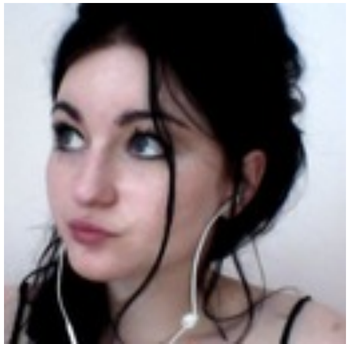
Dynamo:

Theme and Variations

@shanley



Riak

Dynamo: Amazon's Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels

Amazon.com

ABSTRACT

Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters around the world. At this scale, small and large components fail continuously and the way persistent state is managed in the face of these failures drives the reliability and scalability of the software systems.

This paper presents the design and implementation of Dynamo, a highly available key-value storage system that some of Amazon's core services use to provide an "always-on" experience. To achieve this level of availability, Dynamo sacrifices consistency under certain failure scenarios. It makes extensive use of object versioning and application-assisted conflict resolution in a manner that provides a novel interface for developers to use.

Categories and Subject Descriptors

D.4.2 [**Operating Systems**]: Storage Management; D.4.5 [**Operating Systems**]: Reliability; D.4.2 [**Operating Systems**]: Performance;

General Terms

Algorithms, Management, Measurement, Performance, Design, Reliability.

1. INTRODUCTION

Amazon runs a world-wide e-commerce platform that serves tens of millions customers at peak times using tens of thousands of servers located in many data centers around the world. There are strict operational requirements on Amazon's platform in terms of performance, reliability and efficiency, and to support continuous growth the platform needs to be highly scalable. Reliability is one of the most important requirements because even the slightest outage has significant financial consequences and impacts customer trust. In addition, to support continuous growth, the

One of the lessons our organization has learned from operating Amazon's platform is that the reliability and scalability of a system is dependent on how its application state is managed. Amazon uses a highly decentralized, loosely coupled, service oriented architecture consisting of hundreds of services. In this environment there is a particular need for storage technologies that are always available. For example, customers should be able to view and add items to their shopping cart even if disks are failing, network routes are flapping, or data centers are being destroyed by tornados. Therefore, the service responsible for managing shopping carts requires that it can always write to and read from its data store, and that its data needs to be available across multiple data centers.

Dealing with failures in an infrastructure comprised of millions of components is our standard mode of operation; there are always a small but significant number of server and network components that are failing at any given time. As such Amazon's software systems need to be constructed in a manner that treats failure handling as the normal case without impacting availability or performance.

To meet the reliability and scaling needs, Amazon has developed a number of storage technologies, of which the Amazon Simple Storage Service (also available outside of Amazon and known as Amazon S3), is probably the best known. This paper presents the design and implementation of Dynamo, another highly available and scalable distributed data store built for Amazon's platform. Dynamo is used to manage the state of services that have very high reliability requirements and need tight control over the tradeoffs between availability, consistency, cost-effectiveness and performance. Amazon's platform has a very diverse set of applications with different storage requirements. A select set of applications requires a storage technology that is flexible enough to let application designers configure their data store appropriately based on these tradeoffs to achieve high availability and guaranteed performance in the most cost effective manner.

There are many services on Amazon's platform that only need primary-key access to a data store. For many services, such as those that provide best seller lists, shopping carts, customer amazon

Shanley's Amazon.com | Today's Deals | Gift Cards | Help

Shop by Department

Search All Go

Hello, Sha
Your Ac

Today's Deals | Gold Box | Black Friday | Cyber Monday | Top Holiday Deals | All Deals | Coupons | Outlet | Deals & Bargains | Warehouse Deals | Digital Deals

PHILIPS NORELCO All the tools you need from head to toe. HEAD FACE BODY

Gold Box Event: Countdown to Black Friday

New Deals. Every Day.

Stay connected via e-mail, Facebo

Deal of the Day

Nikon COOLPIX S9200 Digital Camera

Power and portability blend perfectly into the COOLPIX S9200. Photograph everything from the nightlife in Cancun to the vibrant details of a flower market with the S9200's 18x wide-angle zoom. Create stunning low-light photos without a flash, thanks to its 16.0 megapixel CMOS sensor. Shoot with high-speed framing rates,.... **read more**

List Price:	~~$299.95~~
Yesterday's Price:	$299.00
Today's Discount:	- $130.00
Gold Box Price:	$169.00 (44% off)

13 Comments | (50)

Share

Prime | Add to Cart

Lightning Deals

10 - 13 of 1

3:00 PM PDT - Jewelry Deal

4:00 PM PDT - Jewelry Deal All

5:00 PM PDT - Clothing Deal

ExOfficio Men's Give-a

List Price:	~~$26.0~~
Amazon's Price:	$16.2
Deal Price:	$13.8

Comments | (124)

Select options to see Lightni availability.

24% now claimed | Select options | Prim

01:07:48 remaining

9:00 PM PDT - Upcoming Deal

COUNTDOWN TO

Black Friday Deals Week

16 : 04 : 09 : 37
Days Hours Mins Secs

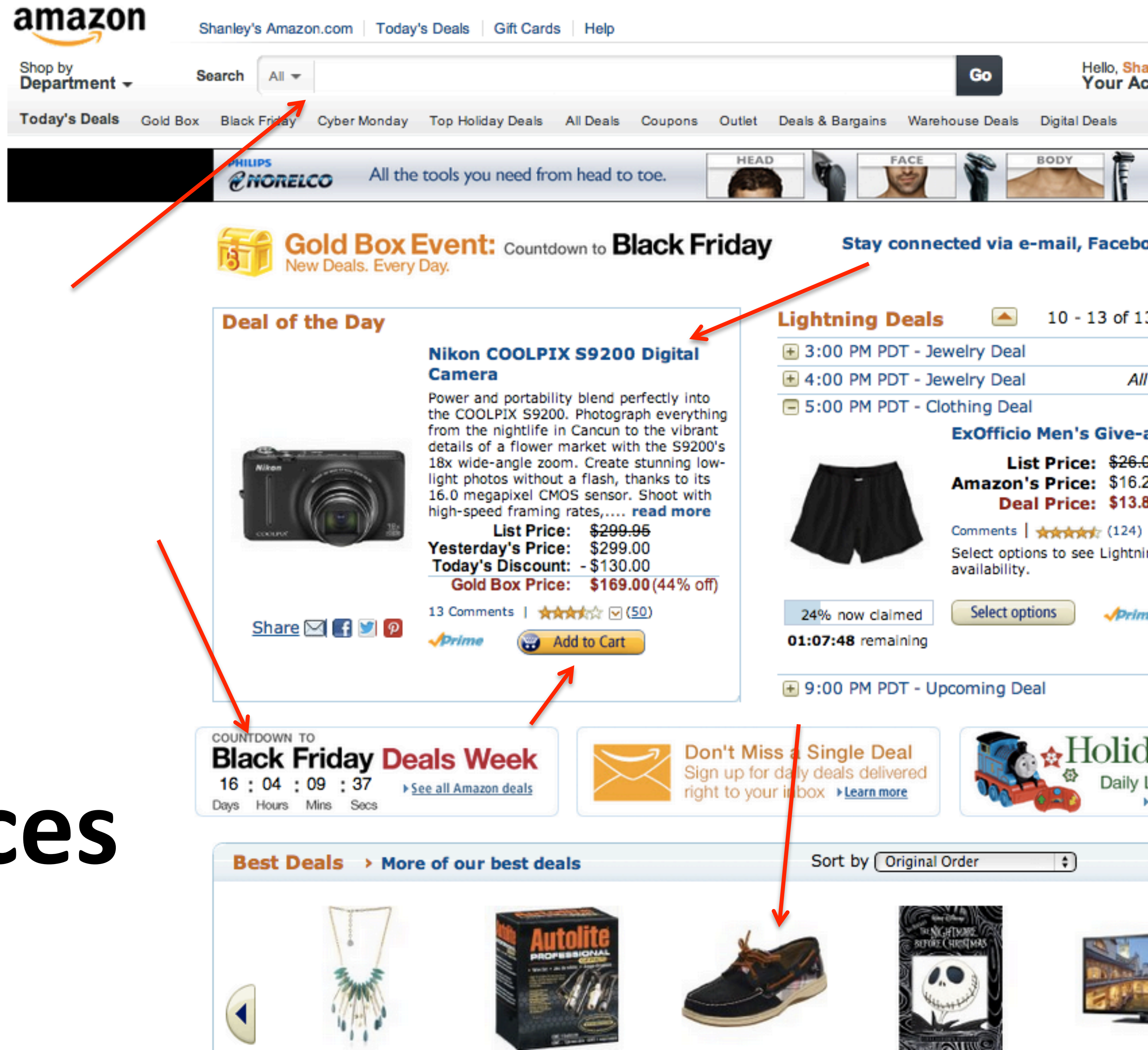
See all Amazon deals

Don't Miss a Single Deal
Sign up for daily deals delivered right to your inbox Learn more

