简述 NoSQL

周振兴@2018

关于名字

来自Last.fm的Johan Oskarsson组织的一个meetup

关于：open source, distributed, non relational databases

目的： figuring out why these newfangled **Dynamo** clones and **BigTables** have become so popular lately

NOSQL meetup

Last.fm
Thursday, June 11, 2009 from 10:00 AM to 5:00 PM (PDT)
San Francisco, CA

Ticket Information

TYPE	REMAINING	END		QUANTITY
Free ticket	Sold Out	Ended	Free	**Sold Out**

Share NOSQL meetup
Share Tweet Like Sign Up to see what your friends like.

Event Details

Introduction
This meetup is about "open source, distributed, non relational databases".
Have you run into limitations with traditional relational databases? Don't mind trading a query language for scalability? Or perhaps you just like shiny new things to try out? Either way this meetup is for you.
Join us in figuring out why these newfangled Dynamo clones and BigTables have become so popular lately. We have gathered presenters from the most interesting projects around to give us all an introduction to the field.

Preliminary schedule
09.45: Doors open
10.00: **Intro session** (Todd Lipcon, Cloudera)
10.40: **Voldemort** (Jay Kreps, Linkedin)
11.20: Short break
11.30: **Cassandra** (Avinash Lakshman, Facebook)
12.10: Free lunch (sponsored by Last.fm)
13.10: **Dynomite** (Cliff Moon, Powerset)
13.50: **HBase** (Ryan Rawson, Stumbleupon)
14.30: Short break
14.40: **Hypertable** (Doug Judd, Zvents)
15.20: **CouchDB** (Chris Anderson, couch.io)
16.00: Short break
16.10: Lightning talks
16.40: Panel discussion
17.00: Relocate to Kate O'Brien's, 579 Howard St. @ 2nd. First round sponsored by Digg

Registration
The event is free but space is limited, please register if you wish to attend.

Location
Magma room, CBS interactive
235 Second Street
San Francisco, CA 94105

Sponsor
Thanks to Last.fm for providing lunch, CBS Interactive for the venue and to Digg for helping out with the beer tab!

Have questions about NOSQL meetup? Contact Last.fm

When & Where

CBS interactive, Magma room
235 Second Street
San Francisco, CA 94105

Thursday, June 11, 2009 from 10:00 AM to 5:00 PM (PDT)

Add to my calendar

Organizer

Last.fm
www.last.fm

Contact the Organizer
View organizer profile

1 past event on Eventbrite

NoSQL产生的背景

- 信息化、互联网化让数据指数增长
  - 超大数据量下的**扩展性**诉求
  - 很多大数据量的场景，**一致性可以妥协**
  - 大量**非/半结构化**的存储场景
- 传统的RDBMS
  - scale-up的成本很高
  - 不具备scale-out的能力
  - 为了一致性，单点失败总会带来不可用

对分布式诉求空前强烈

NoSQL的先行者

- Google BigTable
- Amazon Dynamo

Bigtable: A Distributed Storage System for Structured Data@2006
Dynamo: Amazon’ s Highly Available Key-value Store@2007

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber
{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com
Google, Inc.

Abstract

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many projects at Google store data in Bigtable, including web indexing, Google Earth, and Google Finance. These applications place very different demands on Bigtable, both in terms of data size (from URLs to web pages to satellite imagery) and latency requirements (from backend bulk processing to real-time data serving). Despite these varied demands, Bigtable has successfully achieved scalability and high performance, but Bigtable provides a different interface than such systems. Bigtable does not support a full relational data model; instead, it provides clients with a simple data model that supports dynamic control over data layout and format, and allows clients to reason about the locality properties of the data represented in the underlying storage. Data is indexed using row and column names that can be arbitrary strings. Bigtable also treats data as uninterpreted strings, although clients often serialize various forms of structured and semi-structured data into these strings. Clients

