O'REILLY®

Software Architecture CONFERENCE

MARCH 16–17, 2015: TRAINING
MARCH 17–19, 2015: CONFERENCE
BOSTON, MA

Can You Process 10 Trillion Logs Per Day?

sumologic

Christian Beedgen, CTO & Co-Founder, @raychaser

March 19th, 2015

Who Is Christian?

- Co-Founder & CTO, Sumo Logic since 2010
- Server guy, Chief Architect at ArcSight, 2001 - 2009

Agenda

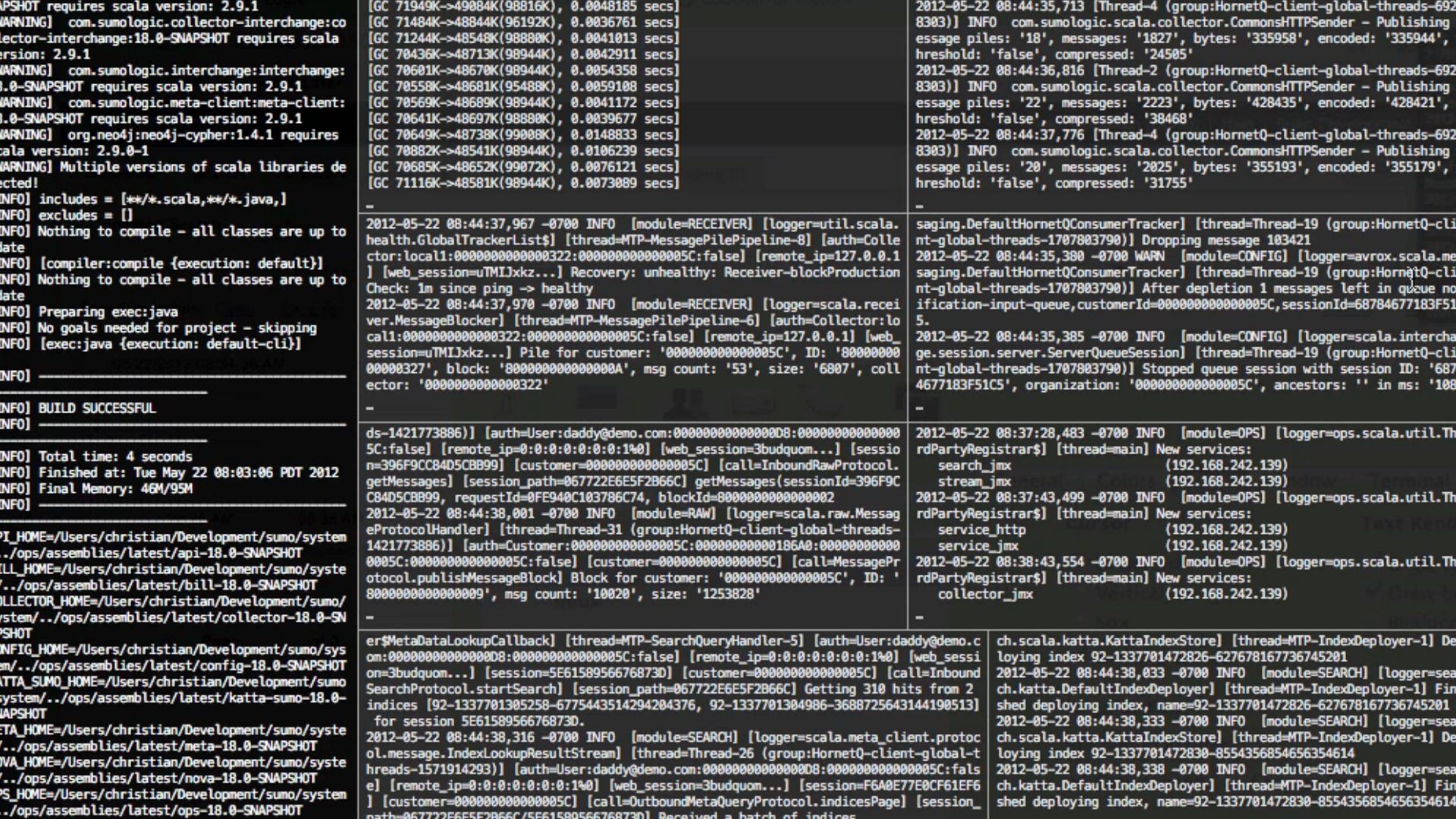
Purpose, Practice, Philosophy

- Why We Are What We Are
- How We Set Things Up
- What We Have Learned

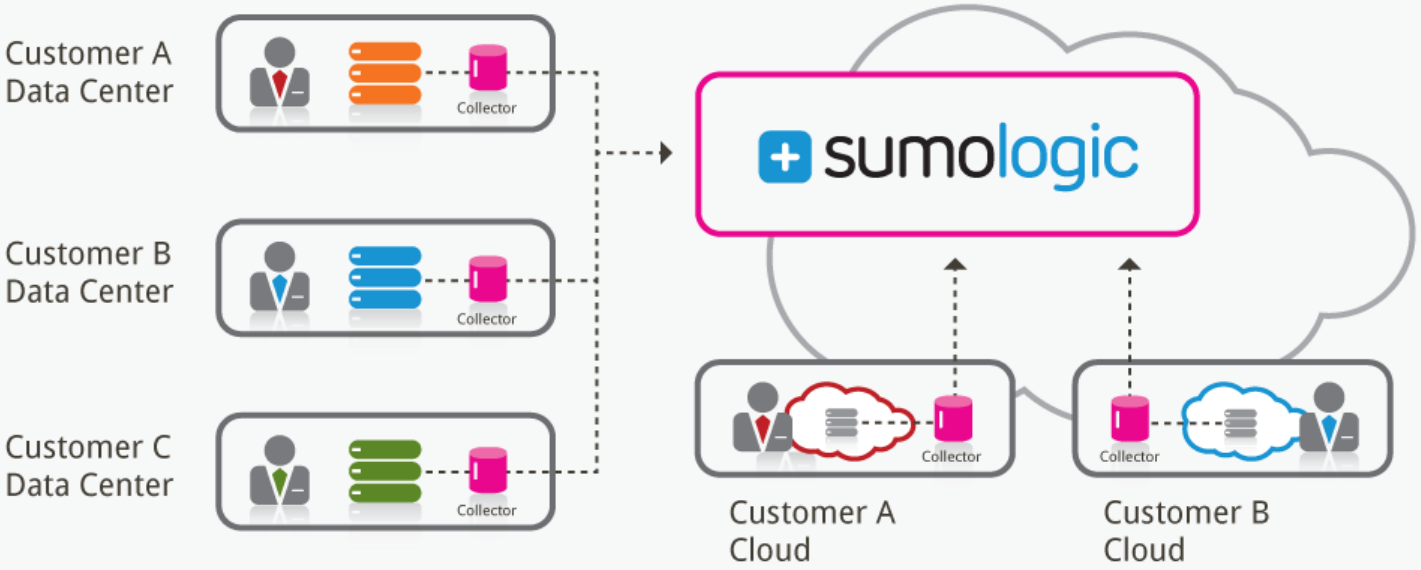

INVISIBLE LOLCAT

Why We Are What We Are

Machine Data Analytics


[GC 71949K->49084K(98816K), 0.0048185 secs]
[GC 71484K->48844K(96192K), 0.0036761 secs]
[GC 71244K->48548K(98880K), 0.0041013 secs]
[GC 70436K->48713K(98944K), 0.0042911 secs]
[GC 70601K->48670K(98944K), 0.0054358 secs]
[GC 70558K->48681K(95488K), 0.0059108 secs]
[GC 70569K->48689K(98944K), 0.0041172 secs]
[GC 70641K->48697K(98880K), 0.0039677 secs]
[GC 70649K->48738K(99008K), 0.0148833 secs]
[GC 70882K->48541K(98944K), 0.0106239 secs]
[GC 70685K->48652K(99072K), 0.0076121 secs]
[GC 71116K->48581K(98944K), 0.0073089 secs]
2012-05-22 08:44:35,713 [Thread-4 (group:HornetQ-client-global-threads-692
8303)] INFO com.sumologic.scala.collector.CommonsHTTPSender - Publishing
essage piles: '18', messages: '1827', bytes: '335958', encoded: '335944',
hreshold: 'false', compressed: '24505'
2012-05-22 08:44:36,816 [Thread-2 (group:HornetQ-client-global-threads-692
8303)] INFO com.sumologic.scala.collector.CommonsHTTPSender - Publishing
essage piles: '22', messages: '2223', bytes: '428435', encoded: '428421',
hreshold: 'false', compressed: '38468'
2012-05-22 08:44:37,776 [Thread-4 (group:HornetQ-client-global-threads-692
8303)] INFO com.sumologic.scala.collector.CommonsHTTPSender - Publishing
essage piles: '20', messages: '2025', bytes: '355193', encoded: '355179',
hreshold: 'false', compressed: '31755'
2012-05-22 08:44:37,967 -0700 INFO [module=RECEIVER] [logger=util.scala.
health.GlobalTrackerList$] [thread=MTP-MessagePilePipeline-8] [auth=Colle
ctor:local1:0000000000000322:000000000000005C:false] [remote_ip=127.0.0.1
] [web_session=uTMIJxkz...] Recovery: unhealthy: Receiver-blockProduction
Check: 1m since ping -> healthy
2012-05-22 08:44:37,970 -0700 INFO [module=RECEIVER] [logger=scala.recei
ver.MessageBlocker] [thread=MTP-MessagePilePipeline-6] [auth=Collector:lo
cal1:0000000000000322:000000000000005C:false] [remote_ip=127.0.0.1] [web_
session=uTMIJxkz...] Pile for customer: '000000000000005C', ID: '80000000
00000327', block: '800000000000000A', msg count: '53', size: '6807', coll
ector: '0000000000000322'
saging.DefaultHornetQConsumerTracker] [thread=Thread-19 (group:HornetQ-cli
nt-global-threads-1707803790)] Dropping message 103421
2012-05-22 08:44:35,380 -0700 WARN [module=CONFIG] [logger=avrox.scala.me
saging.DefaultHornetQConsumerTracker] [thread=Thread-19 (group:HornetQ-cli
nt-global-threads-1707803790)] After depletion 1 messages left in queue no
ification-input-queue,customerId=000000000000005C,sessionId=68784677183F51
5.
2012-05-22 08:44:35,385 -0700 INFO [module=CONFIG] [logger=scala.intercha
ge.session.server.ServerQueueSession] [thread=Thread-19 (group:HornetQ-cli
nt-global-threads-1707803790)] Stopped queue session with session ID: '687
4677183F51C5', organization: '000000000000005C', ancestors: '' in ms: '108
ds-1421773886)] [auth=User:daddy@demo.com:000000000000000D8:00000000000000
5C:false] [remote_ip=0:0:0:0:0:0:0:1%0] [web_session=3budquom...] [sessio
n=396F9CC84D5CBB99] [customer=000000000000005C] [call=InboundRawProtocol.
getMessages] [session_path=067722E6E5F2B66C] getMessages(sessionId=396F9C
C84D5CBB99, requestId=0FE940C103786C74, blockId=8000000000000002
2012-05-22 08:44:38,001 -0700 INFO [module=RAW] [logger=scala.raw.Messag
eProtocolHandler] [thread=Thread-31 (group:HornetQ-client-global-threads-
1421773886)] [auth=Customer:000000000000005C:0000000000186A0:00000000000
0005C:000000000000005C:false] [customer=000000000000005C] [call=MessagePr
otocol.publishMessageBlock] Block for customer: '000000000000005C', ID: '
8000000000000009', msg count: '10020', size: '1253828'
2012-05-22 08:37:28,483 -0700 INFO [module=OPS] [logger=ops.scala.util.Th
rdPartyRegistrar$] [thread=main] New services:
search_jmx (192.168.242.139)
stream_jmx (192.168.242.139)
2012-05-22 08:37:43,499 -0700 INFO [module=OPS] [logger=ops.scala.util.Th
rdPartyRegistrar$] [thread=main] New services:
service_http (192.168.242.139)
service_jmx (192.168.242.139)
2012-05-22 08:38:43,554 -0700 INFO [module=OPS] [logger=ops.scala.util.Th
rdPartyRegistrar$] [thread=main] New services:
collector_jmx (192.168.242.139)
er$MetaDataLookupCallback] [thread=MTP-SearchQueryHandler-5] [auth=User:daddy@demo.c
om:000000000000000D8:000000000000005C:false] [remote_ip=0:0:0:0:0:0:0:1%0] [web_sessi
on=3budquom...] [session=5E61589566768730] [customer=000000000000005C] [call=Inbound
SearchProtocol.startSearch] [session_path=067722E6E5F2B66C] Getting 310 hits from 2
indices [92-1337701305258-6775443514294204376, 92-1337701304986-3688725643144190513]
for session 5E61589566768730.
2012-05-22 08:44:38,316 -0700 INFO [module=SEARCH] [logger=scala.meta_client.protoc
ol.message.IndexLookupResultStream] [thread=Thread-26 (group:HornetQ-client-global-t
hreads-1571914293)] [auth=User:daddy@demo.com:000000000000000D8:000000000000005C:fals
e] [remote_ip=0:0:0:0:0:0:0:1%0] [web_session=3budquom...] [session=F6A0E77E0CF61EF6
] [customer=000000000000005C] [call=OutboundMetaQueryProtocol.indicesPage] [session_
ch.scala.katta.KattaIndexStore] [thread=MTP-IndexDeployer-1] De
loying index 92-1337701472826-627678167736745201
2012-05-22 08:44:38,033 -0700 INFO [module=SEARCH] [logger=sea
ch.katta.DefaultIndexDeployer] [thread=MTP-IndexDeployer-1] Fin
shed deploying index, name=92-1337701472826-627678167736745201
2012-05-22 08:44:38,333 -0700 INFO [module=SEARCH] [logger=sea
ch.scala.katta.KattaIndexStore] [thread=MTP-IndexDeployer-1] De
loying index 92-1337701472830-8554356854656354614
2012-05-22 08:44:38,338 -0700 INFO [module=SEARCH] [logger=sea
ch.katta.DefaultIndexDeployer] [thread=MTP-IndexDeployer-1] Fin
shed deploying index, name=92-1337701472830-8554356854656354614
APSHOT requires scala version: 2.9.1
WARNING] com.sumologic.collector-interchange:co
llector-interchange:18.0-SNAPSHOT requires scala
ersion: 2.9.1
WARNING] com.sumologic.interchange:interchange:
8.0-SNAPSHOT requires scala version: 2.9.1
WARNING] com.sumologic.meta-client:meta-client:
8.0-SNAPSHOT requires scala version: 2.9.1
WARNING] org.neo4j:neo4j-cypher:1.4.1 requires
cala version: 2.9.0-1
WARNING] Multiple versions of scala libraries de
ected!
INFO] includes = [**/*.scala,**/*.java,]
INFO] excludes = []
INFO] Nothing to compile - all classes are up to
date
INFO] [compiler:compile {execution: default}]
INFO] Nothing to compile - all classes are up to
date
INFO] Preparing exec:java
INFO] No goals needed for project - skipping
INFO] [exec:java {execution: default-cli}]
INFO] --
INFO] BUILD SUCCESSFUL
INFO] --
INFO] Total time: 4 seconds
INFO] Finished at: Tue May 22 08:03:06 PDT 2012
INFO] Final Memory: 46M/95M
INFO] --
PI_HOME=/Users/christian/Development/sumo/system
../ops/assemblies/latest/api-18.0-SNAPSHOT
LL_HOME=/Users/christian/Development/sumo/syste
/../ops/assemblies/latest/bill-18.0-SNAPSHOT
OLLECTOR_HOME=/Users/christian/Development/sumo/
ystem/../ops/assemblies/latest/collector-18.0-SN
PSHOT
NFIG_HOME=/Users/christian/Development/sumo/sys
em/../ops/assemblies/latest/config-18.0-SNAPSHOT
ATTA_SUMO_HOME=/Users/christian/Development/sumo
ystem/../ops/assemblies/latest/katta-sumo-18.0-
APSHOT
ETA_HOME=/Users/christian/Development/sumo/syste
/../ops/assemblies/latest/meta-18.0-SNAPSHOT
OVA_HOME=/Users/christian/Development/sumo/syste
/../ops/assemblies/latest/nova-18.0-SNAPSHOT
PS_HOME=/Users/christian/Development/sumo/system
../ops/assemblies/latest/ops-18.0-SNAPSHOT