Holid
Daily

Best Deals › More of our best deals

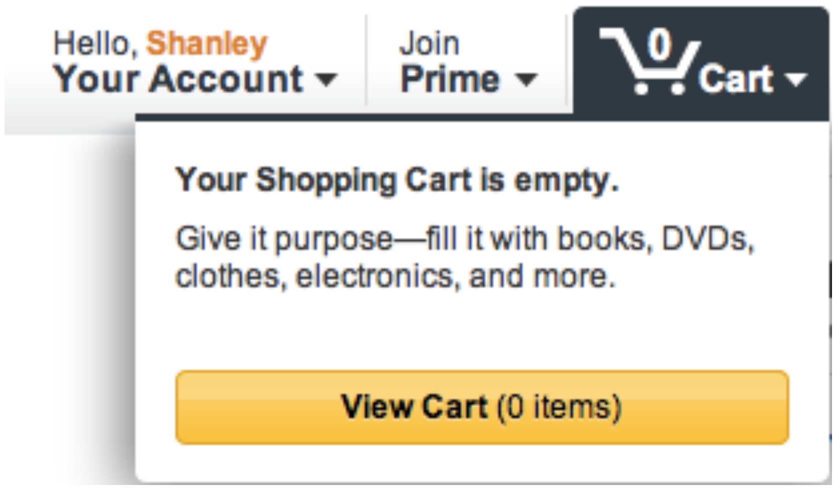
Sort by Original Order



150 Services

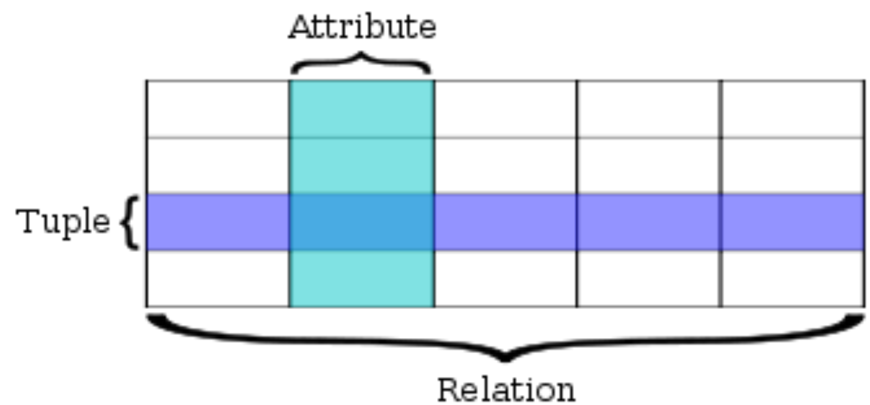
Cassandra

riak


Hello, Shanley
Your Account
Join
Prime
0
Cart
Your Shopping Cart is empty.
Give it purpose—fill it with books, DVDs, clothes, electronics, and more.
View Cart (0 items)

- Global access
- Multiple machines
- Multiple datacenters
- Scale to peak loads easily
- Continuous failure

Sun
ORACLE
EXALOGIC
ELASTIC CLOUD
X



Attribute
Tuple
Relation

Traditionally production systems store their state in relational databases. For many of the more common usage patterns of state persistence, however, a relational database is a solution that is far from ideal.

Most of these services only store and retrieve data by primary key and do not require the complex querying and management functionality offered by an RDBMS.

This excess functionality requires expensive hardware and highly skilled personnel for its operation, making it a very inefficient solution.

In addition, the available replication technologies are limited and typically choose consistency over availability.

Although many advances have been made in the recent years, it is still not easy to scale-out databases or use smart partitioning schemes for load balancing.

Dynamo: Amazon's Highly Available Key-value Store

CAP Theorem

People tend to focus on consistency/availability as the sole driver of emerging database models because it provides a simple and academic explanation for more complex evolutionary factors. In fact, CAP Theorem, according to its original author, "prohibits only a tiny part of the design space: perfect availability and consistency in the presence of partitions, which are rare… there is little reason to forfeit C or A when the system is not partitioned." **In reality, a much larger range of considerations and tradeoffs have informed the "NoSQL" movement…**

Traditionally production systems store their state in relational databases. For many of the more common usage patterns of state persistence, however, a relational database is a solution that is far from ideal.

Most of these services only store and retrieve data by primary key and do not require the complex querying and management functionality offered by an RDBMS.

This excess functionality requires expensive hardware and highly skilled personnel for its operation, making it a very inefficient solution.

In addition, the available replication technologies are limited and typically choose consistency over availability.

Although many advances have been made in the recent years, it is still not easy to scale-out databases or use smart partitioning schemes for load balancing.

Dynamo: Amazon's Highly Available Key-value Store

Spanner is Google's scalable, multi-version, globally- distributed, and synchronously-replicated database... It is the first system to distribute data at global scale and support externally-consistent distributed transactions...

Spanner is designed to scale up to millions of machines across hundreds of datacenters and trillions of database rows... Spanner's main focus is managing cross-datacenter replicated data...

Spanner started... as part of a rewrite of Google's advertising backend called F1 [35]. This backend was originally based on a MySQL database...

Resharding this revenue-critical database as it grew in the number of customers and their data was extremely costly. The last resharding took over two years of intense effort...

Spanner: Google's Globally-Distributed Database

Shanley's Theorem

Database design is driven by a virtuous tension between the requirements of the app, the profile of developer productivity, and the limitations of the operational scenario.

requirements of the app

- Stringent latency requirements measured at the 99.9% percentile
- Highly available
- Always writeable
- Modeled as keys/values

the profile of developer productivity

- Choice to manage conflict resolution themselves or manage on the data store level
- Simple, primary-key only interface
- No need for relational data model

- Functions on commodity hardware
- Each object must be replicated across multiple DCs
- Can scale out one node at a time with minimal impact on system and operators

limitations of the operational scenario.


Bull

requirements of the app

- 1995: Less than 40 million internet users; now: 2.4 billion
- Latency perceived as unavailability
- New types of applications

the profile of developer productivity

- Much more data
- Unstructured data
- New kinds of business requirements
- App scales gracefully without high development overheard

- Scale-out design on less expensive hardware
- Ability to easily meet peak loads
- Run efficiently across multiple sites
- Low operational burden

limitations of the operational scenario

Aspects of the database:

- How to distribute data around the cluster
- Adding new nodes
- Replicating data
- Resolving data conflicts
- Dealing with failure scenarios
- Data model

how to distribute data around the cluster

how to distribute data around the cluster

Bunny Names A-G

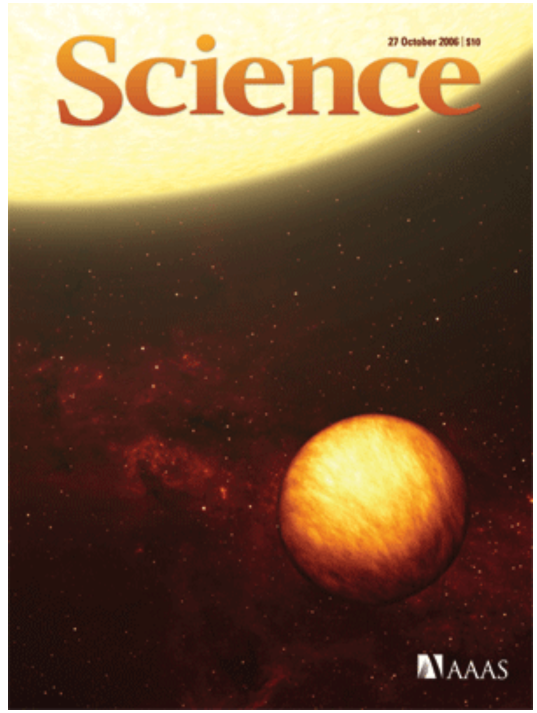
Bunny Names H-R

Bunny Names R-Z

how to distribute data around the cluster

Bunny Names Disproportionately Trend Towards Bunny, Cuddles, Fluffy, Mr. Bunny, Peter Rabbit, Velveteen, Peter Cottontail, and Mitten

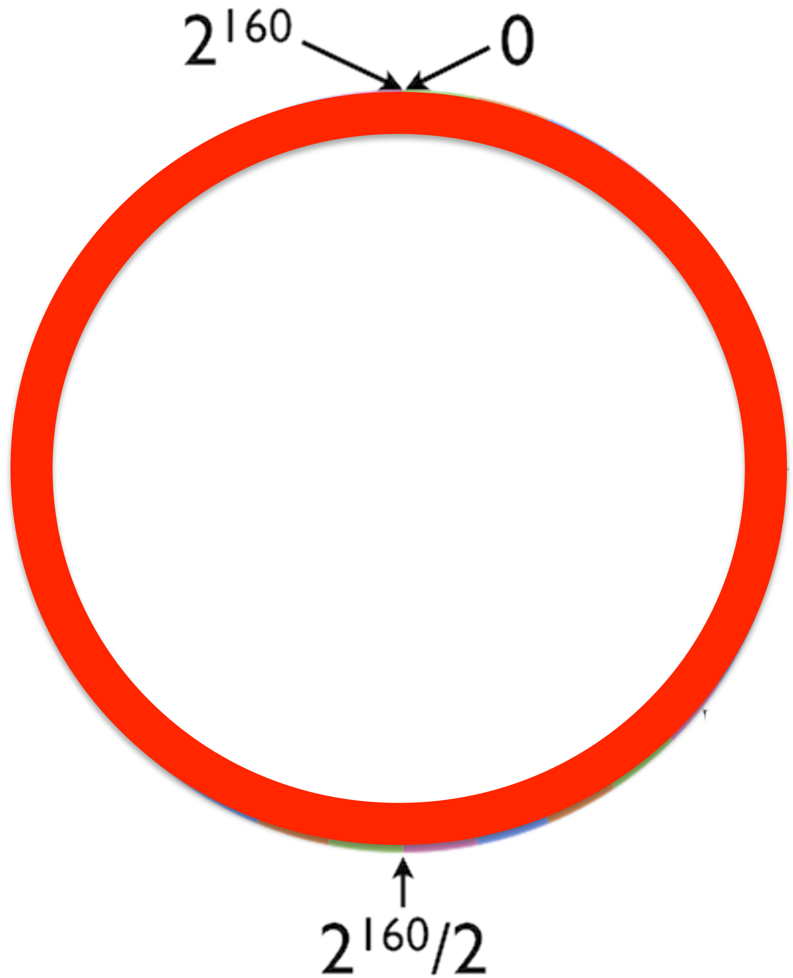
how to distribute data around the cluster



