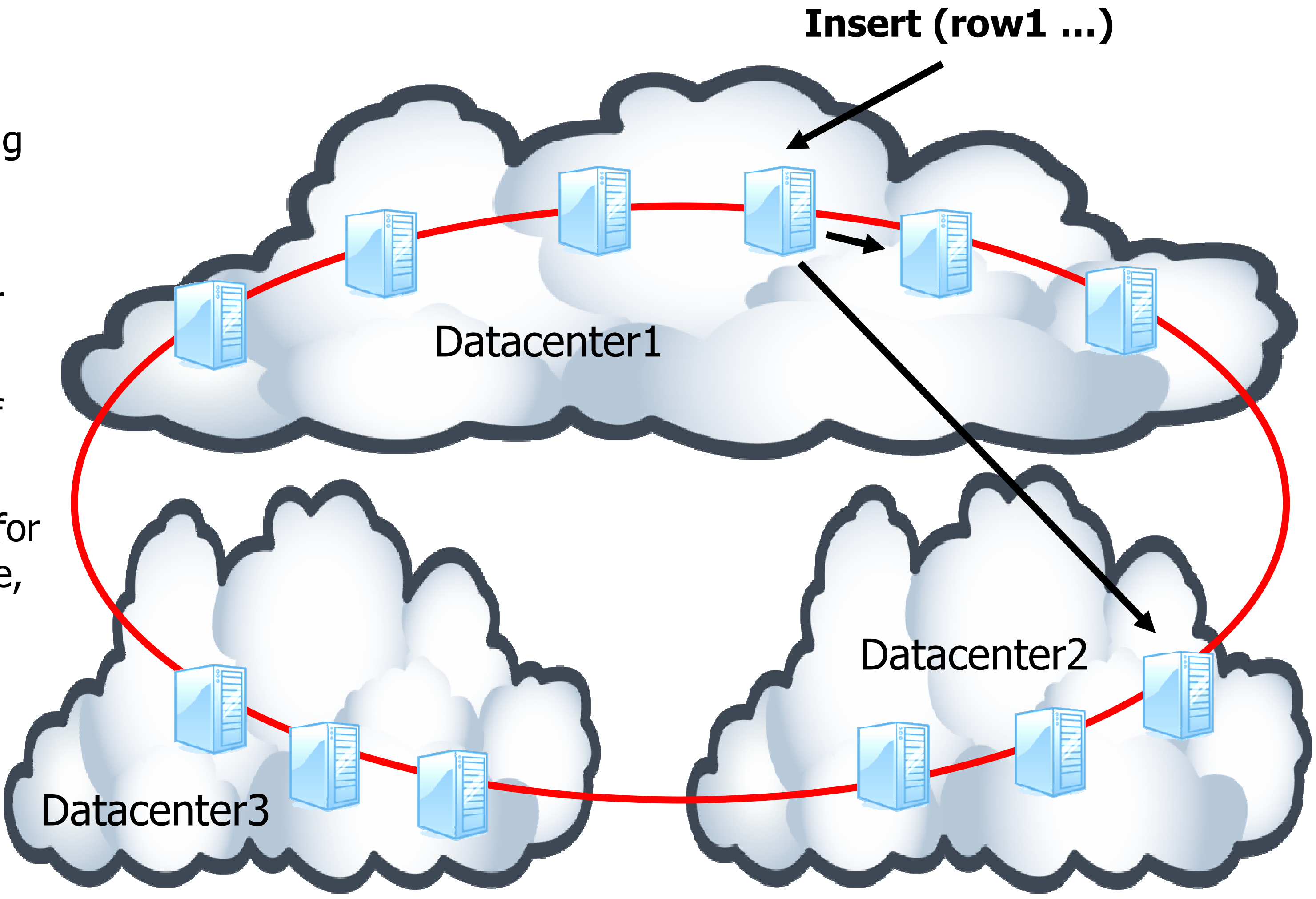
col value

row 1	k127	type: capacitor	farads: 12mf	cost: $1.05	
row 2	k187	type: resistor	ohms: 8k	label: banded	cost: $.25
row 3	k217	...	...		