Sumo Logic Cloud-based Deployment



Why Is Sumo Logic A Service

A Conscious, Fundamental Decision

Simply the best way to deliver Machine Data Analytics

Full control → product development efficiency

Why Is Sumo Logic A Service

A Conscious, Fundamental Decision

Simply the best way to deliver Machine Data Analytics

Full control → product development efficiency

Why Machine Data Analytics As A Service

The Big Data Imperative

- Machine data is actually Big Data
- Big Data enterprise software is expensive and painful
- As a service, we deliver more value at a lower cost

What Is Machine Data

Actually, It's Machine ***Generated*** Data

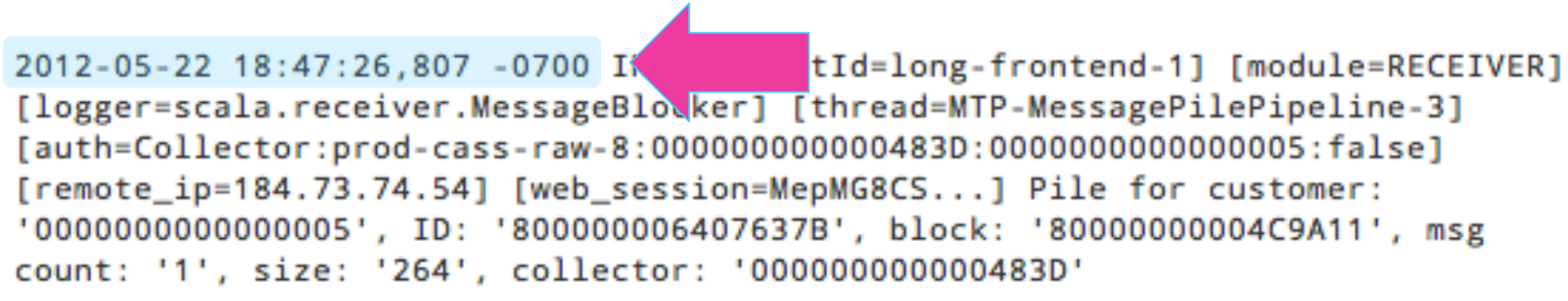
Curt Monash:

"Data that was produced entirely by machines OR data that is more about observing humans than recording their choices."

Daniel Abadi:

"Machine-generated data is data that is generated as a result of a decision of an independent computational agent or a measurement of an event that is not caused by a human action."

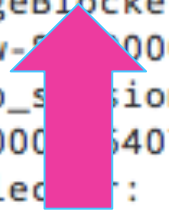
Example



- Timestamp with time zone!

Example

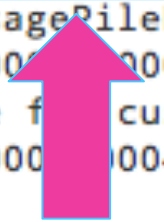
```
2012-05-22 18:47:26,807 -0700 INFO  [hostId=long-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker] [thread=MTP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-5:0000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '800000[illegible]5407637B', block: '80000000004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```



- Timestamp with time zone!
- Log level

Example

```
2012-05-22 18:47:26,807 -0700 INFO  [hostId=long-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker] [thread=MTP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:000000000000483D:0000000 00005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile f  customer:
'0000000000000005', ID: '8000000006407637B', block: '80000  004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```



- Timestamp with time zone!
- Log level
- Host ID & module name (process/service)

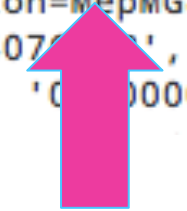
Example

```
2012-05-22 18:47:26,807 -0700 INFO  [host=long-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker] [                TP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:0000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '8000000006407637B', block: '80000000004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```

- Timestamp with time zone!
- Log level
- Host ID & module name (process/service)
- Code location or class

Example

```
2012-05-22 18:47:26,807 -0700 INFO  [hostId=long-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker] [thread=MTP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:000000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '800000006407[illegible]', block: '80000000004C9A11', msg
count: '1', size: '264', collector: '[illegible]000000000483D'
```



- Timestamp with time zone!
- Log level
- Host ID & module name (process/service)
- Code location or class
- Authentication context

Example