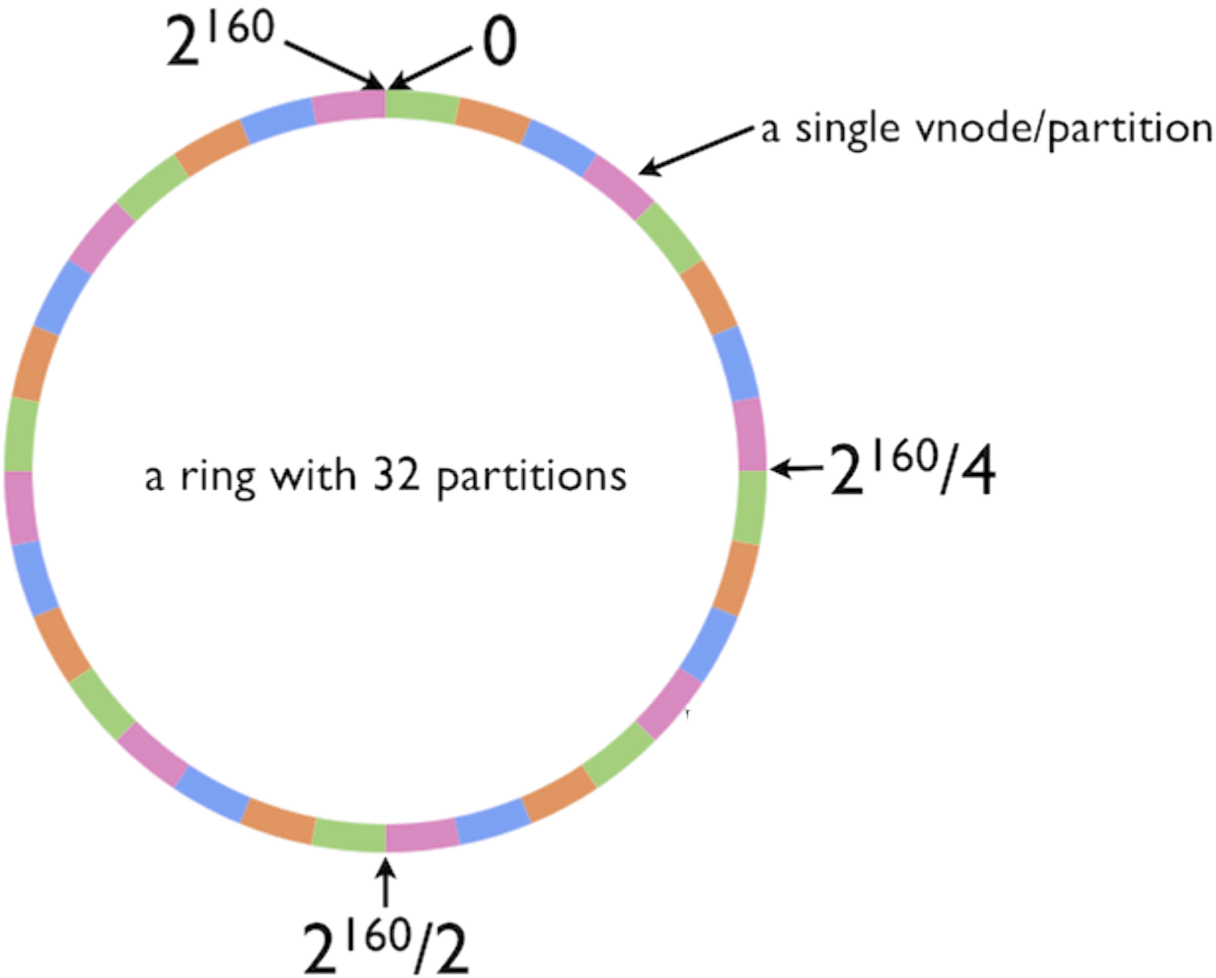
how to distribute data around the cluster

- Reduce risk of hot spots in the database
- Data is automatically assigned to nodes

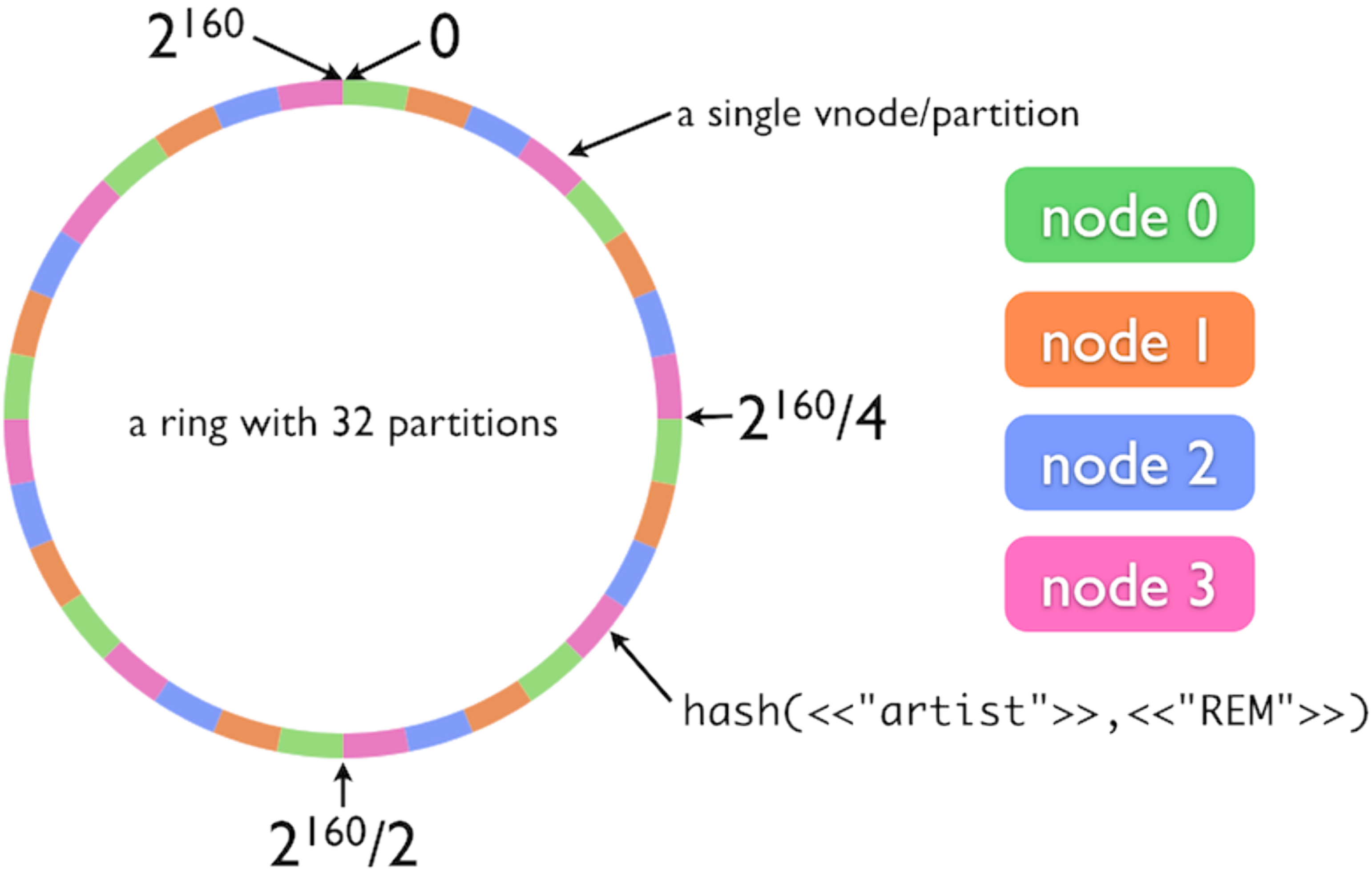
how to distribute data around the cluster



how to distribute data around the cluster



how to distribute data around the cluster



how to distribute data around the cluster

adding new nodes

addir nodes

Bunny Names A-G

adding new nodes

A-G

Bunny Names A-G

Bunny Names A-G

Bunny N

adding new nodes

of nodes that take special roles or extra set of responsibilities. In our experience, symmetry simplifies the process of system provisioning and maintenance.

Decentralization: An extension of symmetry, the design should favor decentralized peer-to-peer techniques over centralized control. In the past, centralized control has resulted in outages and the goal is to avoid it as much as possible. This leads to a simpler, more scalable, and more available system.