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

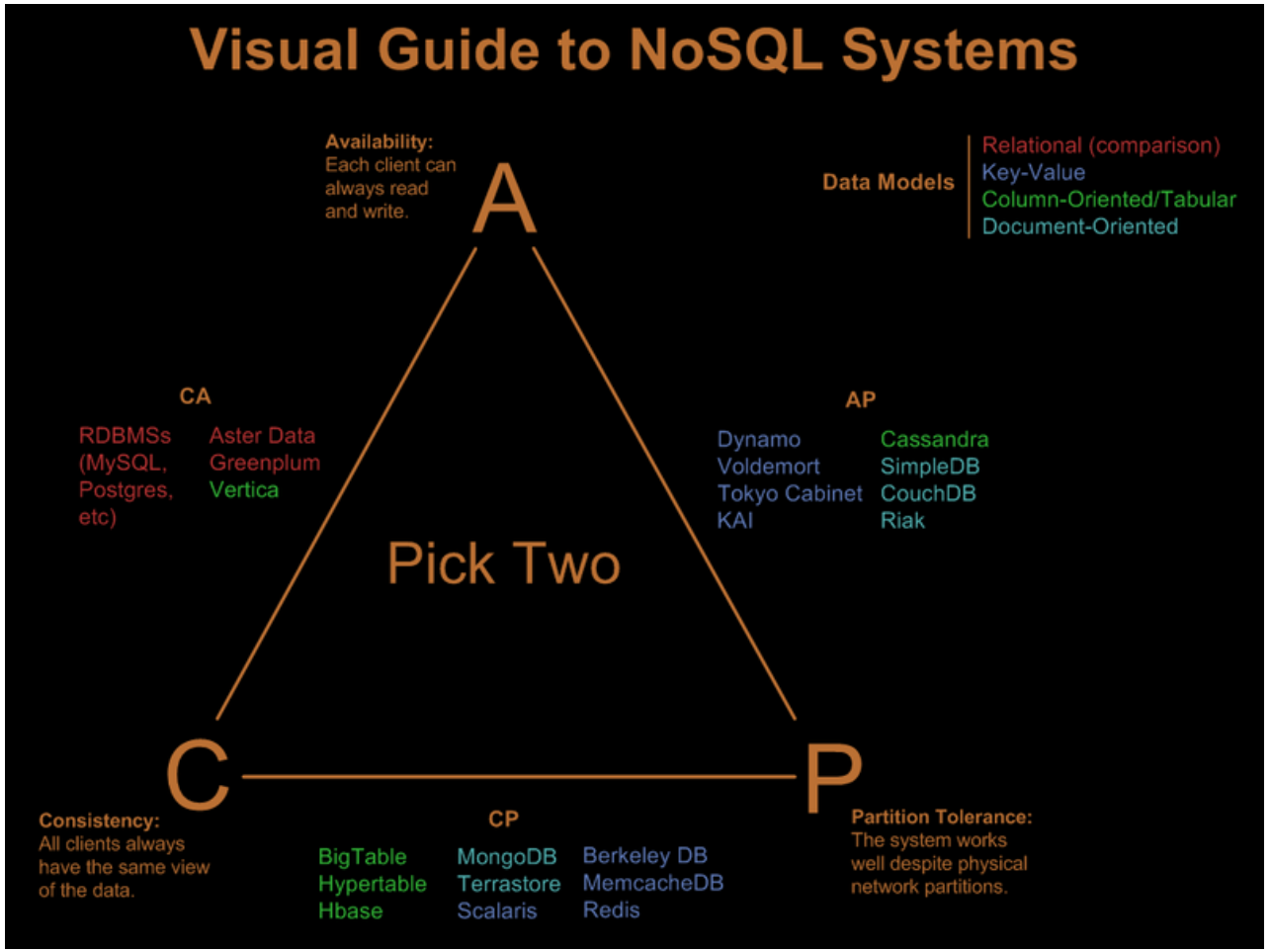
Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites 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.

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.

NoSQL的trade-off



Visual Guide to NoSQL Systems@March 15, 2010

"In some sense, the NoSQL movement is about creating choices that focus on availability first and consistency second; …"

CAP Twelve Years Later: How the "Rules" Have Changed

NoSQL的分类

- Key-Value Store
- Column Family (BigTable/HBase)
- Document Store(MongoDB)
- Graph Database

NoSQL Store

- Key-Value Stores
  - Dynamo Clones
    - Redis
    - Membase
    - Riak
    - Tokyo Cabinet
    - Voldemort
- Document Stores
  - MongoDB
  - CouchDB
  - SimpleDB
- Column Family
  - BigTable Clones
    - Cassandra
    - Hbase
    - HyperTable
- Graph Databases
  - Neo4J
  - Titan
  - InfoGrid
  - AllegroGraph

相关的技术

- Consistent hashing
- Bloom Filter
- Quorum
- Vector Clock
- MerkleTree
- Gossip protocol

一些参考

- http://nosql-database.org/ Your Ultimate Guide to the Non-Relational Universe!
- NoSQL@wikipedia
- NoSQL overview implementation free@slideshare 2011
- HBase 深入浅出@DeveloperWorks 写得非常 “深入浅出”
- CAP Twelve Years Later: How the "Rules" Have Changed
- Visual Guide to NoSQL Systems@2010