```
2012-05-22 18:47:26,807 -0700 INFO  [hostId=long-frontend-1] [module=RECEIVER]
[logger=scala.receiver.MessageBlocker] [thread=MTP-MessagePilePipeline-3]
[auth=Collector:prod-cass-raw-8:000000000000483D:0000000000000005:false]
[remote_ip=184.73.74.54] [web_session=MepMG8CS...] Pile for customer:
'0000000000000005', ID: '8000000064076378', block: '8000000000004C9A11', msg
count: '1', size: '264', collector: '000000000000483D'
```

- Timestamp with time zone!
- Log level
- Host ID & module name (process/service)
- Code location or class
- Authentication context
- Key-value pairs

Machine Data Is Big Data

V For Big Data

- Volume
  - Machine Data is voluminous and will continue to grow
  - Our own application creates 1TB/logs per day easily
- Velocity
  - Machine Data occurs in real-time, and it is time-stamped
  - Needs to be processed in real-time as well
- Variety
  - Machine Data is unstructured, or poly-structured at best
  - Some standard schemas, but sure enough not for the apps you built

Enterprise Software Is An Exercise In Undifferentiated Heavy-Lifting

(For Your Customer)

Big Data Makes Things Worse

A Eulogy For Enterprise Software

- Cluster-able, modern, N+1 scalable Enterprise Software?
  - Old architectures from back when the products were conceived
  - How many Enterprise Software applications deploy on Hadoop?
- It's not just the cost of the software license…
  - Lead time and cost of servers, storage, networking, plus the cost of HA, DR
  - Cost of the people that are maintaining the infrastructure
- Have you ever upgraded Enterprise Software?
  - Latest and greatest is always appealing but not always well tested
  - Upgrade the Oracle database, migrate the configuration, migrate the data, …

More Value At A Lower Cost

A Different Business Model

NO HEAVY-LIFTING REQUIRED

- Using vs. running the product
- Easy on, easy off

LOWER TCO

- No server, storage, admin cost
- Vendor economies of scale

Who Watches The Watchmen?

This Stuff Actually Matters In Real Life

- How much monitoring does your monitoring solution require?
- How much does your monitoring solution add to your monitored footprint?
- Do you know what infinite recursion is all about?

Why Is Sumo Logic A Service

A Conscious, Fundamental Decision

Simply the best way to deliver Machine Data Analytics

Full control → product development efficiency

Velocity Increases

Visibility Is Perfect

Cost Decreases

Velocity Increases

Visibility Is Perfect

Cost Decreases

Deployment Environment Is Controlled

Control Affords Predictability

- No more surprises due to out of control circumstances
  - Misread technical documentation, sheer ignorance, lack of time
  - Obscure runtime issues creating hard to track down issues


IBM
IBM
System p5

Deployment Environment Is Controlled

Control Affords Predictability

- No more surprises due to out of control circumstances
  - Misread technical documentation, sheer ignorance, lack of time
  - Obscure runtime issues creating hard to track down issues
- Code and test against the actual runtime environment
  - Control over the full stack removes a lot of variables
  - Actually testing in production is still hard, but at least there's progress

Only One Production Branch

Fear The Cartesian Product

- Less dimensions in the testing matrix
  - One and only one product and version to test
  - Bugs on arcane platform not even the developers know
- Every release of enterprise software decreases velocity
  - Customers try to avoid the upgrade hassle and cost
  - Laggard customer fall further behind, forcing support for older and older versions

Velocity Increases

Visibility Is Perfect

Cost Decreases

The Point Here Is That If You Don’t Exploit The Visibility You Now Have, You Really Should Whack Yourself In The Face Daily

Velocity Increases

Visibility Is Perfect

Cost Decreases

Cost Decreases Along Many Dimensions

Obvious & Hidden Dimensions Play Here

- Lower support costs
  - Much better visibility leads to faster MTTI of customer issues
  - Can also reinvest savings into even better support and have happier customers
- What is the true cost of pissed off customers?
  - Immediate financial impact due to churn
  - Mid-term brand damage because word travels quickly
- Test and maintenance spend reduction
  - Test matrix is greatly reduced
  - No maintenance developer spend

Why Is Sumo Logic A Service

A Conscious, Fundamental Decision

Simply the best way to deliver Machine Data Analytics

Full control → product development efficiency

What Decisions Did We Make?

Don’t Forget, A Bunch Of Software Developers Started This Company

We Need To

- Focus on what we know and what we have learned
- Do everything in code that can be done in code
- Evolve from software architects to system architects
- Acknowledge our responsibility for the runtime behavior

How We Set Things Up

Multitenancy

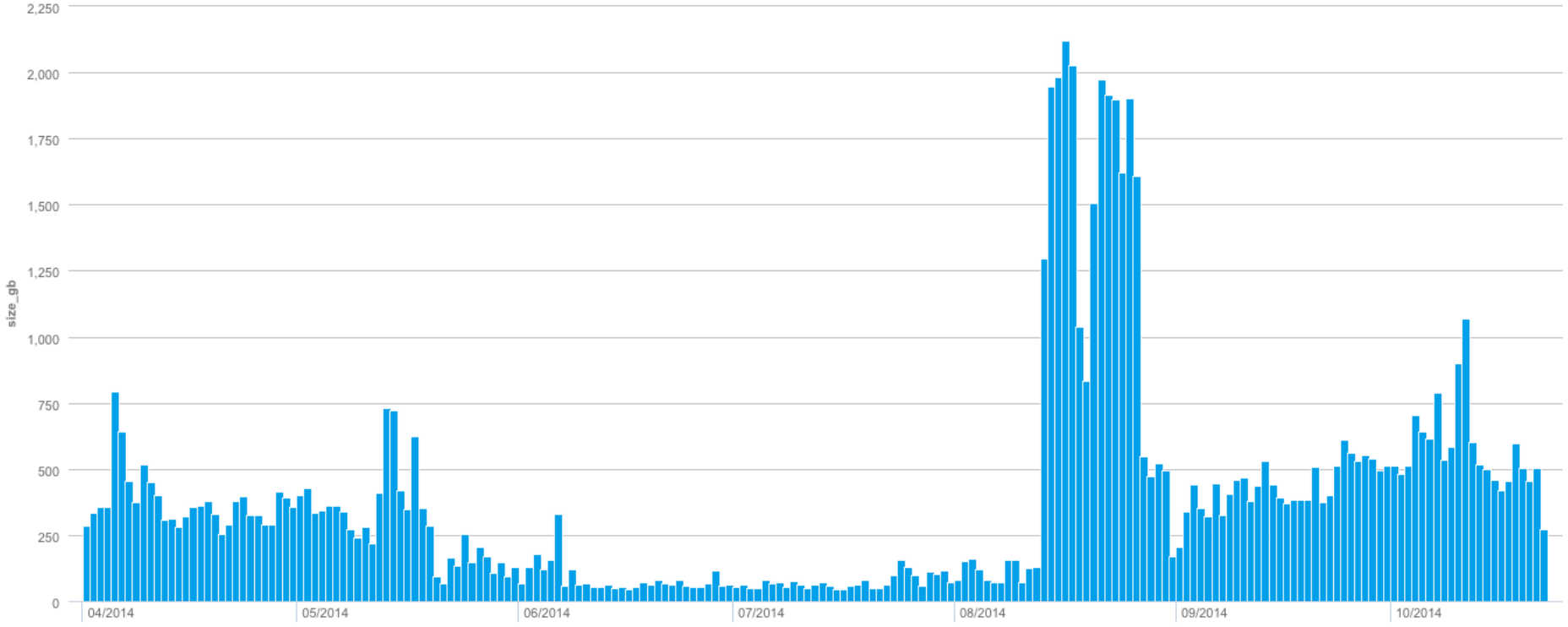
Adaptability


Multitenancy

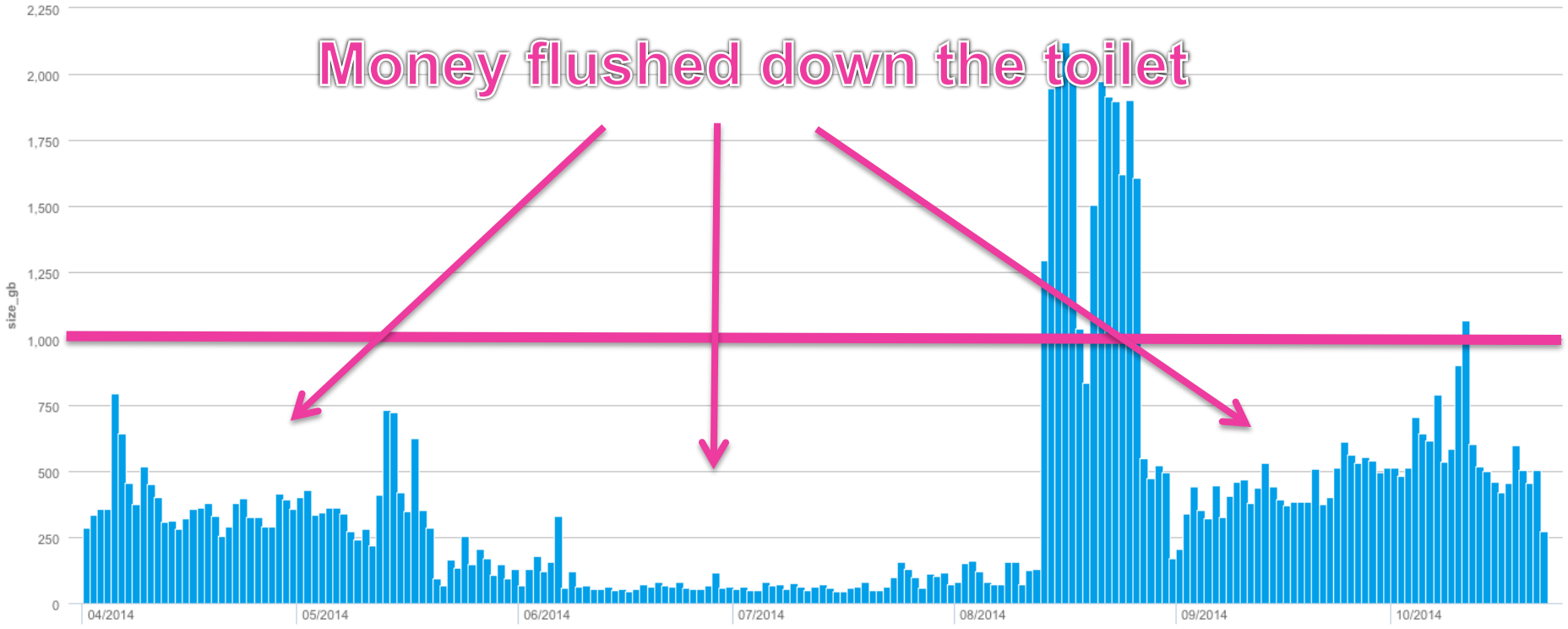
Better Economics

Differentiated Pricing

- No fixed, per-customer costs
  - No fixed provisioned infrastructure as compared to managed service offerings
  - Customers are cattle, not pets – no per-customer administration cost
- Provision for what is actually used
  - Customer usage is not uniform but varies by time of day/week/year
  - The good old “Sum of peaks vs. peak of sums” argument

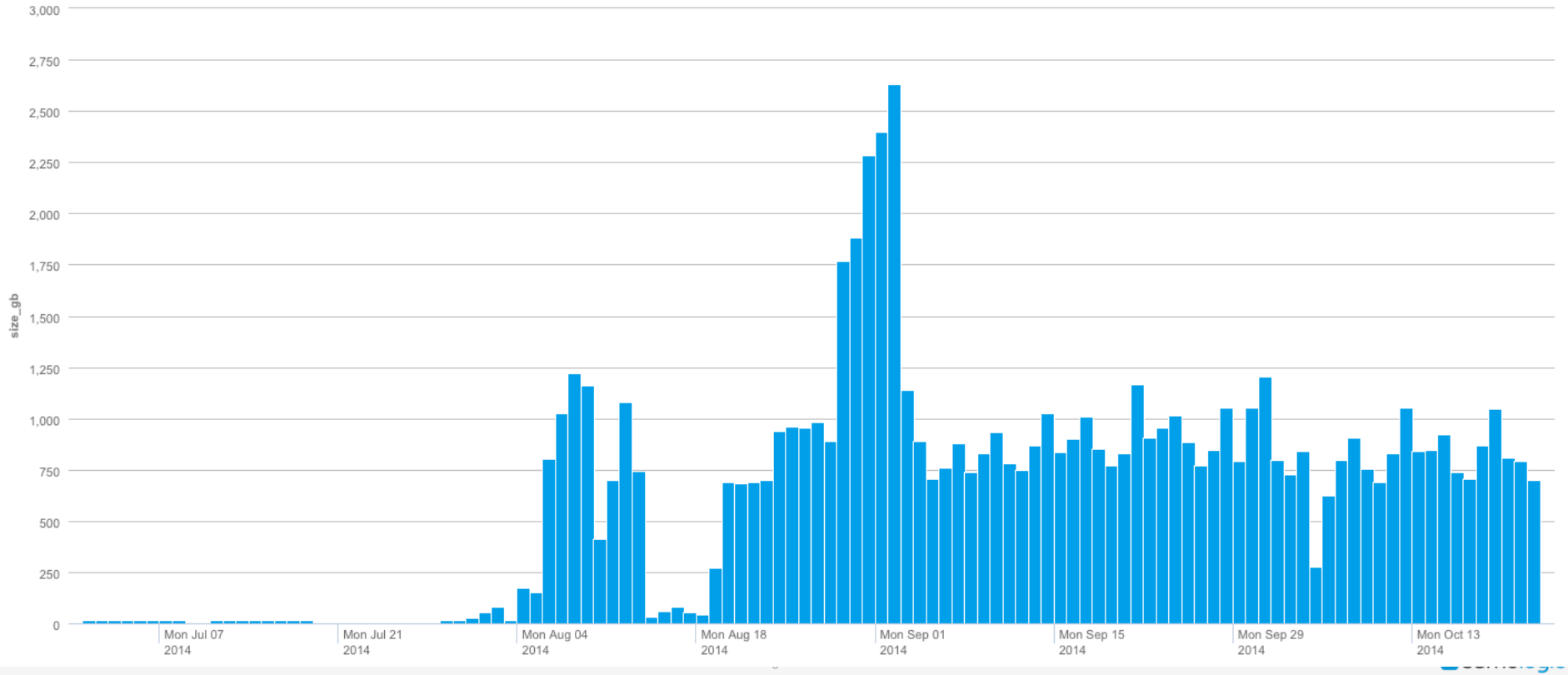


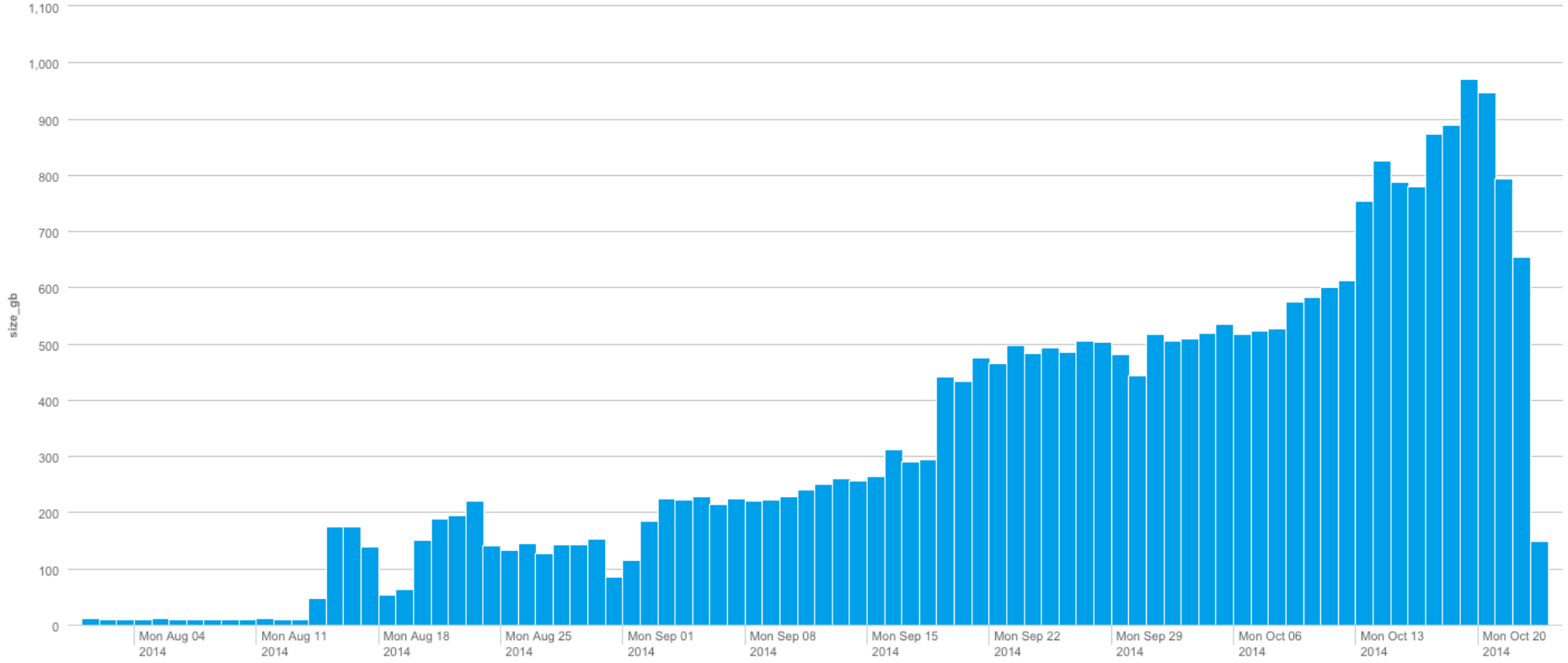
Just one typical Sumo Logic customer - 8x Variance!



Just one typical Sumo Logic customer - 8x Variance!

Here’s another one – spike at 2.5 of steady…





Or… Sweet, incremental, unfettered growth

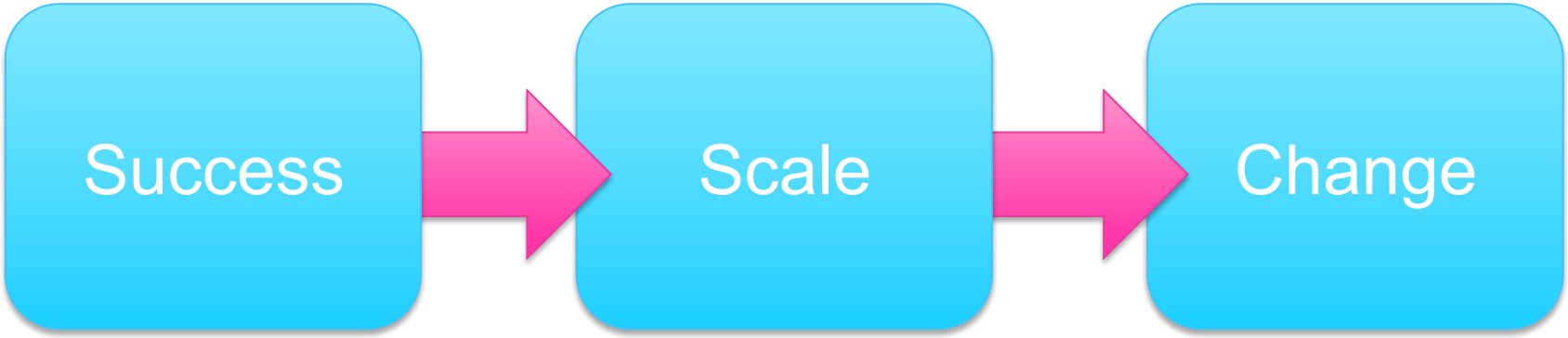
Even More Product Development Efficiency

There Simply Is Just One System

- No version drift
  - Only one version of the code, only one version of the configuration
  - No time wasted debugging custom, one-off stuff
- Much better update cycle times
  - No per-customer configuration stands in the way
  - No manual steps anywhere


Adaptability

Architect For An Extreme Rate Of Change



Change Is Our Success

Success Leads To Scale

Pretty Clever, Huh?

- Charging for ingested data is our business model
- We make more money if we sell more daily ingested data
- We need to be able to scale to ever more daily data

Scaling Implies Change

There's The Catch!

- Scaling challenges assumptions about system behavior
- To adopt to the new reality, changes are required
- So in order to scale we need to be able to make changes

We Don’t Know Nothing

…But That We Do Know

- Cannot afford a test system the size of Prod
- Anything short of Prod doesn’t accurately reflect reality
- Reality will surprise you and now the unknown is known

Software-Defined Software

Because We Are Software Developers