Heterogeneity: The system needs to be able to exploit heterogeneity in the infrastructure it runs on. e.g. the work distribution must be proportional to the capabilities of the individual servers. This is essential in adding new nodes with higher capacity without having to upgrade all hosts at once.

3. RELATED WORK

3.1 Peer to Peer Systems

There are several peer-to-peer (P2P) systems that have looked at the problem of data storage and distribution. The first generation of P2P systems, such as Freenet and Gnutella[1], were predominantly used as file sharing systems. These were examples of unstructured P2P networks where the overlay links between peers were established arbitrarily. In these networks, a search query is usually flooded through the network to find as many peers as possible that share the data. P2P systems evolved to the next generation into what is widely known as structured P2P networks. These networks employ a globally consistent protocol to ensure that any node can efficiently route a search query to some peer that has the desired data. Systems like Pastry [16] and Chord [20] use routing mechanisms to ensure that queries can be answered within a bounded number of hops. To reduce the additional latency introduced by multi-hop routing, some P2P systems (e.g., [14]) employ O(1) routing where each peer maintains enough routing information locally so that it can route requests (to access a data item) to the appropriate peer within a constant number of hops.

[1] http://freenetproject.org/, http://www.gnutella.org

Update conflicts are
resolution procedures
system that does not
achieves high availability and scalability using replication. The Google File System [6] is another distributed file system built for hosting the state of Google's internal applications. GFS uses a simple design with a single master server for hosting the entire metadata and where the data is split into chunks and stored in chunkservers. Bayou is a distributed relational database system that allows disconnected operations and provides eventual data consistency [21].

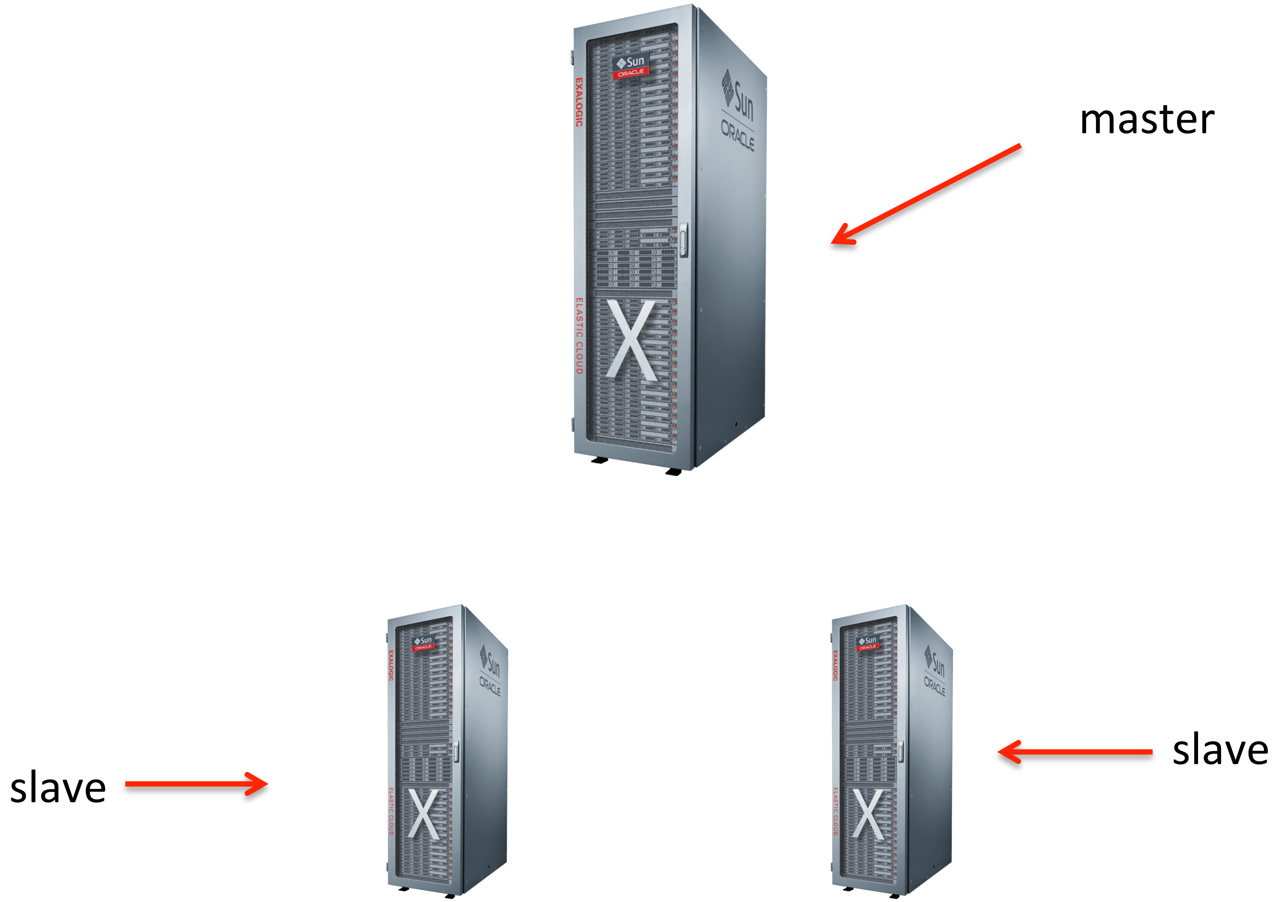
Among these systems, Bayou, Coda and Ficus allow disconnected operations and are resilient to issues such as network partitions and outages. These systems differ on their conflict resolution procedures. For instance, Coda and Ficus perform system level conflict resolution and Bayou allows application level resolution. All of them, however, guarantee eventual consistency. Similar to these systems, Dynamo allows read and write operations to continue even during network partitions and resolves updated conflicts using different conflict resolution mechanisms. Distributed block storage systems like FAB [18] split large size objects into smaller blocks and stores each block in a highly available manner. In comparison to these systems, a key-value store is more suitable in this case because: (a) it is intended to store relatively small objects (size < 1M) and (b) key-value stores are easier to configure on a per-application basis. Antiquity is a wide-area distributed storage system designed to handle multiple server failures [23]. It uses a secure log to preserve data integrity, replicates each log on multiple servers for durability, and uses Byzantine fault tolerance protocols to ensure data consistency. In contrast to Antiquity, Dynamo does not focus on the problem of data integrity and security and is built for a trusted environment. Bigtable is a distributed storage system for managing structured data. It maintains a sparse, multi-dimensional sorted map and allows applications to access their data using multiple attributes [2]. Compared to Bigtable, Dynamo targets applications that require only key/value ac
availability where updates a
network partitions or server f

adding new nodes

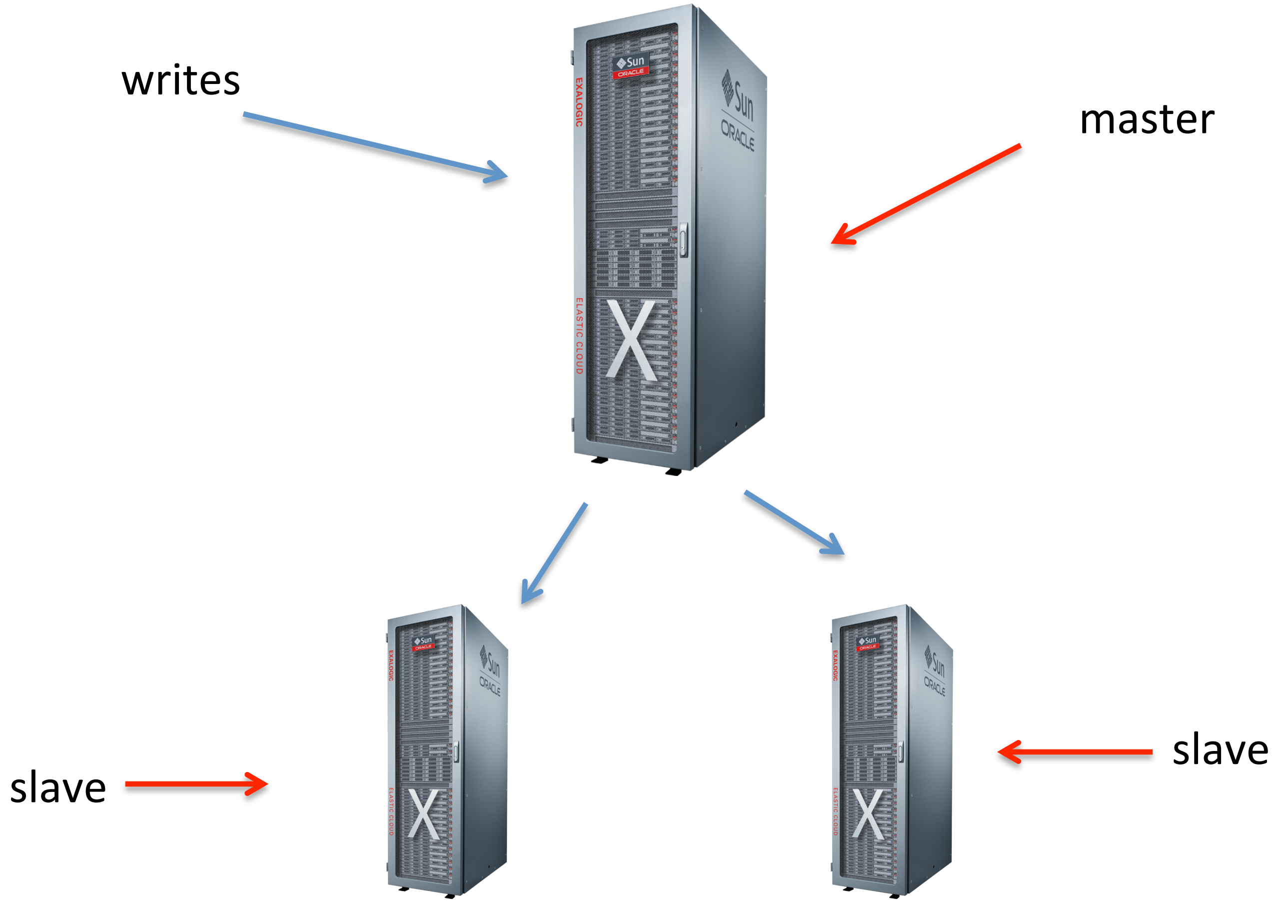
- “decoupling of partitioning and partition placement”

