“You never truly understand the universe until you smash two atoms together”

– Some guy on Bill Mahr the other night

tl;dr

For time based data: Using memory mapped files with Log Structured Merge trees and hyper-efficient key and data schema design, it is possible to achieve 200,000 writes per second on commodity hardware

Continuum

Time x Space + Atom

Language

Language

Column Family

Binary Logs

Leader Election Algorithm

Cluster

Nested Query

Slave

Shard

Retention Policy

Write Amplification

ZooKeeper

Master

Site Config

Unique Constraint

Compaction

Replica Set

Read Amplification

Continuous Query

ZooKeeper

Consolidation Function

Language

Column Family

Binary Logs

Leader Election Algorithm

Cluster

Nested Query

Slave

Shard

Retention Policy

Write Amplification

Unique Constraint

Zookeeper

Master

Site Config

Compaction

Replica Set

Read Amplification

Continuous Query

Zookeeper

Consolidation Function

Language

Atom

Particles

Quark

Time

Space

Universe

Time Slice

Hot

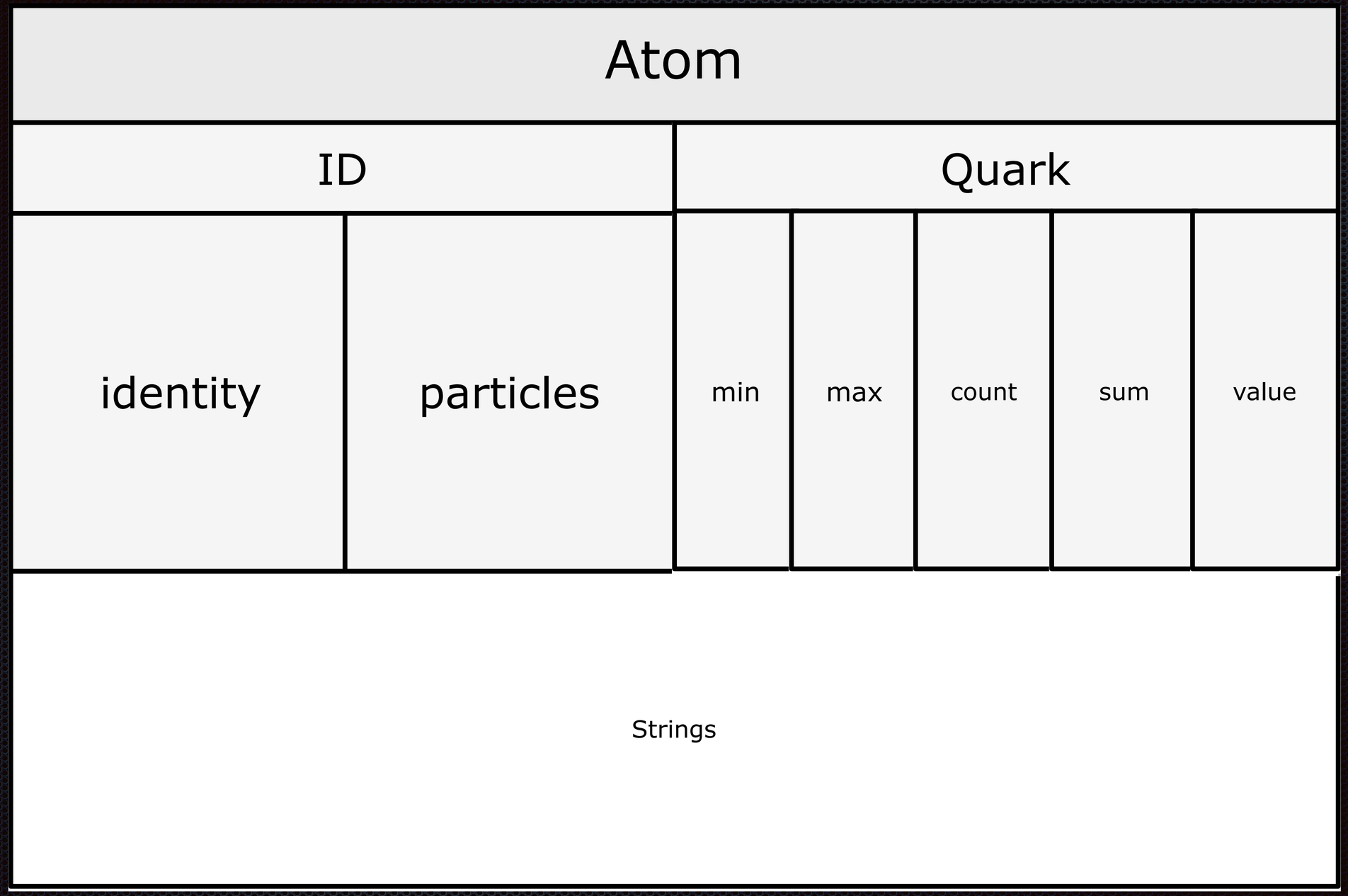