Change Needs To Be Fully Automated

And By Automation, We Mean Software

- Change needs to be applied at minimum latency
- There is absolutely no room for error
- Not exactly the ideal tasks for humans

How?

Reuse Ruthlessly

Decompose Vertically

Decouple Dramatically

Layer Horizontally

Everything Continuously

Reuse Ruthlessly

Decompose Vertically

Decouple Dramatically

Layer Horizontally

Everything Continuously

Stand On The Shoulders Of Giants

The View Is So Much Better Up There

- Developers have always known how to do this
  - Operating system, programming languages, libraries
  - Focus on the value that your code can add
- Today, we also reuse on the level of services
  - AWS has turned the datacenter into an API
  - I have not seen a datacenter in 8+ years

Don’t Reinvent The Wheel

Invent New Wheels

Don’t re-write Lucene

Don’t re-write messaging

Do create distributed indexing

Do write your own query engine

Reuse Ruthlessly

Reduce The Area Of Responsibility For Change

Reuse Ruthlessly

Decompose Vertically

Decouple Dramatically

Layer Horizontally

Everything Continuously

Identify The Main Functions Of The System

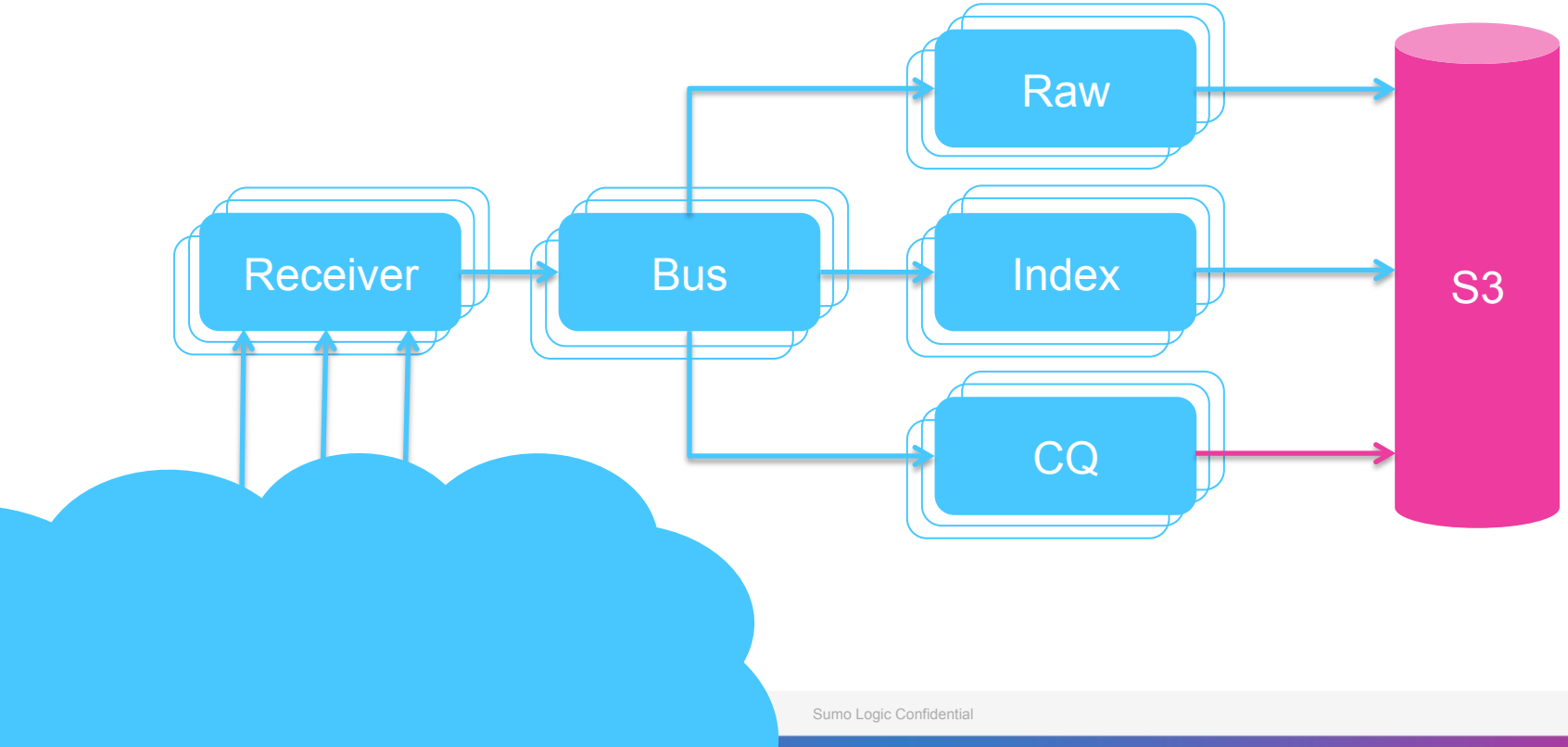
Then, Break Those Down Into Major Parts

We need to

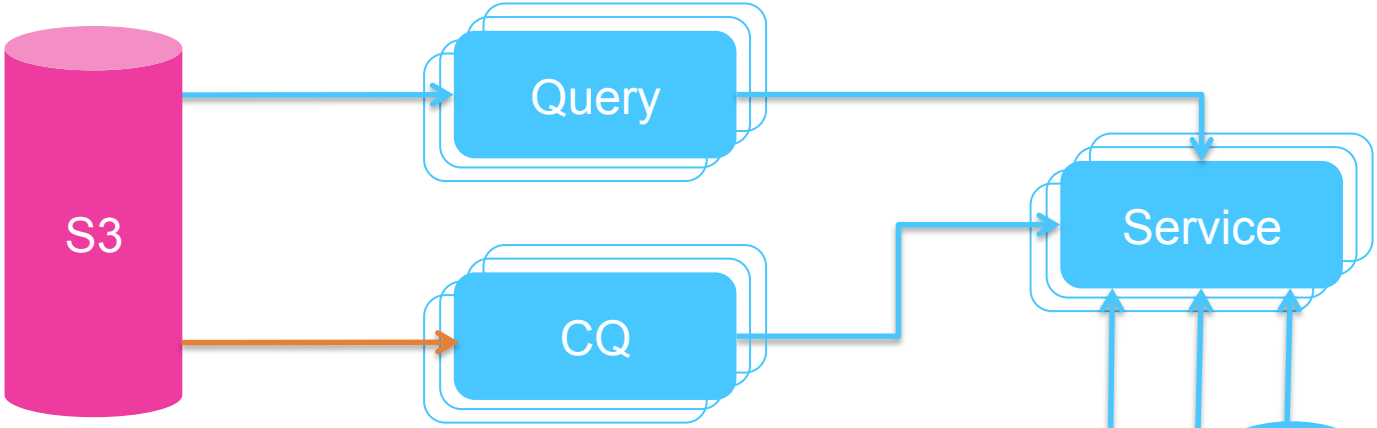
- Ingest data and index it so it can be queried
- Provide ad-hoc query capabilities for analytics
- Be able to update certain query results continuously

Also, shared stuff: Configuration, Encryption, API

Ingestion Path



Analytics Path



Decompose Vertically

Experts Will Emerge To Deal With Change In Any Area

Reuse Ruthlessly

Decompose Vertically

Decouple Dramatically

Layer Horizontally

Everything Continuously

Facade pattern

From Wikipedia, the free encyclopedia

The **facade pattern** (or **façade pattern**) is a software design pattern commonly used with object-oriented programming. The name is by analogy to an architectural facade.

A facade is an object that provides a simplified interface to a larger body of code, such as a class library. A facade can:

- make a software library easier to use, understand and test, since the facade has convenient methods for common tasks;
- make the library more readable, for the same reason;
- reduce dependencies of outside code on the inner workings of a library, since most code uses the facade, thus allowing more flexibility in developing the system;
- wrap a poorly designed collection of APIs with a single well-designed API (as per task needs).

The Facade design pattern is often used when a system is very complex or difficult to understand because the system has a large number of interdependent classes or its source code is unavailable. This pattern hides the complexities of the larger system and provides a simpler interface to the client. It typically involves a single wrapper class which contains a set of members required by client. These members access the system on behalf of the facade client and hide the implementation details.

Definitions [edit]

The OASIS group[4] and the Open Group[5] have both created formal definitions. OASIS defines SOA as:

> A paradigm for organizing and utilizing distributed capabilities that may be under the control of d
> means to offer, discover, interact with and use capabilities to produce desired effects consistent
> expectations.

The Open Group's definition is:

> Service-Oriented Architecture (SOA) is an architectural style that supports service-orientation.
>
> Service-orientation is a way of thinking in terms of services and service-based development and
>
> A service:
>
> - Is a logical representation of a repeatable business activity that has a specified outcome (e.g
> consolidate drilling reports)
> - Is self-contained
> - May be composed of other services
> - Is a “black box” to consumers of the service

Internal SOA

We Always Thought About It That Way Intuitively

- Now that you have things decomposed…
- …they can be decoupled!
- Avro over messaging bus, or RPC
- Documented protocols
- No poking at private parts


PAVLOV?
THAT NAME RINGS A BELL

I thought of objects
being like biological cells
and/or individual computers on a network,
only able to communicate with messages
(so messaging came at the very beginning)

– Alan Kay

Decouple Dramatically

You Will Have To Change Engines Mid-Flight

Reuse Ruthlessly

Decompose Vertically

Decouple Dramatically

Layer Horizontally

Everything Continuously

No Magic Here

This Is Just Obvious Best Practice

- Services can reuse code
  - Communication, configuration and utility libraries
  - Global functionality, service discovery, feature flags
- Service-level layering
  - Lower level utility services reused by higher level services
  - Can also work great if you want to support multiple implementation languages

Layer Horizontally

Always Try To Limit The Size Of Any Change

Reuse Ruthlessly

Decompose Vertically

Decouple Dramatically

Layer Horizontally

Everything Continuously

Continuous Integration

Continuous Delivery

Continuous Automation

Everything Continuously

Build A Change Delivery Highway That You Can Trust

Reuse Ruthlessly

Decompose Vertically

Decouple Dramatically

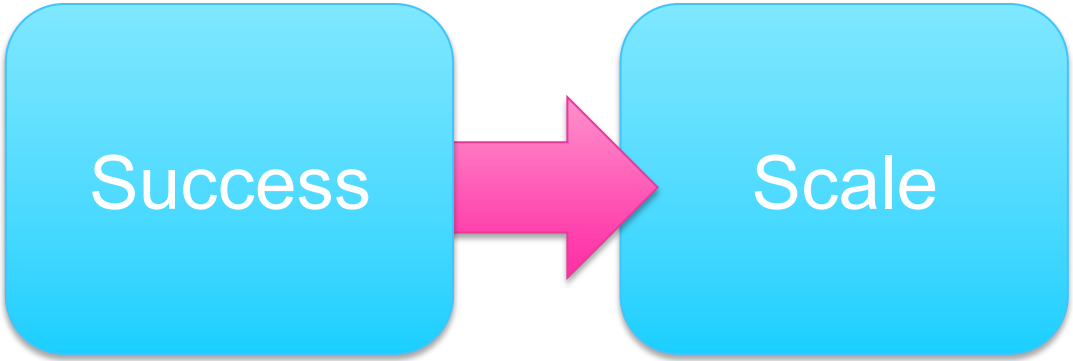
Layer Horizontally

Everything Continuously

Architect For An Extreme Rate Of Change

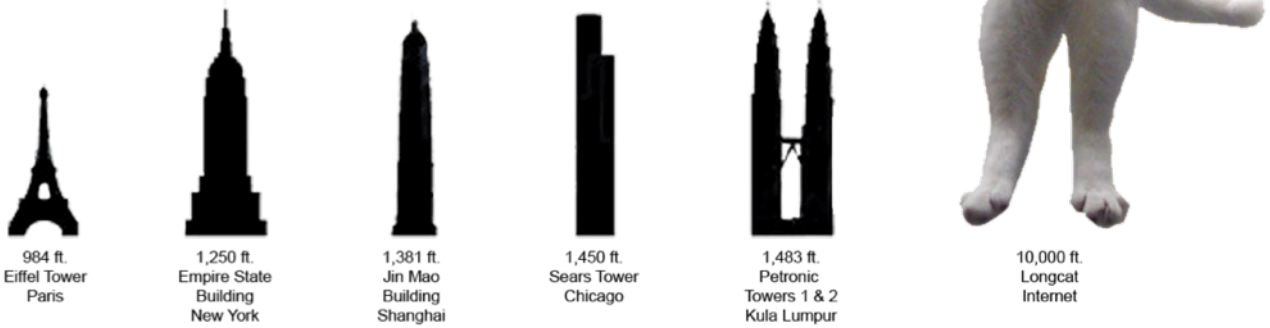
What We Have Learned

Decomposition Supports Scaling


Success
Scale

In **2010**, we knew that success will look something like this…

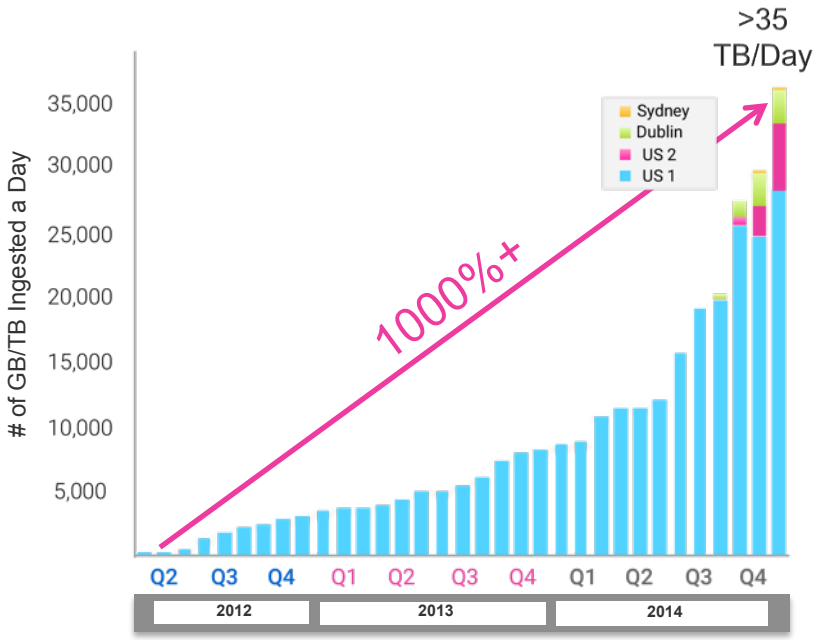
In **2010**, we knew that success will look something like this…



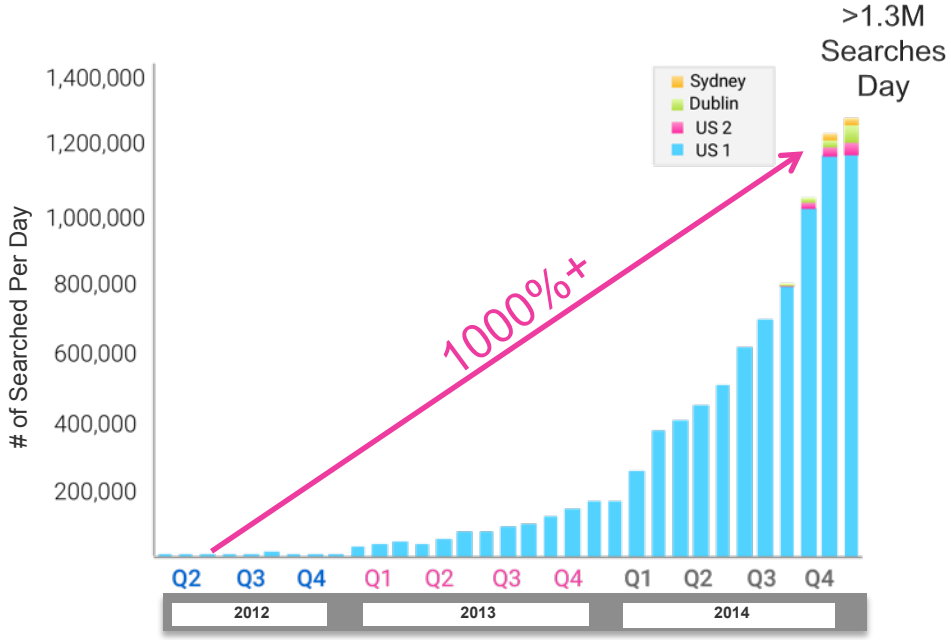
Our Service Momentum

Massive Data and Usage Growth

Daily Ingest



Searches



Decomposition Supports Scaling

We Have Absolutely Experienced Scale Induced By Success

- Full physical separation between services
  - Each node runs exactly one service
  - Each service is run on a cluster of nodes (N+2)
- Each service can scale independently
  - We require wildly different numbers of nodes based on the service
  - Some services can run on 3 nodes, some require 1000s

Software-Defined Software

Build, Run – It Is All The Same

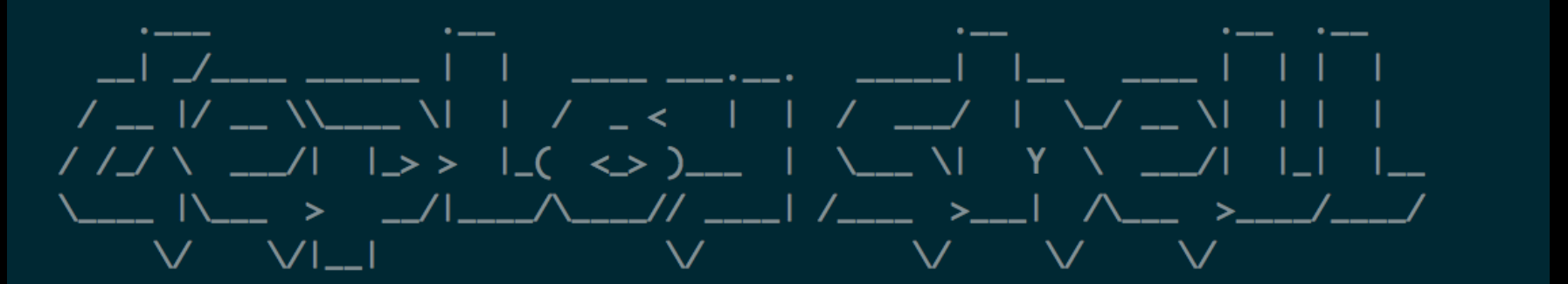
No Functionality, No Feature That Doesn't Need To Be Operated

- You don't just write code, you run a system
- You write more code to run the system
- We have all become system architects
- Deployment, operations – build by the Chief Architect
- There is more going on here than just config management

dsh

A Custom Command Line Program To Operate Sumo Logic

- Model-driven, describe desired state, run to make it so
- High performance due to parallelization
- Covers all layers of the stack – AWS, OS, Sumo Logic
- Easy to use and extend, scriptable CLI
- Developer-friendly, Scala-based, high-level APIs