adding new nodes

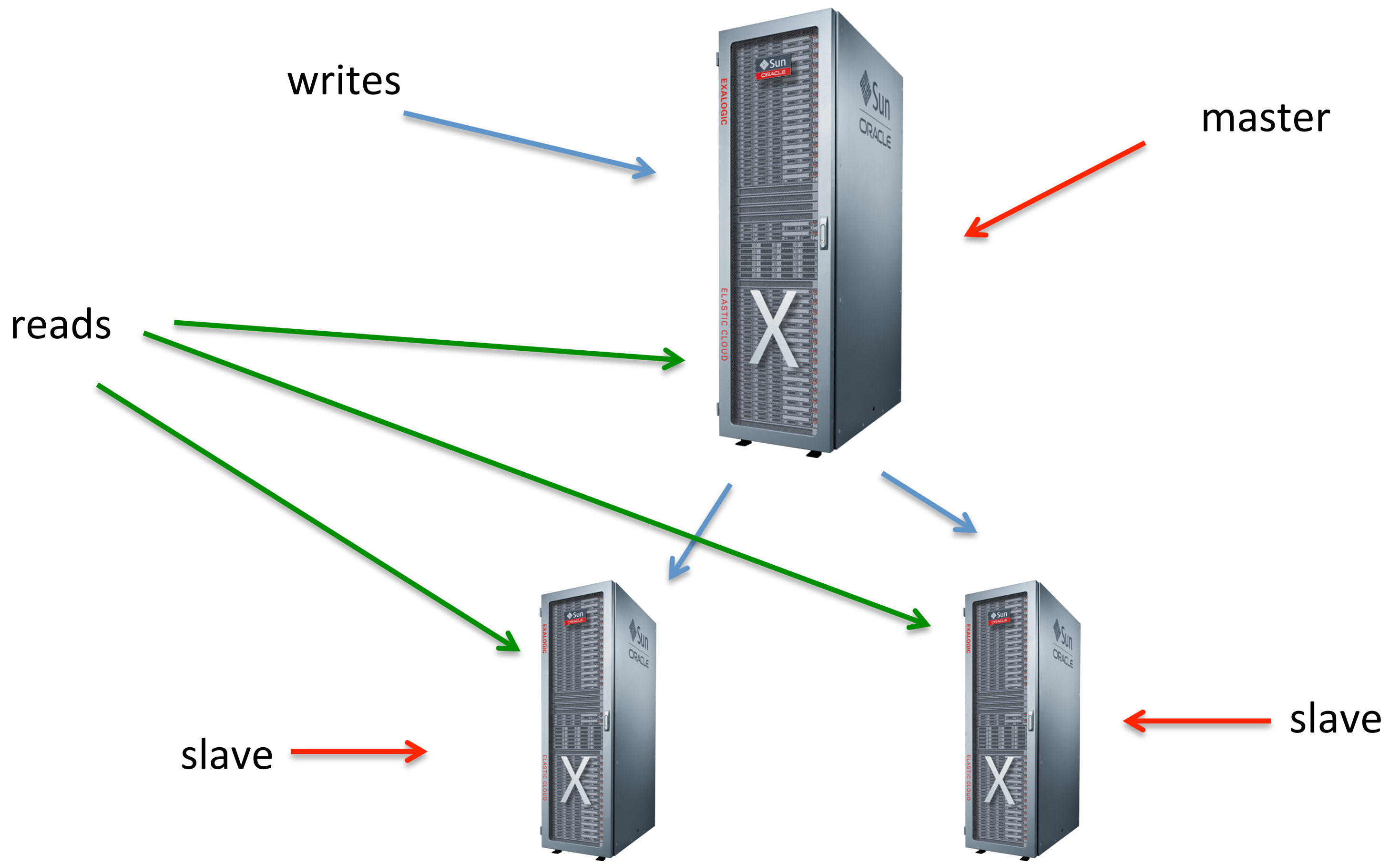
replicating data



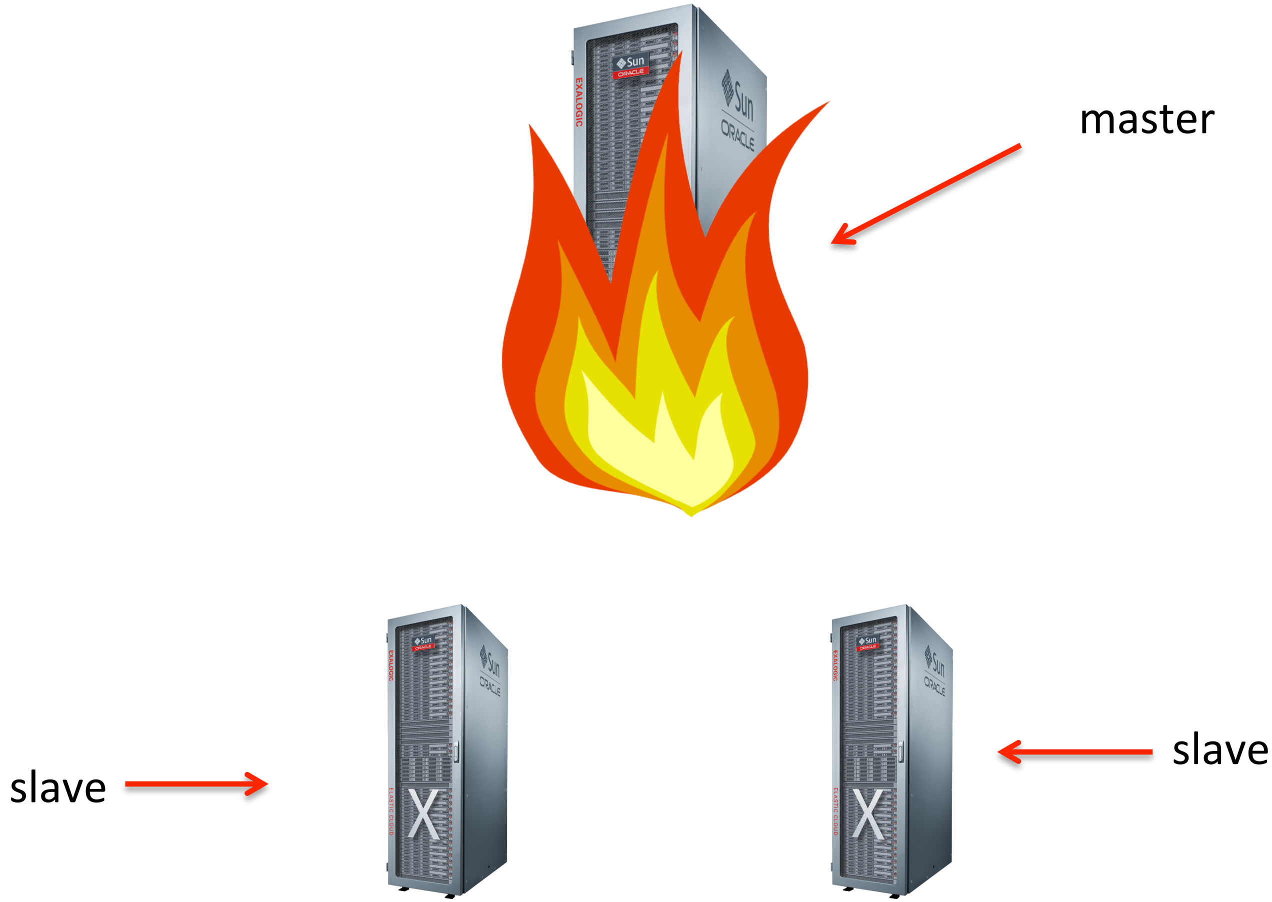
replicating data



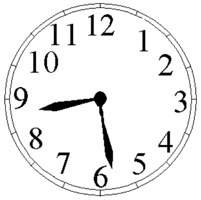
replicating data



replicating data

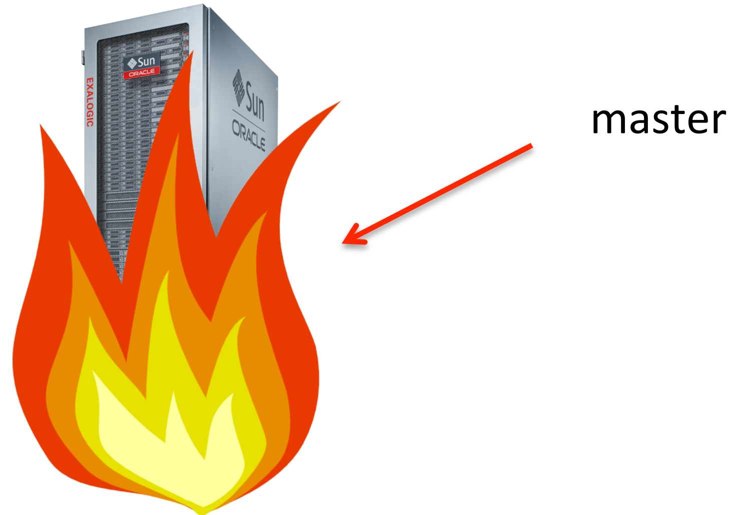


replicating data

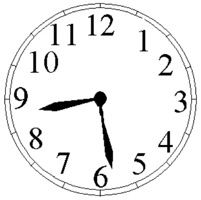


Availability Clock

1. Time to figure out the master is gone
2. Master election



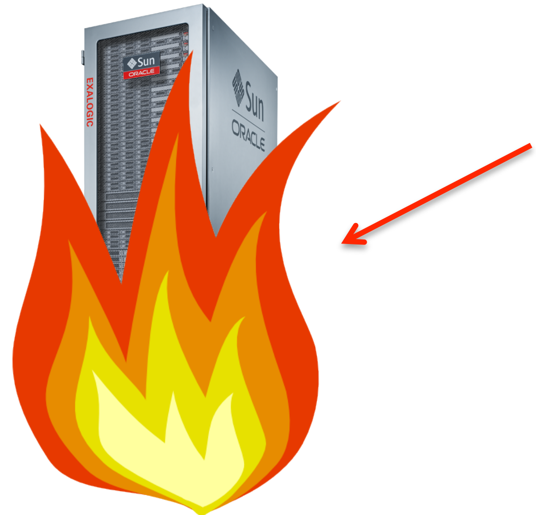
replicating data