Cold

Glacier

Earthlings

~~*Bugs*~~ Spiders From Mars

Let’s Talk About Data


Atom
ID
Quark
identity
particles
min
max
count
sum
value
Strings

Let’s Talk About Data

Time Series - Single measurement over time

Let’s Talk About Data

Google

time series data

https://onlinecourses.science.psu.edu/stat510/node/47

All Images News Videos Shopping More Search tools

About 418,000,000 results (0.45 seconds)

Scholarly articles for **time series data**

Some properties of **time series data** and their use in … - Granger - Cited by 2359

Time series: **data** analysis and theory - Brillinger - Cited by 4419

Neural network **time series** forecasting of financial … - Azoff - Cited by 621

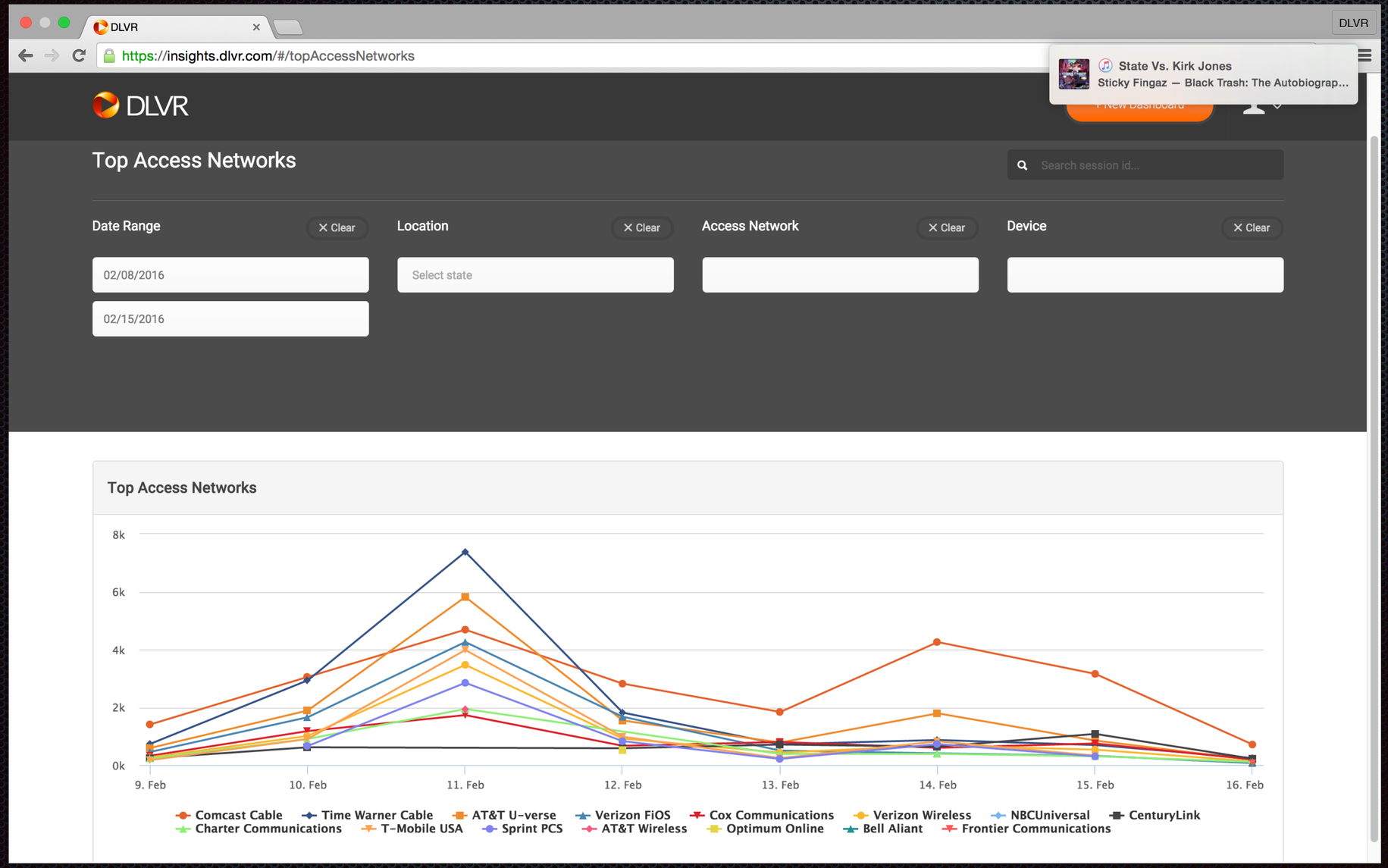
Definition: A univariate **time series** is a sequence of measurements of the same variable collected over **time**. Most often, the measurements are made at regular **time** intervals. One difference from standard linear regression is that the **data** are not necessarily independent and not necessarily identically distributed.