```
  Tip: Find instance by tag using: instance search TagA=a&TagB=b

  Enter your commands below. Type 'help' for help.
[dsh-19.61-SNAPSHOT] $ dep sel prod
Selecting deployment 'prod'...
 Password for account 'production-master' (/Volumes/IronKeyBackup/sumo-accounts/p
Account production-master loaded from /Volumes/IronKeyBackup/sumo-accounts/produc
Configuring http logging to 'https://long-events.sumologic.net/receiver/v1/http/Z
331 running instances.

[dsh-19.61-SNAPSHOT] /production-master/prod$ inst terminate index-5
Terminate the following instances
 prod:index-5                    (i-0d331c69)
Proceed [Y/N]:
```

Deployments Are Model-Driven

Sie Ist Ein Model & Sie Sieht Gut Aus

- Model contains concepts
  - Deployment
  - Cluster
  - AWS Resources (Amazon S3, Amazon Elastic Load Balancing, Amazon DynamoDB, Amazon RDS, etc.)
  - Software assemblies
  - AWS configuration (IAM users, security groups, etc.)
- Human-readable names: `prod-index-5`

```
<role description="ZooKeeper node" name="zookeeper">
  <instanceType>m1.medium</instanceType>
  <amiId>ubuntu-ebs-amd64</amiId>
  <sshUser>ubuntu</sshUser>
  <properties>
    <property name="zookeeper_binary_url" value="s3_url:zookeeper-${zookeeper_binary_version}.tar.gz"/>
    <property name="depman_binary_url" value="ass-url:depman"/>
    <property name="depman_jvm_min_memory" value="256"/>
    <property name="depman_jvm_max_memory" value="512"/>
    <property name="zookeeper_jvm_min_memory" value="1024"/>
    <property name="zookeeper_jvm_max_memory" value="1920"/>
  </properties>
  <assemblies>
    <assembly name="collector"/>
    <assembly name="health"/>
    <assembly name="gyoji"/>
    <assembly name="ganglia-monitor"/>
    <assembly name="depman"/>
    <assembly name="service_registry"/>
  </assemblies>
</role>
```

```
deployment_name = long
deployment_descriptor = compact_deployment
deployment_account_name = infrastructure
deployment_ssh_key = long-west-2012-03-26
deployment_region = us-west-1

receiver_url = https:\/\/collectors.sumologic.com

cluster.analytics.size = 2

cluster.frontend.size = 2
cluster.frontend.volume.gb = 3
cluster.frontend.set.collector.gigabytes = 15