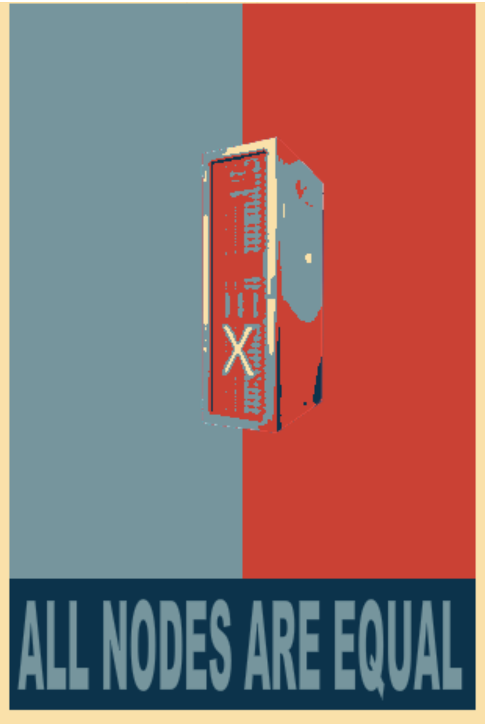
Availability Clock

Consistency > Availability

Unavailable to writes until data can be confirmed correct



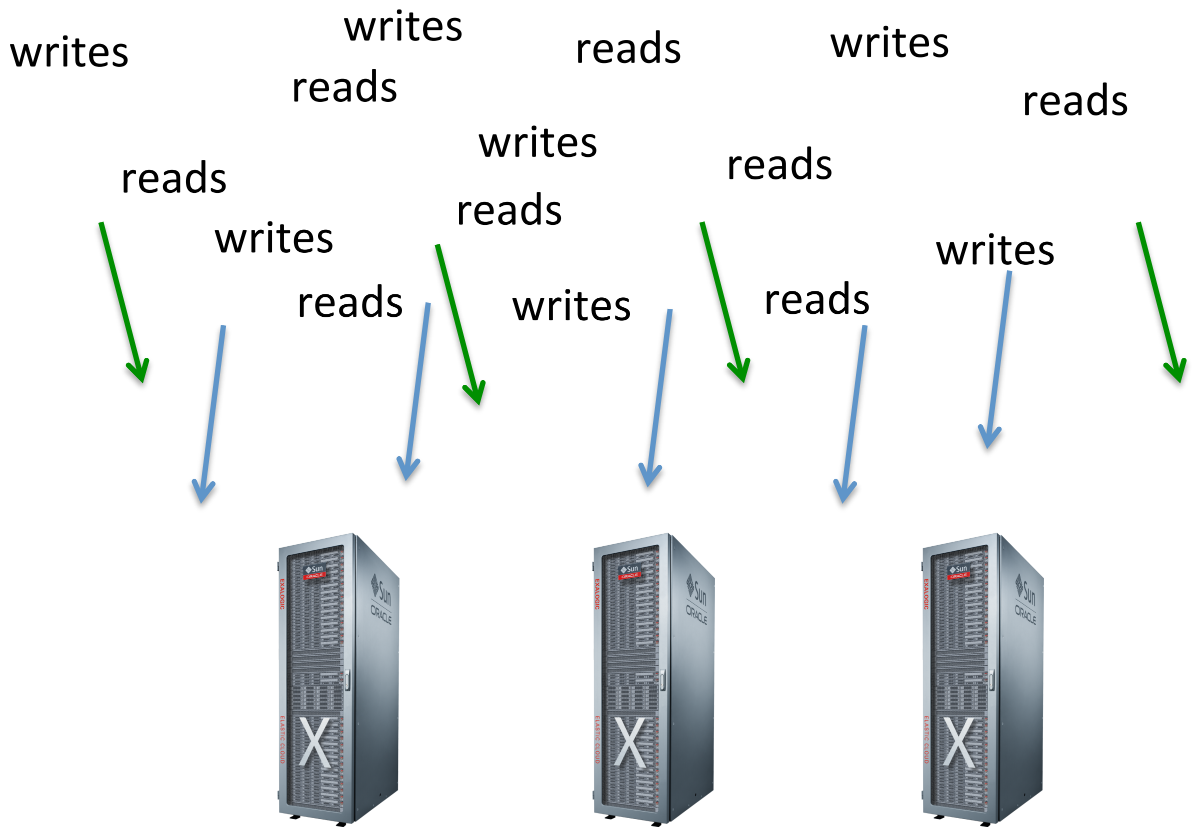
replicating data



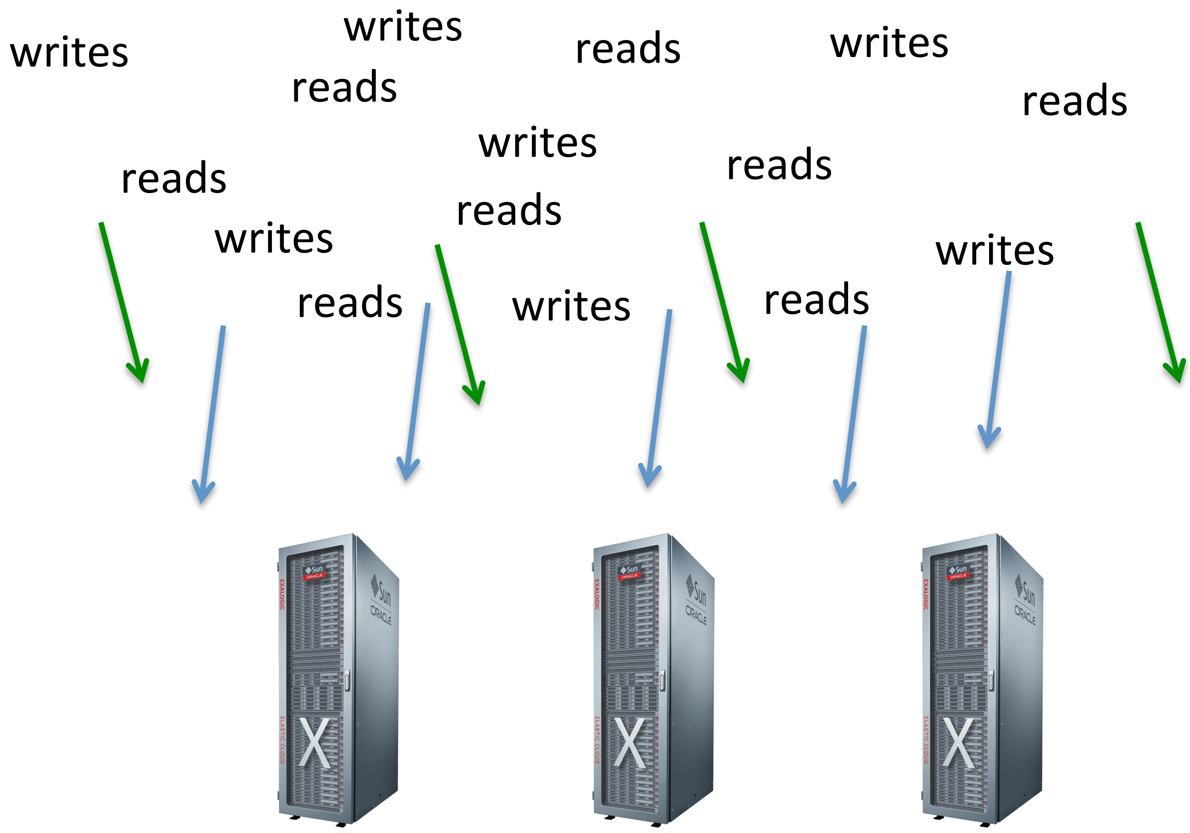
replicating data

“Every node in Dynamo should have the same set of responsibilities as its peers; there should be no distinguished node or nodes that take special roles or extra set of responsibilities.”