1.1 Overview of Time Series Characteristics | STAT 510

https://onlinecourses.science.psu.edu/stat510/node/47

Feedback

Let’s Talk About Data



Let’s Talk About Data

Time Series - Single measurement over time

Time Key ~~Value~~ - Related measurements over time

Let's Talk About Data

Google

time key value

All | Images | Shopping | News | Videos | More | Search tools

About 244,000,000 results (0.40 seconds)

NO

key-value store for time series data? - Stack Overflow
stackoverflow.com/questions/.../**key-value**-store-for-**time**-series-data
Nov 5, 2009 - I've been using SQL Server to store historical **time** series data for a ... It sounds like MongoDB would be a very good fit. Updates and inserts are ...

NO

Ask HN: Is there a key-time-value database? | Hacker News
https://news.ycombinator.com/item?id=672528 Hacker News
I'd also like to be able to retrieve the **key-values** for a segment of **time** in the past. ... This page has links to a number of **time** series database systems. ----- ...

Kinda, not really

Ask HN: Key-Value Store Database with Historical Lookup ...
https://news.ycombinator.com/item?id=8344115 Hacker News
Sep 20, 2014 - Does anyone know any **key-value** store database/service with ability to .... So, if the value of a key 'x' was 'y' at **time** '4', then you could do get(x, ...

Not even close!!