cluster.katta.size = 54
cluster.katta.set.collector.gigabytes = 15

cluster.cq.size = 2

cluster.config.size = 1
cluster.config.volume.gb = 100
cluster.config.set.collector.gigabytes = 15
```

Differential Deployment

As Good As An Episode Of House, M.D.

- Start by finding existing resources
  - Use tagging where it is available
  - Name prefixes (“prod_xxx”) where it isn’t (security groups, IAM, …)
- Fix differences to model
  - Start “missing” instances
  - Change security group rules, missing IAM users
- Proceed with caution
  - Never delete anything that holds data
  - Amazon EBS, Amazon DynamoDB, Amazon S3, Amazon RDS

Programmable Infrastructure Is Real

Embrace, Evolve & Include It In The Architecture

Microservices

Monolithic vs Microservices

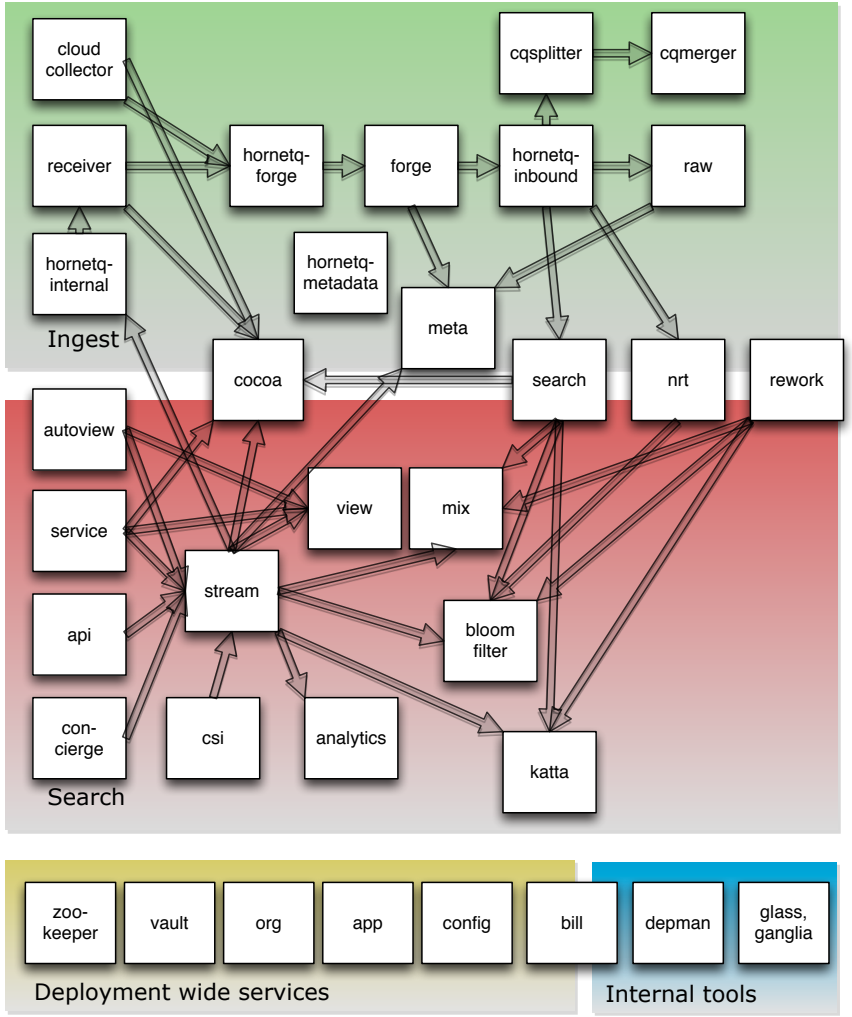
Monolithic

Microservices

How I Actually Visualize Microservices

Factoring Is Still Important

- Highly cohesive loosely coupled is an ideal
- The same ideal OO is striving for
- Here’s a snapshot of the current Sumo factoring



- 2 to the power of 5 services (“32”), 170+ modules
- Don’t even ask about the # of dependencies
- At least 3 of each – everything is a separately scalable cluster

Refactor-able Infrastructure

- The same old thing all over again
- Now that infrastructure is code, keep refactoring
- Split things that don’t belong, join others
- Service abstractions can help keeping impact low
- Moving around the code, vs. the protocols

Service Groups Scale

- This is really a level of granularity optimization
- One system: too heavy – 32 systems: too fleeting
- Build a service group, deploy against baseline, test
- During deployment, deploy by service group
- Balances crosscutting integration tests with turnaround

What Is Left To Do?

- Still a notion of a common version across all services
  - Weekly “major” releases, end of quarter release freezes
  - Even releasing one week of changes can perturb the force majorly
- True continuous delivery
  - Red/black deployments
  - How to do this in a system with a very high write rate?
- Tooling to support partial updates
  - Our own system is a great way to monitor and make decisions
  - Work in progress…

You Already Know How To Build Systems

Everything Now Has One More Layer Of Abstraction

There Is No Place Like Production

You Cannot Simulate The Big Datas

- There's simply no substitute for Production
- This doesn't mean you shouldn't have nightly, staging, …
- This doesn't mean you shouldn't have integration tests
- This doesn't mean you shouldn't test manually
- But there's just a class of issues you will not find

- You can't move Production data into testing
- You can't afford a second Production size system

So Now What?

- Instrument, instrument, instrument
- Monitor, monitor, monitor
- Alert on symptoms

https://docs.google.com/document/d/199PqyG3UsyXlwieHaqbGiWVa8eMWi8zzAn0YfcApr8Q/mobilebasic?pli=1&viewopt=127#h.dmn6k1rdb6jf

- Basically, don’t worry about 100% CPU, etc.
- Alert on customer impact
- Message ingestion delayed, search takes too long, …

Don’t Alert If You Don’t Have A Playbook

Stefan Zier
Chief Architect
Sumo Logic

Glass (prod) Overview Services Sumo Configuration Sumo Health Customer Information Dev Util Other System Release: 19.112-68 User: christian

Share Current Search **Preloaded Searches:** All Unhealthy - Recently Unhealthy

Showing 1 to 25 of 200 entries (filtered from 145,326 total entries - Clear Filters) (Export Data as CSV)

First Previous 1 2 3 4 5 Next Last Show 25 entries

Star...	Storage Time	unhealthy	Node	Service	Tracker	NOT(Scheduled maintenance) NOT(corrupted accounts) NOT(katta-sumo_marker)	Last Changed Time	Last Status Change	Documentation	Playbook	