- Columns grouped into Column Families
  - Column families are stored separately (like vertical partitioning)

Cassandra Distributed Architecture

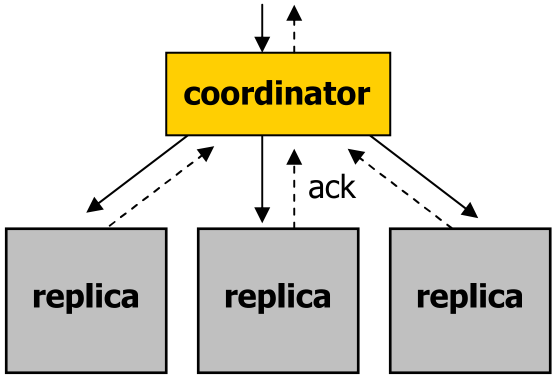
- Based on DHT ring
- Replication cross-rack and cross-datacenter (sync or async)
- No single point of failure
- Gossip protocols for membership, failure, DHT map, etc



Scale, Fault-tolerance, Elasticity, Low-cost

Eventual Consistency

- **CAP Theorem** [Brewer00]
  - Can only get 2 of **C**onsistency, **A**vailability, or **P**artition tolerance
- Cassandra relaxes **C** to eventual consistency
  - Emphasis is on performance and availability
  - Allow concurrent read/write to any replica – latest write wins on conflict
- **Knobs to tradeoff consistency and performance**
  - Writes are sent to all N replicas in parallel
  - Can choose to read from **R** replicas and wait for **W** acks for writes
  - Tune R and W to 1,2,3…,N for latency requirements



coordinator acts as a
simple state machine

Email Schema in Cassandra

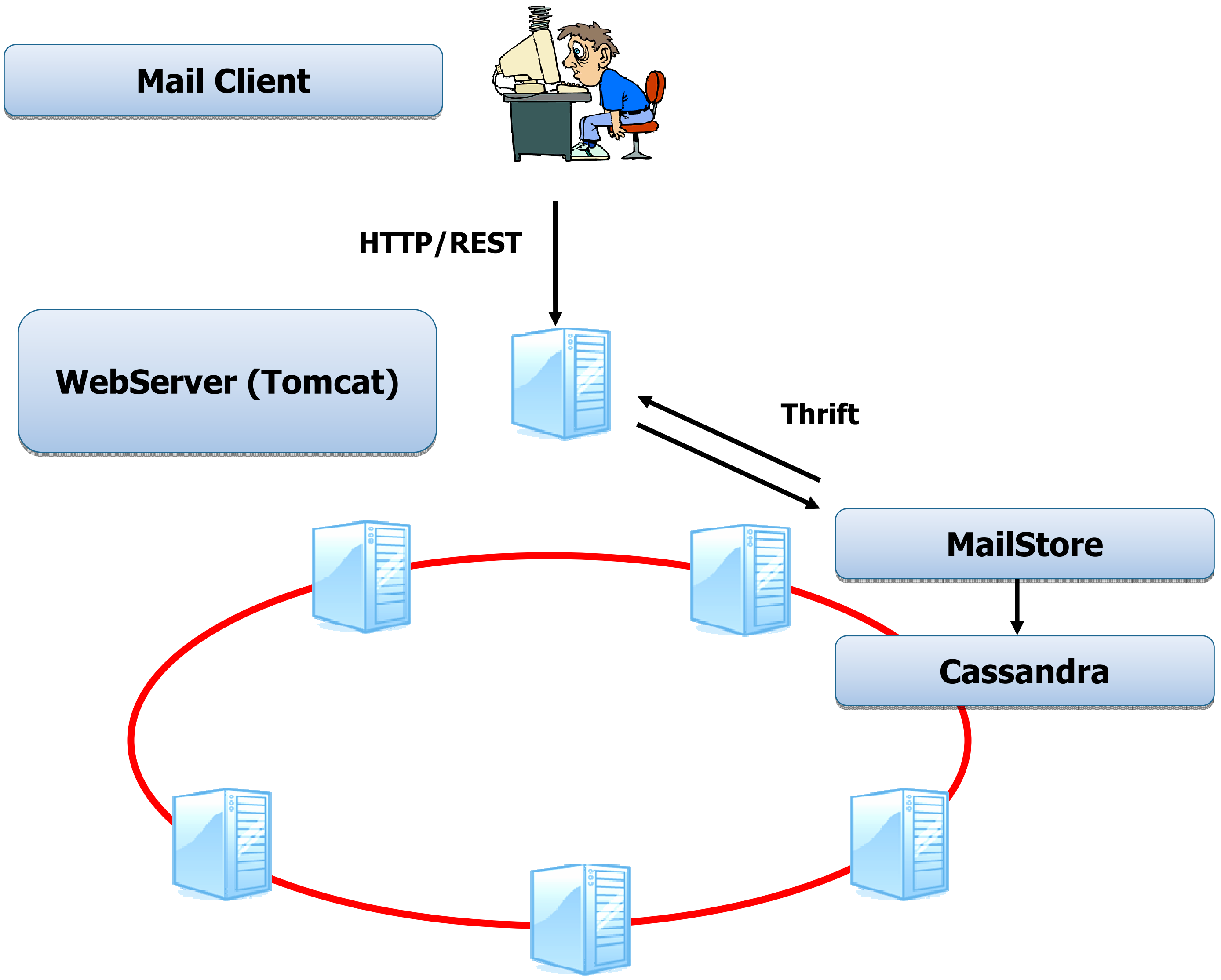
- Row key - User id
  - All data for one user is on a single node
  - Currently no sharing of messages across users
- Column Families
  - MailList - Message id : full message
  - HeaderList - Message id : message headers + metadata
  - CollectionIndex - Collection id + sort key : message id