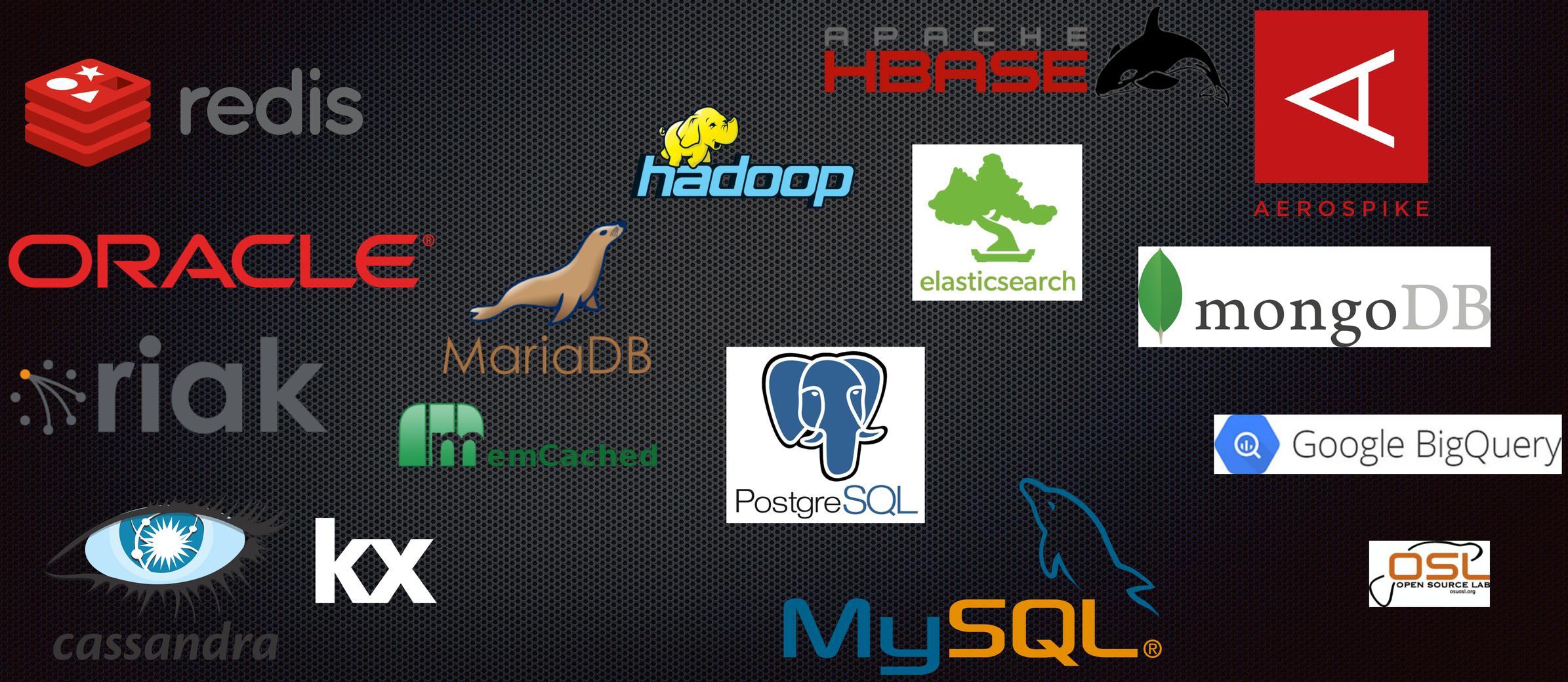
Patent US20130103658 - Time series data mapping into a ...
www.google.com/patents/US20130103658 Google
Apr 25, 2013 - A method for storing **time** series data in a **key-value** database includes receiving **time** series data relating to the occurrence of an event.

Troll!

Kinda

SET – Redis
redis.io/commands/SET Redis
Set **key** to hold the string **value** . If **key** already holds a **value**, it is overwritten, regardless of its type. Any previous **time** to live associated with the **key** is discarded ...

Want to Make a Database?



It Works!