Starred	**Storage Time**	**Status**	**Node**	**Service**	**Tracker**	**Message**	**Last Changed Time**	**Last Status Change**	**Documentation**	**Playbook**	
★	15-03-19 04:04 (16m26s ago)	unhealthy	prod-vault-1	vault	vault_index_corrupted	S3 read after write got different result (name: 000000000001EDD12/0000013E78A3C980, exists: false})	15-02-27 10:52 (19d16h28m52s ago)	15-03-19 04:04 (16m26s ago)	Turns unhealthy when After writing index we get different as a result. Likely double write occured.	vault_index_corrupted	al
★	15-03-19 04:04 (16m59s ago)	unhealthy	prod-index-1	search	katta_nodes_unassigned	Some katta nodes aren't assigned to any tier: katta- :20000	15-03-11 17:44 (7d10h37m12s ago)	15-03-19 04:04 (16m59s ago)	Turns unhealthy when some katta nodes aren't assigned to any tier	katta_nodes_unassigned	al
★	15-03-19 04:04 (16m59s ago)	unhealthy	prod-index-2	search	katta_nodes_unassigned	Some katta nodes aren't assigned to any tier: katta- :20000	15-03-11 17:43 (7d10h38m8s ago)	15-03-19 04:04 (16m59s ago)	Turns unhealthy when some katta nodes aren't assigned to any tier	katta_nodes_unassigned	al

Last Status Change	Documentation	Playbook	Sched...
Last Status Change	**Documentation**	**Playbook**	**Schedule**
15-03-19 04:04 (16m26s ago)	Turns unhealthy when After writing index we get different as a result. Likely double write occured.	vault_index_corrupted	always
15-03-19 04:04 (16m59s ago)	Turns unhealthy when some katta nodes aren't assigned to any tier	katta…s_unassig…	always

vault_index_corrupted

jakozaur edited this page on Dec 12, 2014 · 2 revisions

Ignore this alert on long 19.106. It's not a real issue.

When does this fire?

When after saving vault key we can't read back the same byte. It can be signal of some serious troubles.

It's a doomsday alert preventing data corruption, it should not fire at all.

What do I do?

It if is long 19.106 then ignore that.

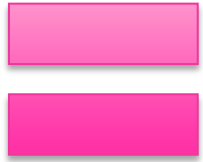
Otherwise contact SME.

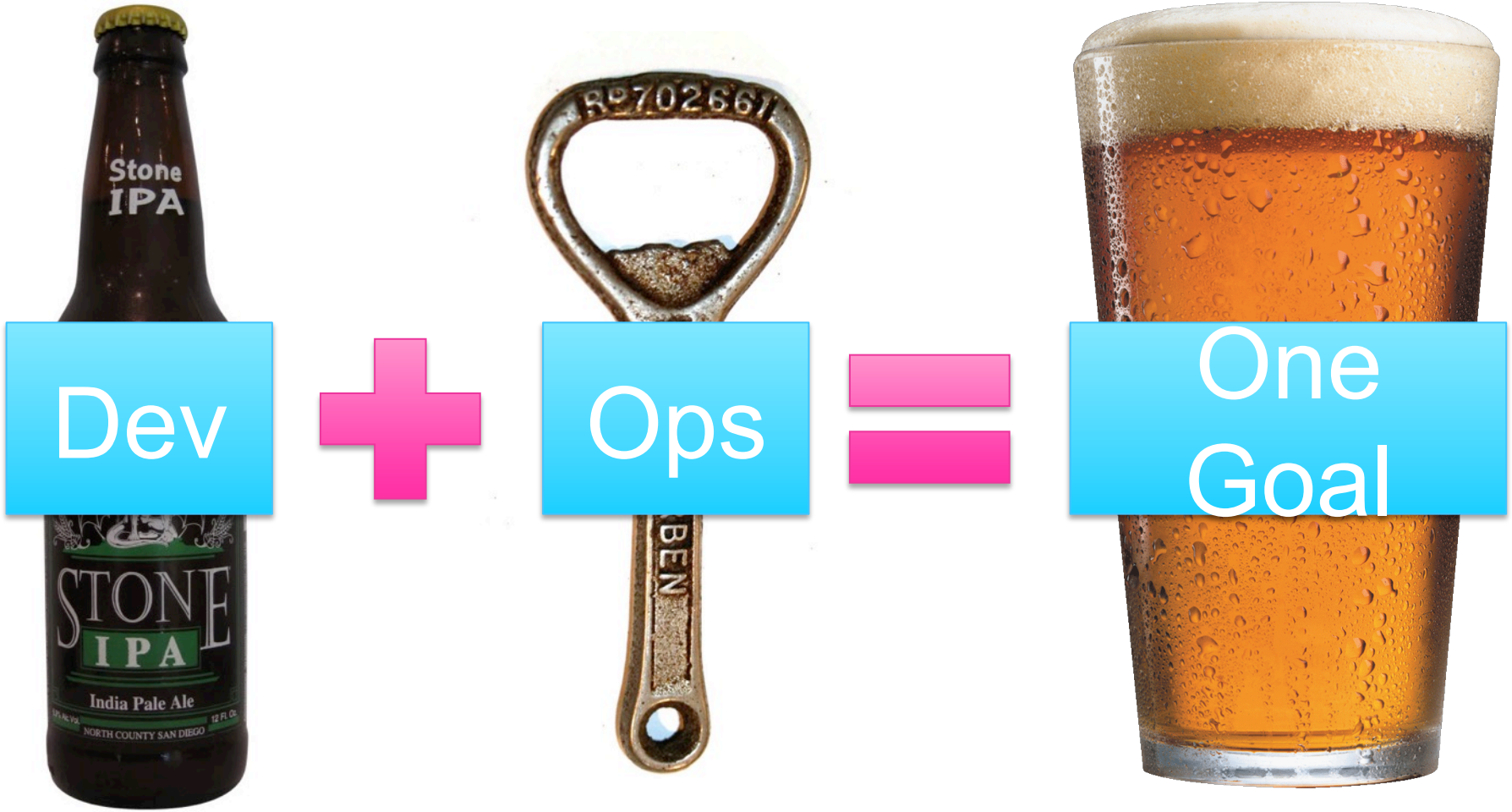
Start collecting what has happened:

1. Check in AWS S3 console that key. Does it have multiple versions? If so are they have the same key.
2. Check vault logs with that key name from health check. Watch out for double create.


Stone
IPA
STONE
IPA
India Pale Ale
NORTH COUNTY SAN DIEGO
12 FL OZ


THORBEN




Stone
IPA
STONE
IPA
India Pale Ale
NORTH COUNTY SAN DIEGO
Dev
Ops
One
Goal

It’s Not Just Another Layer Of Abstraction

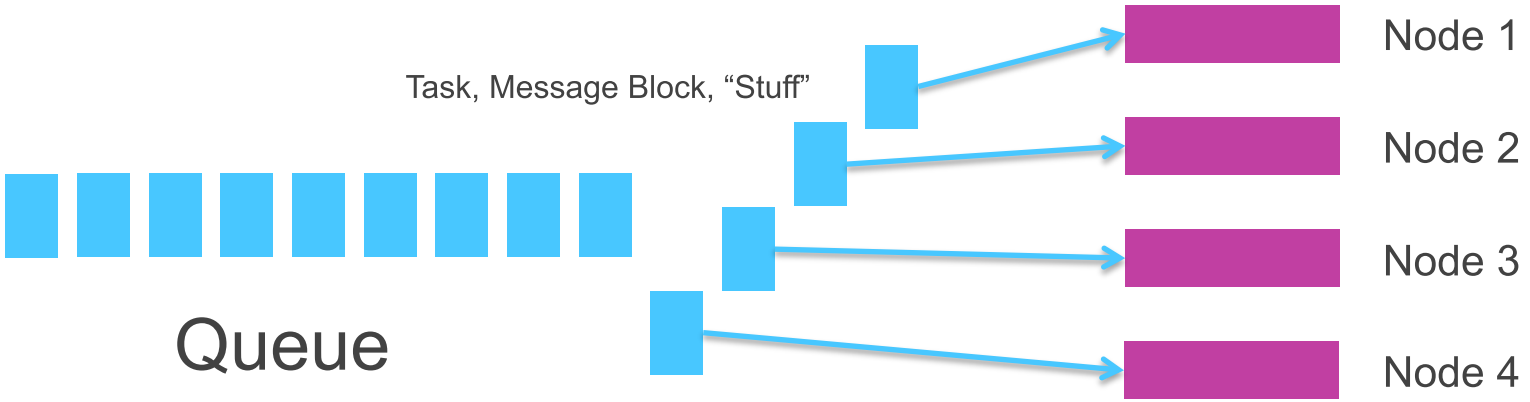
The Damn Thing Actually Is Always On

The Perils Of Horizontal Scaling

bad(N+2)

Horizontal Scaling Gone Bad

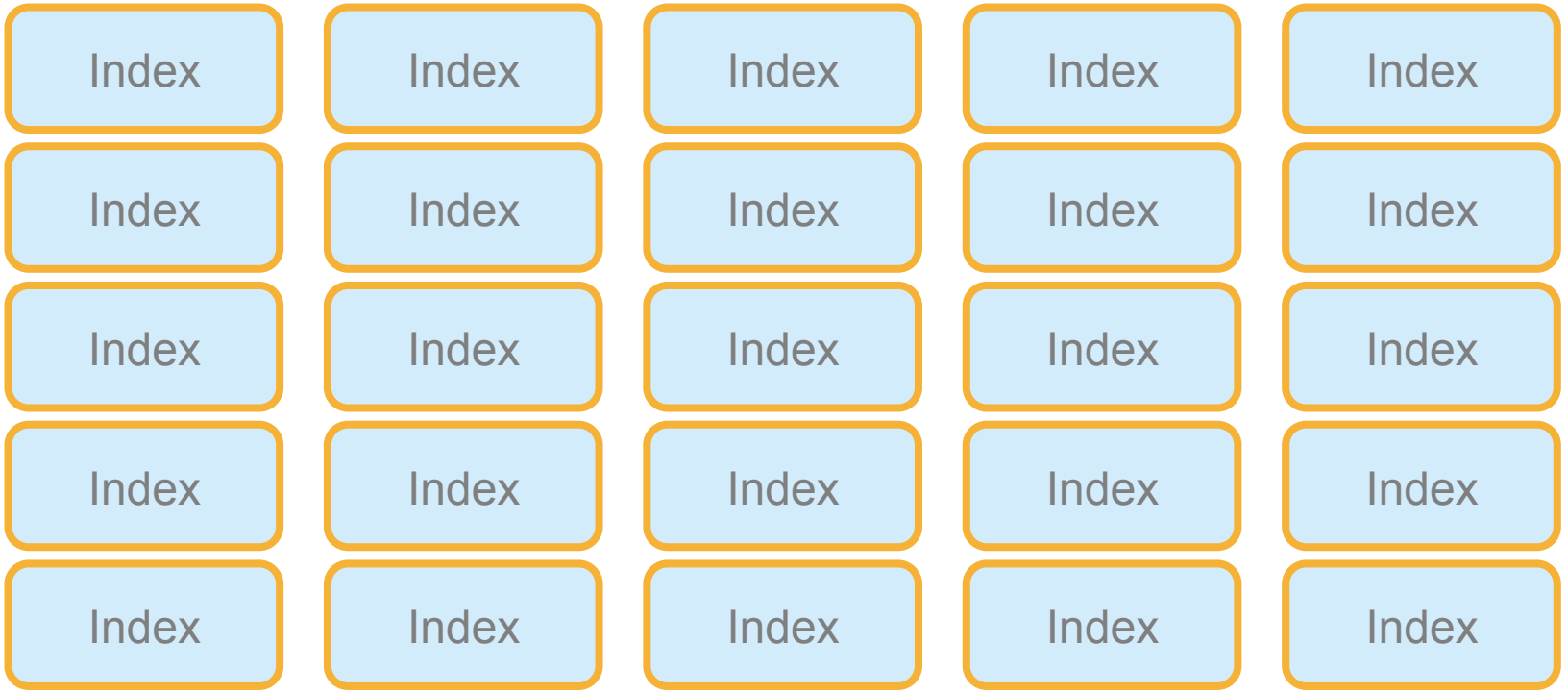
- The ideal scenario: work stealing
- One queue of tasks, bunch of workers
- Grab from queue, work work work, happy



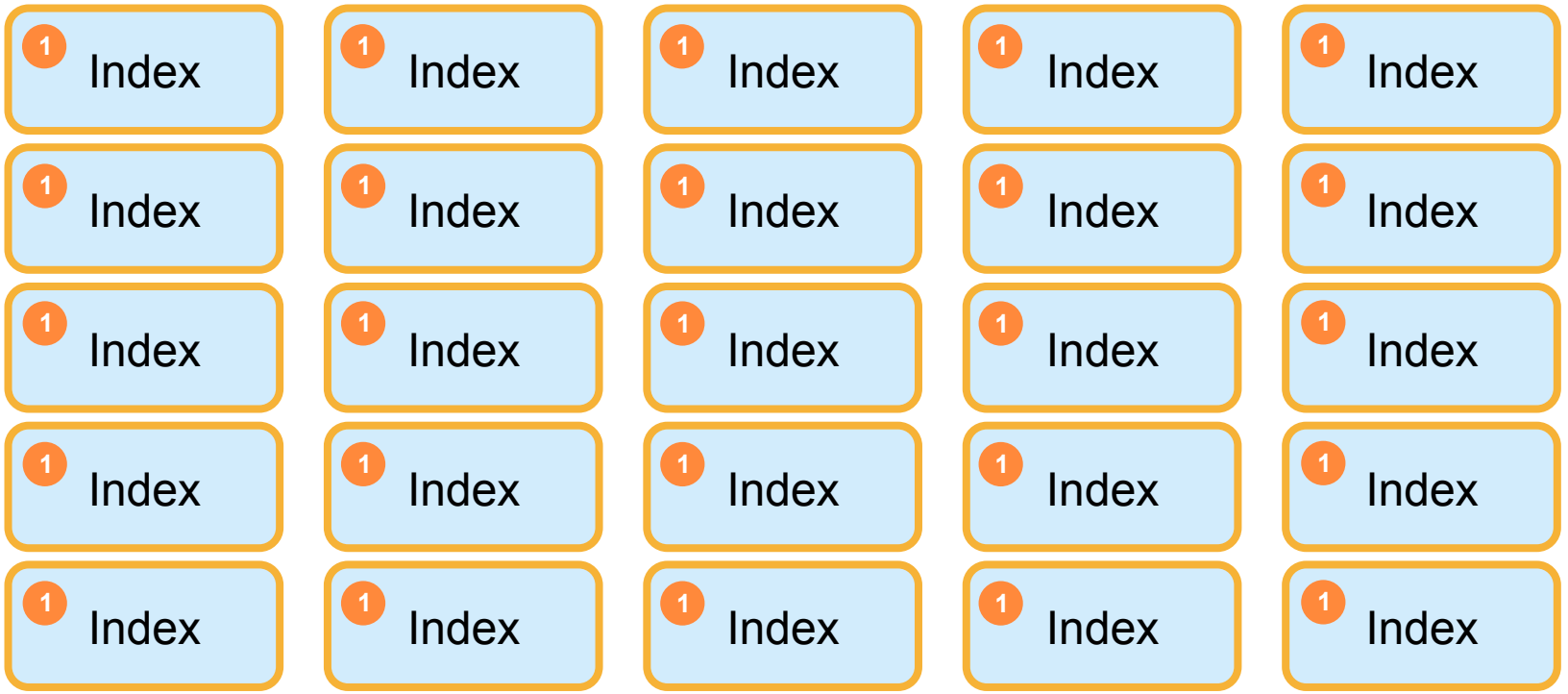
Horizontal Scaling Gone Bad

- Scaling out a multi-tenant processing system
- 1000s of customers, 1000s of machines
- Parallelism is good, but locality has to be considered
- 1 customer distributed over 1000 machines is bad
- No single machine getting enough load for that customer
- Batches & shards will become too small
- Metadata and in-memory structures grow too much

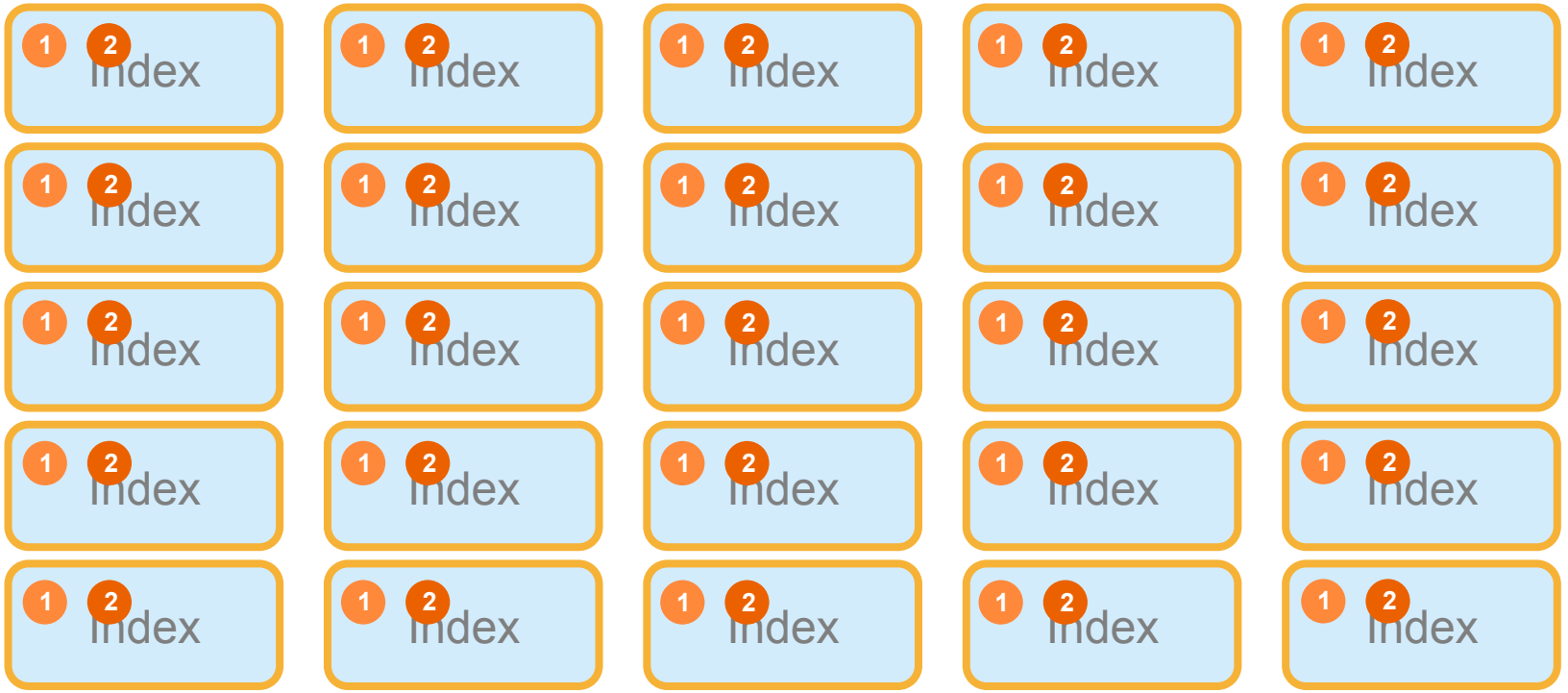
Horizontal Scaling Gone Bad



Horizontal Scaling Gone Bad

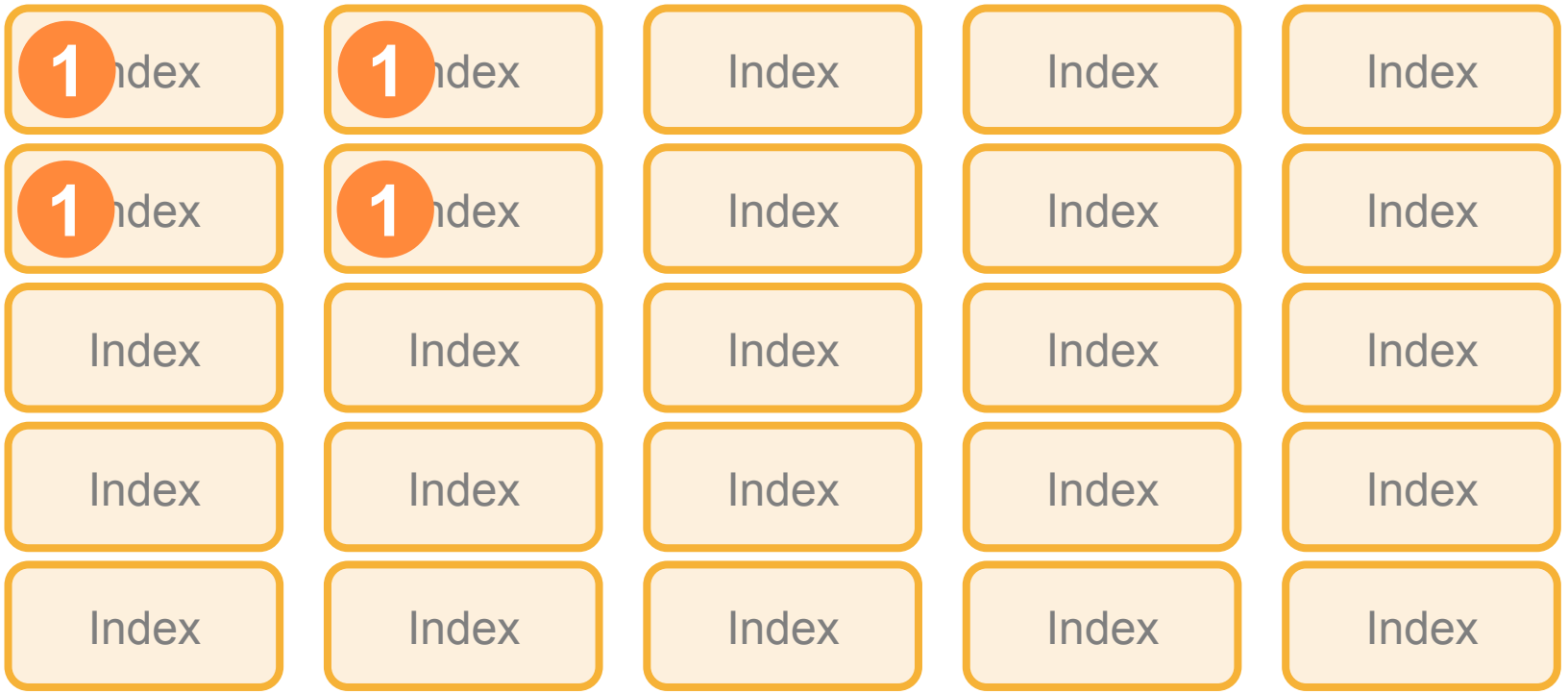


Horizontal Scaling Gone Bad

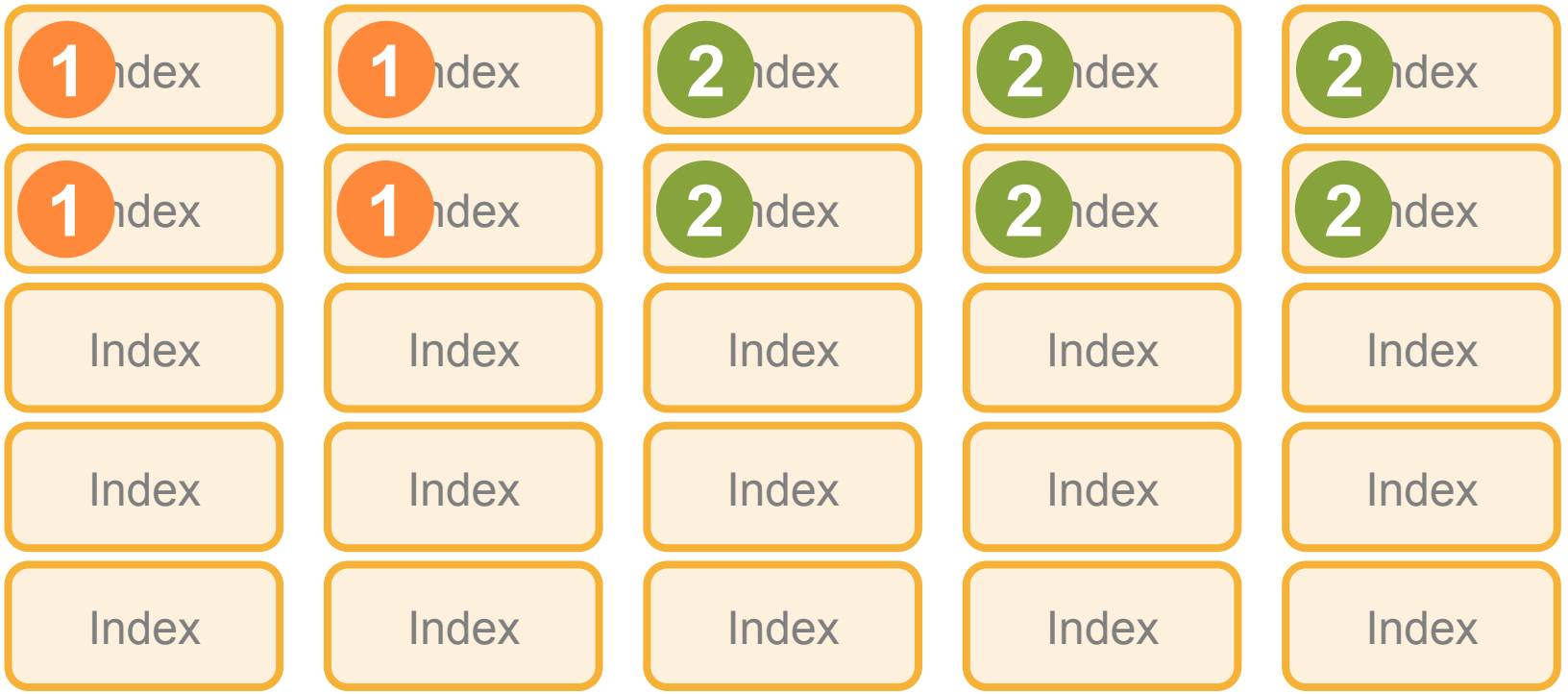


Horizontal Scaling Gone Bad

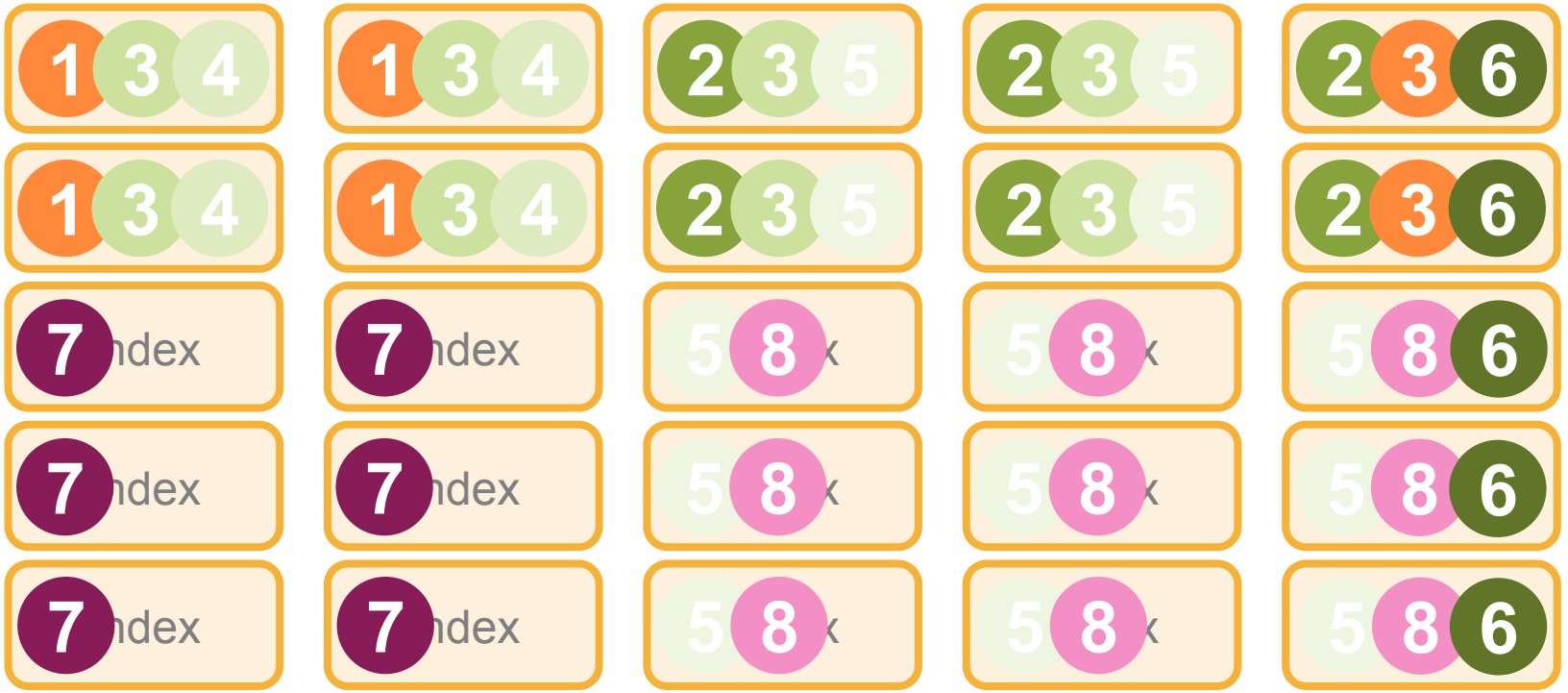
Horizontal Scaling With Partitioning



Horizontal Scaling With Partitioning



Horizontal Scaling With Partitioning



Partitioning By Customer

- Each cluster elects a leader node via Zookeeper
- Leader runs the partitioning logic

