Wisdom from Crowds of Machines
Analytics and Big Data Summit
September 19, 2013

Chetan Conikee               Irfan Ahmad
About Us

CloudPhysics' mission is to discover the underlying principles that govern systems behavior and use this knowledge to transform computing.

Chetan Conikee
Chief Data Officer

• Founder, SeekVence
• Data Architect - Intuit
• Founding Engineer - Business Signatures (Entrust), CashEdge (Fiserv), Smartpipes (Sophos)

Irfan Ahmad
Chief Technology Officer

• VMware tech lead - DRS (Distributed Resource Scheduling) Team
• Transmeta (microprocessors)
Collective Intelligence

Secure multi-tenant architecture

- Perf Stat Dataset
- Inventory Dataset
- Event, Task Dataset

Parsers | Analyzers | Parsers
---|---|---
Schema Normalization
Ingest

Scale Out Query Engine
- Predictive Analytics
- Benchmarks
- Modeling
- Rules Engines

Scalable Cache

Apps Card Card Card Card Card

Web/App Servers

HTTPS
High-resolution performance, config, fault, change, event, task data

Collector vApp

HTTPS

CloudPhysics, Inc. Copyright 2013. All rights reserved.
We collect

• 100 billion+ data points /day
• Avg 1.3 million properties/datacenter
• Continuous time-series data stream
• VMs, servers, networks, storage
• Configuration, state, tasks, events
Our dialect is Scala

Productive
Fast
Functional
Statically typed

Actors - an elegant way to handle concurrency

... and
inter-op Java with Scala
Our Software ToolSet

Apache Kafka

mongoDB

Spark

redis

Cassandra

AngularJS

AnsibleWorks
Data Pipeline Criteria

- Secure Data Retention
- **NO** data-loss guarantee
- Scale out consumption process on demand
- Query near-to-real-time
- Query historical data on demand (replay)
Secure Data Retention

- Data organized and partitioned by time
- Stored directly in highly scalable, distributed object storage
- Encryption for data at rest - AES-256
- Processing pipeline starts from object storage
How do we scale?

courtesy: http://www.capricemotorinn.com/Dam1__html
Circa 2012

- A cluster of Play! + Akka app servers
- TechCrunch moment at VMworld 2012
- Exponential customer growth
HTTP Endpoints - Play!, Akka and Scala

- Lightweight Objects with Universal Concurrency primitives
- Message Passing Concurrency & no shared state
- Location Transparent & distributable by design
- Scale UP and Scale OUT on demand
- You get a PERFECT FABRIC for the CLOUD
Actor Model Design Issues

• **MessageBox** and **MessageProcessing** units are tightly coupled

• Increased backlog of streamed messages into the **MessageBox** can overload an Actor’s memory footprint

• Negatively Impact responsiveness leading to application crashes
courtesy: http://www.earthyreport.com
Let’s discuss design patterns

Effective Akka
Jamie Allen

Akka Concurrency
Derek Wyatt

A must read for serious Akka hAkkers

Work-Stealing or Work-Pull Pattern
Re-Design

• Use Akka durable unbounded mailbox (filesystem or Redis based)

• Use Akka work-stealing pattern and distribute load based on worker’s processing queue depth (deployed zookeeper for cluster co-ordination and service discovery)

• Akka Cluster was early in development (http://doc.akka.io/docs/akka/snapshot/common/cluster.html#cluster)
Decoupling Messaging from Processing - Why?

- Redundancy
- Scalability
- Resiliency
- Ordering Guarantee
- Asynchronous Communication
- Replay

Choices
- JMS
- AWS SQS
- ZeroMQ
- RabbitMQ
- Kafka
We chose **Apache Kafka**

Open Sourced by LinkedIn SNA 2011 and written in Scala

Kafka has:

- Broker(s)
- Topic(s)
  - Partition(s)
  - Replication
- Zookeeper Ensemble
- Consumer Grouping
- Buffering of Messages in
  - Producer & Consumer

```
Broker-1
  Topic-1/Part-1
  /Part-2
  Topic-2/Part-1

Broker-2
  Topic-1/Part-1
  /Part-2
  Topic-2/Part-1

Broker-3
  Topic-1/Part-1
  /Part-2
  Topic-2/Part-1

Producer

Consumer-Group
Consumer 1 ... n

Producer

Consumer-Group
Consumer 1

Zookeeper
```

PUSH
PULL

CloudPhysics Copyright 2013. All Rights Reserved.
Why Kafka? Disk Layout

Sequential file I/O on commit log

Simple storage

Batch writes and reads

Zero-copy transfer from files to socket

Compression(Batched)- GZIP and Snappy

Message caching in file system page cache

- Avoid double buffering and GC overhead
‘DOJO’
Data Collection/Processing Pipeline

Generation-2

CloudPhysics Observer(s) -> HTTP Cluster (Producers)

HTTP Play! Akka Cluster (Producers) -> Kafka - Broker

Kafka - Broker -> T1, T2, T3

T1, T2, T3 -> HTTP Play! Akka Cluster (Consumers)

HTTP Play! Akka Cluster (Consumers) -> HTTP Play! Akka Cluster Auto Scaling Group

HTTP Play! Akka Cluster Auto Scaling Group -> AWS S3-Object Store

AWS S3-Object Store -> Object Storage

Object Storage -> Kafka Consumer

Kafka Consumer -> HDFS Consumer

HDFS Consumer -> HDFS

HDFS -> Collective Intelligence

Collective Intelligence -> Spark Cluster Compute

Spark Cluster Compute -> HDFS

HDFS -> Storage

Storage -> KairosDB Cassandra Cluster

KairosDB Cassandra Cluster -> Event Metrics Store

Event Metrics Store -> Configuration Store

Configuration Store -> MongoDB Sharded Cluster

MongoDB Sharded Cluster -> Performance Metrics Store

Performance Metrics Store -> KairosDB Cassandra Cluster

KairosDB Cassandra Cluster

CloudPhysics Observer(s)
Kafka Tuning Considerations
Increased Partition Configuration (for high volume topics)

• Advantages
  • Increases I/O parallelism for writes
  • Increases degree of parallelism for consumers
  • Tune memory of consumer based on
    \[(\text{number of threads in consumer group}) \times (\text{queuedchunks.max}) \times (\text{fetch.size})\]
    • Increase Heap size
    • Reduce consumer thread count
    • Lower fetch size or max queue size

• Overheads
  • More open file handles
  • More offsets to be check-pointed increasing load on
    zookeeper ensemble
Kafka Performance Benchmarks

Consumer Throughput

Message size : 200 bytes
Batch size : 200 messages
Fetch size : 1 MB
Flush interval : 600 messages

No. of Consumers : 50
Data Consumed : 100 MB
No. of messages consumed : 6000000

courtesy:http://kafka.apache.org
Query Engine (Enso)

- An type-safe Scala API/DSL library (domain specific language) that auto-generates MongoDB and Cassandra query operators
- Allows querying complex deeply nested JSON structures
- Supports grouping, aggregation and filtering
- Supports on-demand querying of historical data
- Supports resource-management query priorities
IT/DevOps Orchestration

• The Dojo Cluster is a part of an auto scaling group
  • Compute nodes are added/deleted based on throughput characteristics - e.g. lag in consumption from Kafka brokers.

• We are currently leveraging AnsibleWorks IT Orchestration Engine
Questions?