Continuum.java - continuum-core - [~/Code/dlvr/continuum/continuum-core]

```java
public enum Dimension {
    SPACE, TIME
}

static class LoadTest {

    private final Continuum continuum;

    LoadTest(Continuum continuum) { this.continuum = continuum; }

    private final TimerTask timer = new MetricTimer().schedule(new Runnable() {
        @Override
        public void run() { System.out.println(Metrics.report()); }
    }, 5000);

    private static Atom createAtom() {
        return Continuum.satom()
                .name("series" + Maths.randInt(0, 100))
                .value(Maths.randDouble(Double.MIN_VALUE, Double.MAX_VALUE))
                .timestamp(System.currentTimeMillis())
                .build();
    }

    // 50MB/10M metrics
    public static void main(String[] args) throws Exception {
        FileSystemReference ref = new FileSystemReference("/tmp/LOAD");
        Builder b = series().base(ref);
```

Run Continuum$LoadTest

```
{write={count=193142411, mean_rate=201243.30849447634, mean=0.004680550949295433, median=0.004391, stdev=0.0016824343115439803, min=0.003657, max=0.040335, 99perc
{write={count=194160635, mean_rate=201255.1808211215, mean=0.004656570776674677, median=0.004377, stdev=0.001620669952874056, min=0.003646, max=0.040335, 99percen
{write={count=195180862, mean_rate=201269.18695676333, mean=0.004621476757415440 5, median=0.004354, stdev=0.001572756931305596, min=0.003646, max=0.040335, 99perc
{write={count=196188599, mean_rate=201271.01194582597, mean=0.004621920348841012, median=0.004335, stdev=0.0016230744177522499, min=0.003681, max=0.040335, 99perc
{write={count=197202259, mean_rate=201278.7350501656, mean=0.0046099856034321, median=0.00432, stdev=0.0015861841318842145, min=0.003607, max=0.040335, 99percenti
{write={count=198214457, mean_rate=201284.09778615343, mean=0.004589140131375 42, median=0.00431, stdev=0.0015634799915567124, min=0.003607, max=0.040335, 99percen
{write={count=199216768, mean_rate=201280.18643271917, mean=0.004586093187664203, median=0.004321, stdev=0.001525130169666449, min=0.003607, max=0.040335, 99perce
{write={count=200229392, mean_rate=201285.74081516571, mean=0.004571936656326582, median=0.004334, stdev=0.0015026497865150562, min=0.003607, max=0.040335, 99perc
{write={count=201242607, mean_rate=201292.77595731305, mean=0.004571138478974875, median=0.004327, stdev=0.0014685230561687233, min=0.003607, max=0.040335, 99perc
{write={count=202231165, mean_rate=201274.7771274693, mean=0.004574652117722131, median=0.004337, stdev=0.0014145063387866134, min=0.003607, max=0.040335, 99perce
```

200M — 201k/s — 62x better than prod

1. bash

```
Ozzie:~ zack$ du -h /tmp/LOAD
1.0G    /tmp/LOAD/continuum.0.db
1.0G    /tmp/LOAD
Ozzie:~ zack$
```