```
Set[Customer], Set[Instance] → Map[Instance, Set[Customer]]
```

- Partitioning written to Zookeeper
- Example: indexer node knows which customer's message blocks to pull from message bus

When Not To Scale (Without Bounds)

Less Is More, Locality Matters

Copy & Paste Scaling

So You Keep Adding Customers…

- Your current system is getting achy
- Next order of magnitude on the horizon
- You start to understand what you need to rebuild

- Your quarter ends in 21 days…

Sometimes, You Need To Break The Rules

- Copy & Paste ScalingTM
- Copy your deployment descriptor files & metadata
- Point them at a different region and pull the trigger
- Instant 2x scaling!

Chose Your Battles Wisely

Sometimes, Pragmatism Will Win Over Fundamentalism

Fin

@raychaser

Nostalgia For The Future

The Limits Of Physical Separation

- Every service runs on its own set of instances
  - We have consciously reinforced service decoupling by full physical separation
  - This was very important but we now have the discipline to keep things lose!
- Instances are right-sized for each service
  - Is this really the best approach for cost efficiency
  - Are we not using CPU on one cluster but heavy I/O and vice-versa on another?
- Denser packing and more dynamic placement
  - Just one type of instance plus Mesos, Kubernetes, etc. to schedule the processes?
  - Docker makes sense in this context, but we are JVM-based…

Data Is A Movement

- Heavily based around the idea of a physical pipeline
  - This causes an enormous amount of data movement
  - Should we be moving the data to the computation?
- Data movement logical hard as well in light of partitioning
  - Some of our systems attempt to partition based on load
  - Others use more static assignment of tenants to nodes in a cluster
- Locality for caches in light of partitioning
  - In-memory and ephemeral disk caches bound to instances
  - Dynamically adjusting resources much harder in this scenario

State-Based Auctioning

- Work-stealing based on a closed loop system
- Every instance is a data instance for memory and “disk”
- Every piece of data is tagged with tenant, etc.
- Clients don’t address RPCs to instances, just submit request
- Instances compete based on their local knowledge
- Caller will get a promise from the auction winning instance
- Ultimately caller will get the result
- Periodically, state across instances is centralized
- Quotas and limits are computed and distributed to instances
- Prevent over-distribution to maintain locality or cost-envelope

Assembly Details

Receiver

- HTTPS endpoint behind Elastic Load Balancing
- Decompress messages from Collector
- Extract timestamps from messages
- Aggregate messages per-customer into blocks
- Flush blocks to message bus
- Ack to Collector

Raw

- Receive message blocks from message bus
- Encrypt message blocks
- Different key for every day for every customer
- Flush encrypted message blocks to Amazon S3
- Copy blocks as CSV to customer’s Amazon S3 bucket
- Ack to message bus

Index

- Receive message blocks from message bus
- Cache message block on disk and ack to message bus
- Add message blocks to Lucene indexes
- Deal with wildly varying timestamps
- Flush index shards to Amazon S3
- Update meta data database with index shard info

Continuous Query

- Receive message blocks from message bus
- Evaluate each message against all search expressions
- Push matching messages into respective pipelines
- Ack to message bus
- Flush results periodically for pickup by client
- Persist checkpoints periodically to Amazon S3

Query

- Fully distributed streaming query engine
- Materialize messages matching search expression
- Push messages through a pipeline of operators
- First stage – non-aggregation operators
- Second stage – aggregation operators
- Present both raw message results as well as aggregates
- Results update periodically for interactive UI experience

Software-Defined Software

Making It Fast

- Parallelize all the things
  - Upload to Amazon S3 while booting instances while creating IAM users while setting up security groups while…
  - Hyper-concurrent rolling restarts
- Fast enough for development
  - Write new code or fix a bug, compile locally
  - Push code to development deployment and make it live
- Optimize data transfers
  - Use Amazon S3 hashes to only transfer new files
  - Only upload changed JARs

Making It Reliable

- Check prerequisites before you even try
  - Does Prod account have room for this many instances?
  - Do I have the required permissions for the AWS APIs?
  - Any model discrepancies I can't automatically resolve? Too many Amazon EBS volumes?
- Handle common failures automatically
  - No m1.large in us-east-1b? Move Amazon EBS volumes to us-west-1c and try there
  - Hitting the AWS API rate limit? Throttle and try again
  - SSH didn't come up on the instance? Kill it and launch another
  - Eventual consistency in AWS– query until it has the expected state (tags)

Making It Secure

- Different AWS accounts
  - Per developer
  - Production
- `account.xml`
  - All credentials for one AWS account (AWS keys, SSH keys)
  - Password-protected
- IAM
  - One user per Sumo component
  - Minimal IAM policy
  - Inject AWS credentials
- Security Groups
  - Part of the model
  - Minimal privileges

Making It Safe

- Let mistakes happen at most once
- Add safeguards to prevent operator mistakes
- Type in the deployment name before deleting anything
- Disallow risky operations in production (shutdown Prod)
- Don’t allow –SNAPSHOT code to be deployed in production

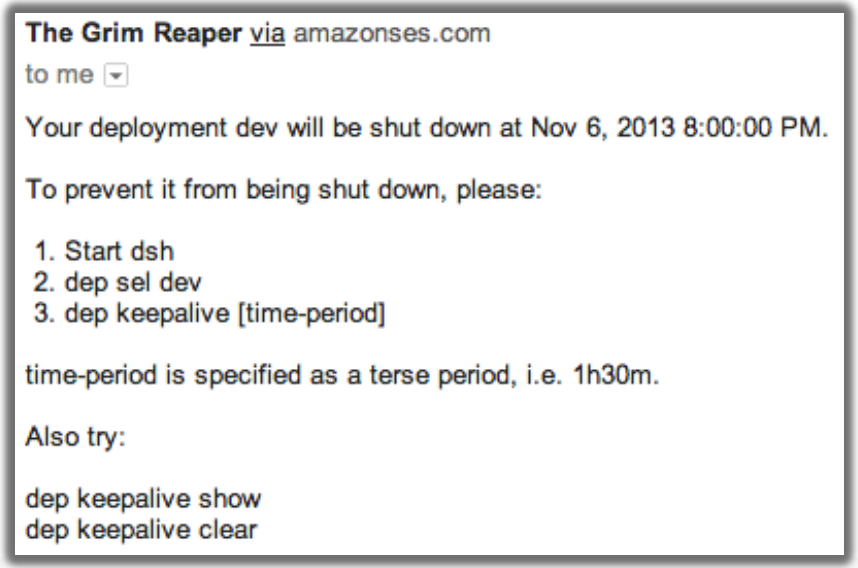
Making It Easy

- Automate best practices
  - Distribute instances over availability zones evenly
  - Register instances in Elastic Load Balancing and match AZs to instances
  - Tag all resources consistently
- Consistent naming
  - Generate SSH with logical names

```
Host prod-index-5
        HostName 54.242.
        UserKnownHostsFile=/dev/null
        IdentitiesOnly=yes
        StrictHostKeyChecking=no
```

Making It Affordable

- Developers forget to shut stuff down
  - Deployment reaper automatically shuts down deployments
  - Daily cost emails
- Per-team budgets
  - Manager responsible to keep within budget



The Grim Reaper via amazonses.com

to me

Your deployment dev will be shut down at Nov 6, 2013 8:00:00 PM.

To prevent it from being shut down, please:

1. Start dsh
2. dep sel dev
3. dep keepalive [time-period]

time-period is specified as a terse period, i.e. 1h30m.

Also try:

dep keepalive show
dep keepalive clear

Pitfalls

- Base AMI plus scripted installation prevents auto scaling
- Security group updates cause TCP disconnects
- This is fixed in the VPC stack, however
- Parallelism can cause stampedes (for example, Amazon DynamoDB)
- Tagging API rate limits are easy to hit