replicating data



replicating data



- Clients can read / write to any node
- All updates reach all replicas eventually

replicating data

- w and r values

replicating data

- number of replicas that need to participate in a read/write for a success response

replicating data

put()

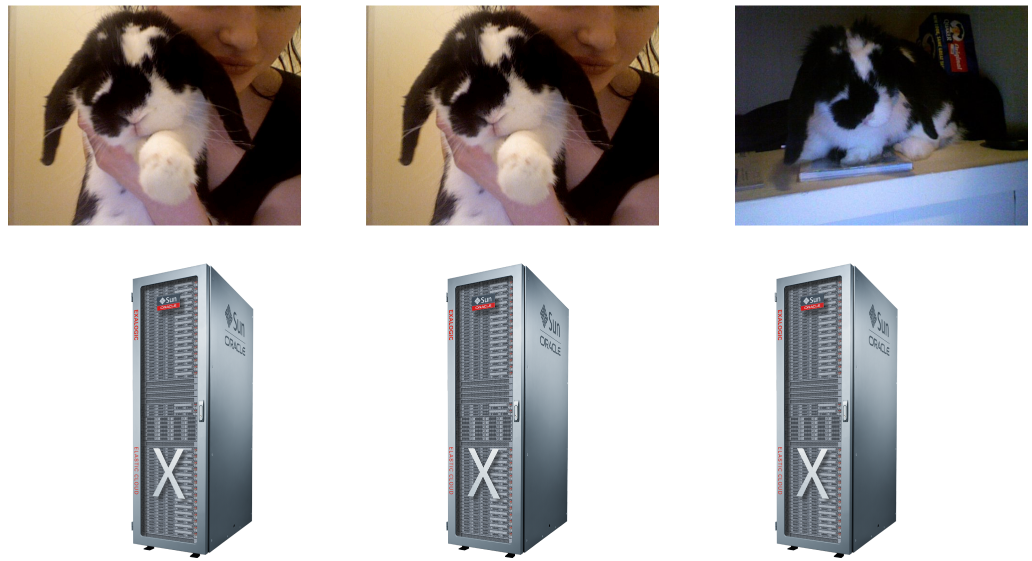
- w = 1
- only one node needs to be available to complete write request

replicating data

put( **)**

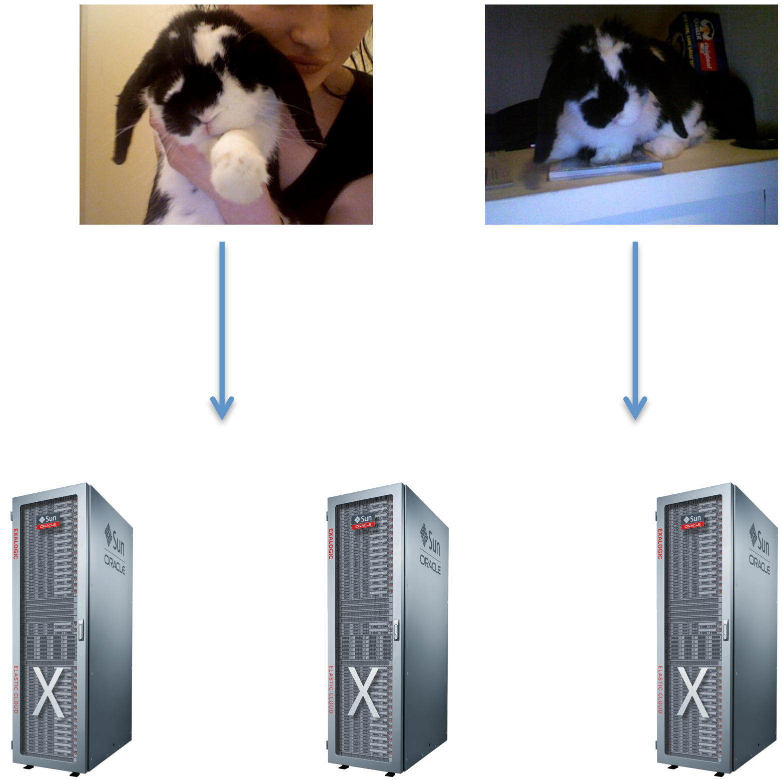
- w = 3

replicating data



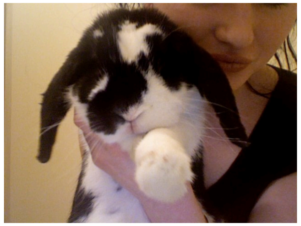
- reads when not all writes have propagated (laggy or down node)

resolving conflicts



- different clients update at the exact same time

resolving conflicts

 a85hYGBgzm

DKBVIsTFUPPmcwJ

TLmsTIcmsJ1nA8qz

K7HcQwqfB0hzNac

xCYWcA1ZIgsA

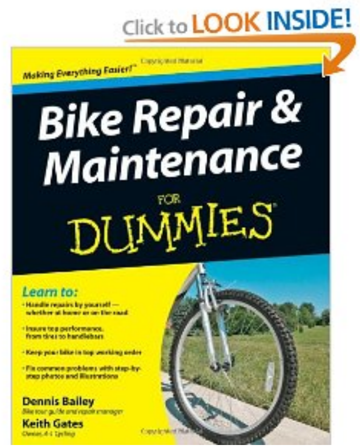
Whether one object is a direct descendant of the other
Whether the objects are direct descendants of a common parent
Whether the objects are unrelated in recent heritage

- vector clocks that show relationships between objects

resolving conflicts

- vector clock is updated when objects are updated
- last-write wins or conflicts can be resolved on client side

resolving conflicts



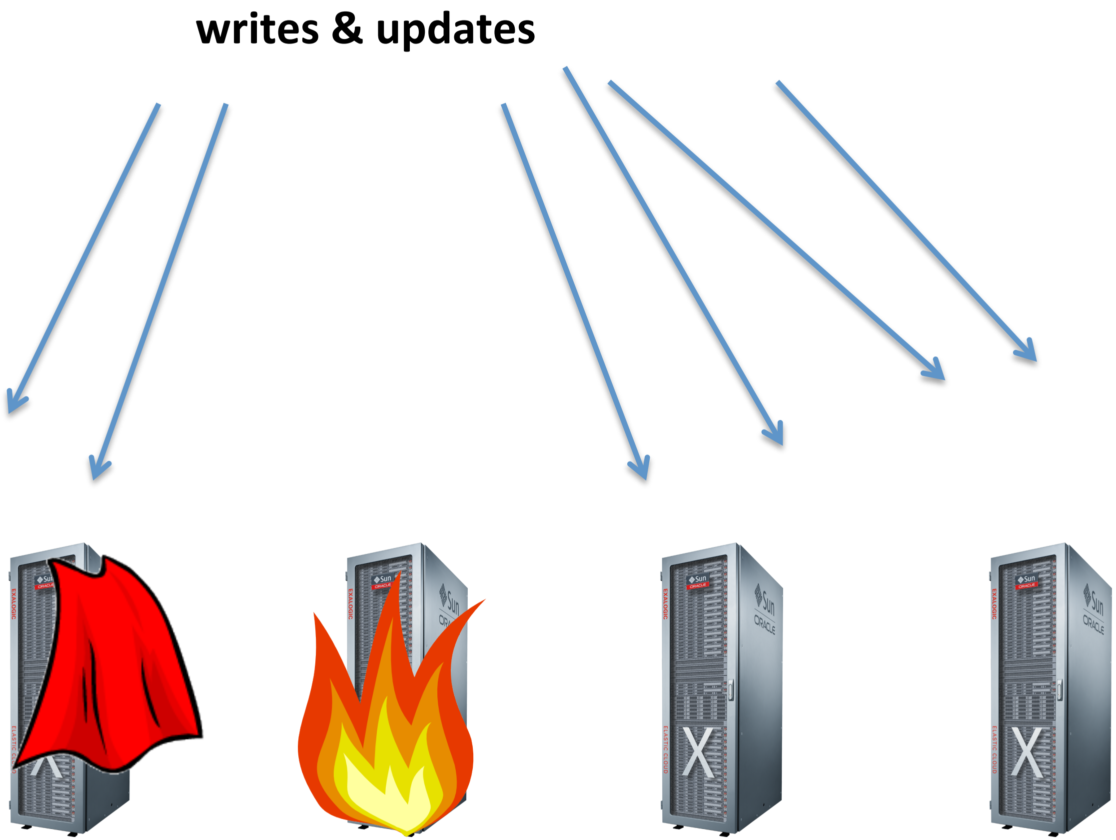
- if stale responses are returned as part of the read, those replicas are updated

resolving conflicts

failure conditions

- n = 3

failure conditions



- n = 3

failure conditions

- hinted handoff

failure conditions

developing apps

"Most of these services only store and retrieve data by primary key and do not require the complex querying and management functionality offered by an RDBMS."