    e.g., Inbox/Date/2009-07-10-14:20:56 : message1000
    Inbox/Sender/Mike Brown : message1000
  - Others
    - CollectionMetadata, ThreadList, ThreadIndex
- Full message stored separately from index and metadata
- Data format - JSON

Typical Operations

- Cassandra APIs used
  - get_column(row, CF, column)
  - get_columns(row, CF, columns[])
  - get_slice(row, CF, startColumn, asc/desc, count)
  - Efficient with row/column index support in Cassandra
- ListMessages
  - get_slice(Jun, CollectionIndex, Inbox/Date/current_date/, desc, 50) to obtain the first 50 messageIDs in Inbox
  - get_columns(Jun, HeaderList, messageID[])
- GetMessage
  - get_column(Jun, MailList, messageID)
- SortMessages by Sender
  - get_slice(Jun, CollectionIndex, Inbox/Sender//, asc, 50)
  - get_columns(Jun, HeaderList, messageID[])

BlueRunner Deployment



Mail Client

HTTP/REST

WebServer (Tomcat)

Thrift

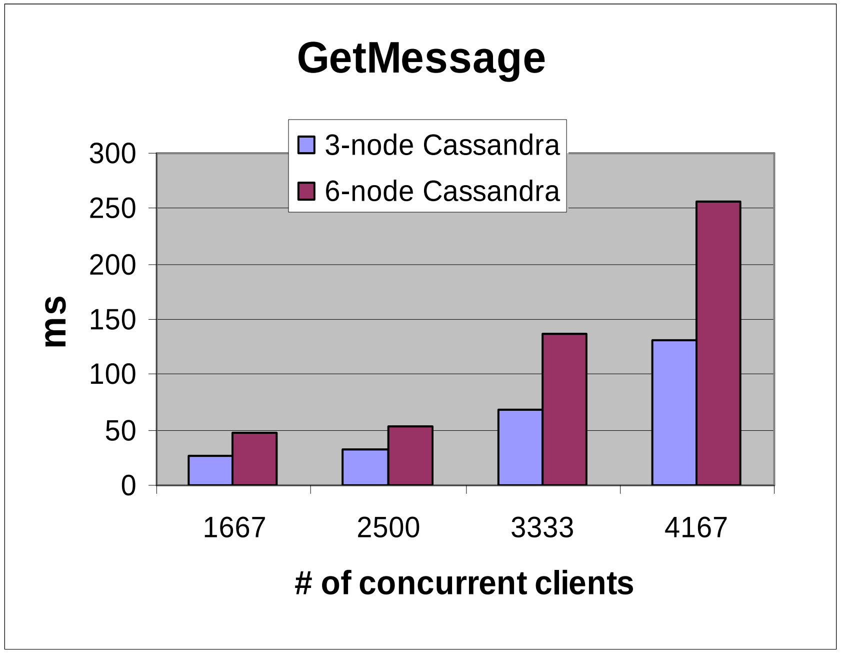
MailStore

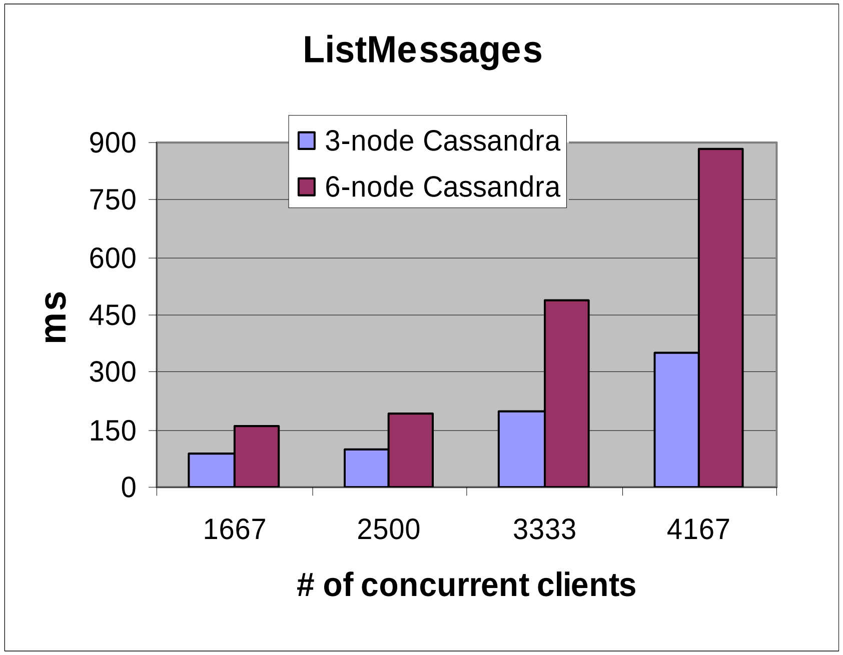
Cassandra

Experimental Setup

- 6-node cluster and each node with
  - 2 quad-core CPUs
  - 16 GB memory
  - 5 SATA disks (1 for Cassandra commit log and 4 for data)
- Data
  - generated 1800 mailboxes per node
    - 250-16K messages per mailbox
    - ~50GB w/o replication
  - Cassandra replication set to 2
- Workload
  - Varying # of concurrent clients from 1600 to 4100 per node
  - Each client repeatedly
    - opens up an inbox
    - looks at a few message
    - go to sleep for a while

Preliminary Result (median response time in ms)





- Able to sustain 2500 concurrent clients per node with reasonable response time
  - average: ~100 requests per sec per node

Summary

- Cassandra-based backend very promising
  - Enabling scalability, availability, and elasticity
  - Flexible data model a good fit
- Future work
  - Many places can be improved
    - Alternative schema design
    - Secondary index/full-text index support
    - Enabling MapReduce-based analytics on the backend
  - Other potential collaborative apps on Cassandra
  - Research on better reasoning btw consistency and availability