developing apps

- “schema-less”
- more flexibility, agility

developing apps

app type	key	value
Session	User/Session ID	Session Data

developing apps

app type	key	value
Session	User/Session ID	Session Data
Advertising	Campaign ID	Ad Content

developing apps

app type	key	value
Session	User/Session ID	Session Data
Advertising	Campaign ID	Ad Content
Logs	Date	Log File

developing apps

app type	key	value
Session	User/Session ID	Session Data
Advertising	Campaign ID	Ad Content
Logs	Date	Log File
Sensor	Date, Date/Time	Updates

developing apps

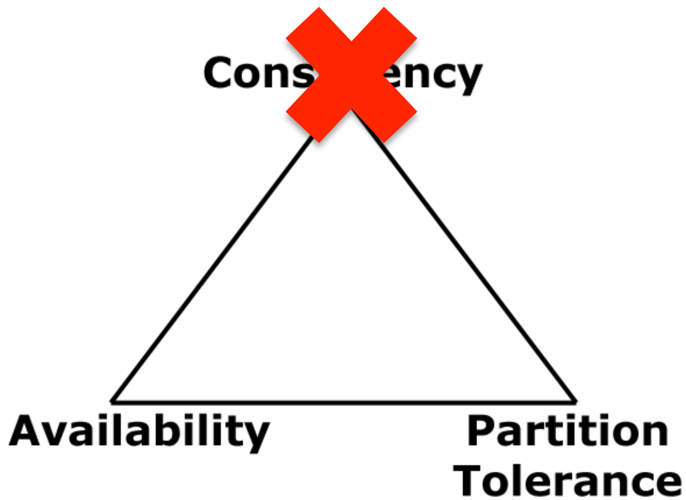
app type	key	value
Session	User/Session ID	Session Data
Advertising	Campaign ID	Ad Content
Logs	Date	Log File
Sensor	Date, Date/Time	Updates
User Data	Login, Email, UUID	User Attributes

developing apps

app type	key	value
Session	User/Session ID	Session Data
Advertising	Campaign ID	Ad Content
Logs	Date	Log File
Sensor	Date, Date/Time	Updates
User Data	Login, Email, UUID	User Attributes
Content	Title, Integer, Etc.	Text, JSON, XML

developing apps

future



future

future

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A comprehensive study of
Convergent and Commutative Replicated Data Types

Marc Shapiro, INRIA & LIP6, Paris, France
Nuno Preguiça, CITI, Universidade Nova de Lisboa, Portugal
Carlos Baquero, Universidade do Minho, Portugal

Marek Zawirski, INRIA & UPMC, Paris, France

13 Jan 2011

more data types

- counters
- sets
- sever side structure and conflict resolution policy

more data types

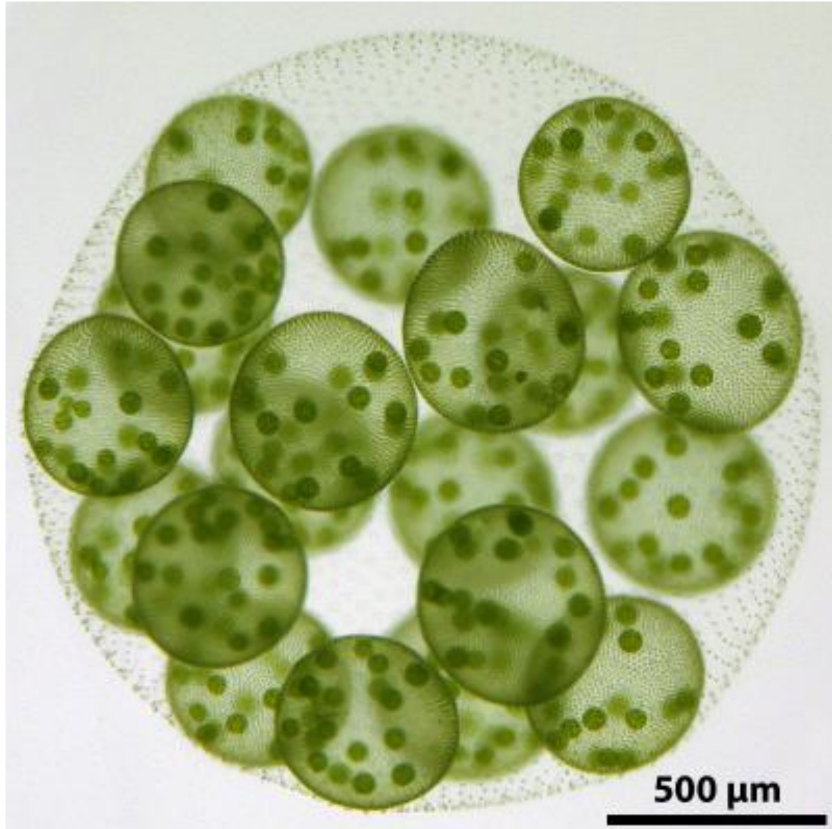
there is little reason to forfeit C or A when the system is not partitioned

strong consistency

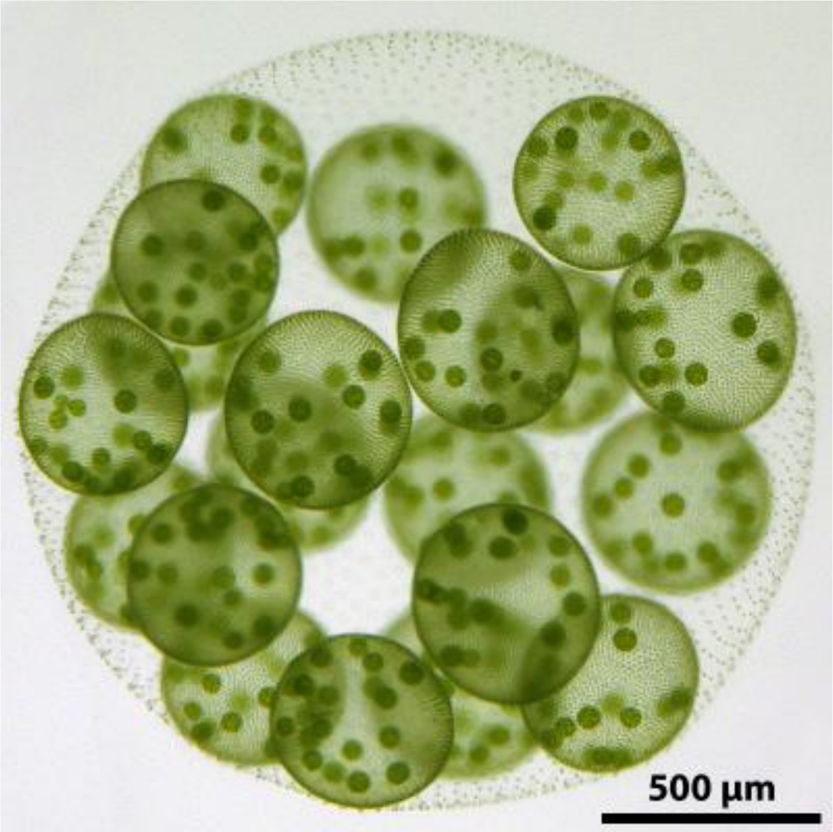
- conditional writes
- consistent reads

strong consistency

other advanced features

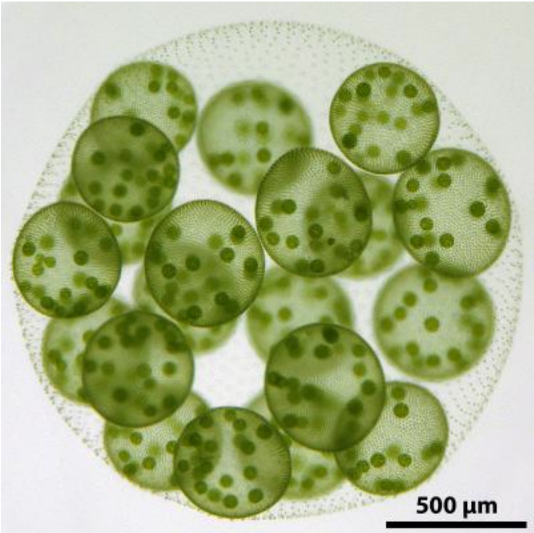


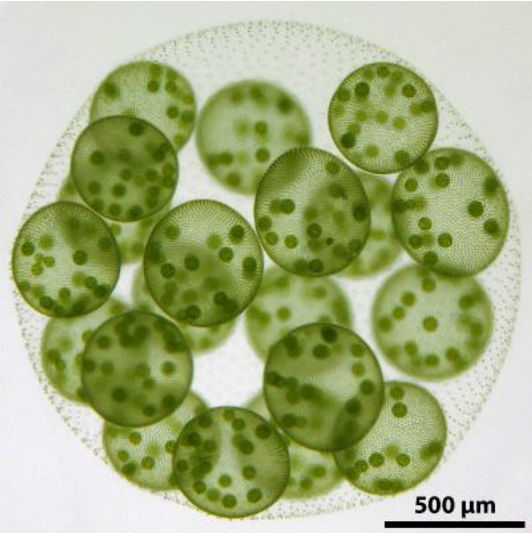
other advanced features



- metadata
- aggregation tasks
- search

other advanced features





- metadata
- aggregation tasks
- search

other advanced features

other advanced features

In summary…

rapid evolutionary change

significant events

explosion of new systems

evolving into higher-order systems

We’re hiring.

shanley@basho.com