7 bytes overhead

vador.dlvr1.net:8080/api/1. × vador.dlvr1.net:8080/vitals/ ×

DLVR

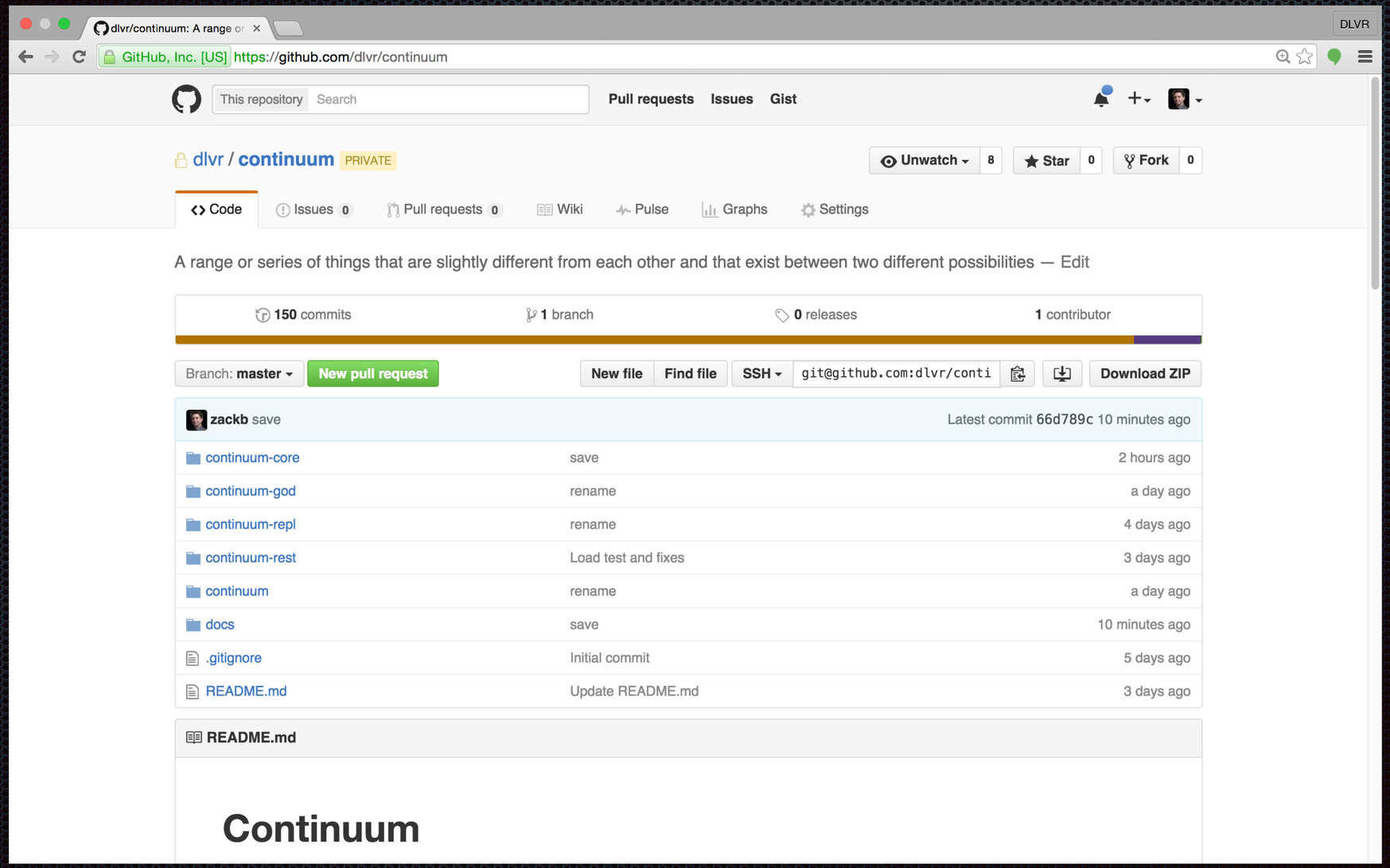
vador.dlvr1.net:8080/vitals/metrics

+ - View source


```
- nats.consumer.time: {
    count: 7779209,
    mean_rate: 35.70777342430476,
    mean: 0.15351877152486065,
    median: 0.095407,
    stdev: 0.15196565140687143,
    min: 0.044572,
    max: 1.28298,
    99percentile: 745390,
  - rate: {
        1m: 49.23820605591593,
        5m: 35.12324097342526,
        15m: 36.19935051537591
    }
},
- nats.producer.time: {
    count: 7779209,
    mean_rate: 35.7077721838476,
    mean: 0.006846830319100383,
    median: 0.003275,
    stdev: 0.03333083542177447,
    min: 0.001197,
    max: 0.954849,
    99percentile: 33801,
  - rate: {
        1m: 49.23832601855991,
        5m: 35.12364372816789,
        15m: 36.2000877360731
    }
},
```


Prod(1x SSD)

Want to Make a Database?



continuum-core-0.1.jar

```
Atom atom = atom()
        .name('temperature')
        .particles([location: 'lax'])
        .value(96.1)
        .timestamp(System.currentTimeMillis())
```

```
Continuum c = Continuum.continuum()
                .base(new File('/tmp/continuum'))
                .open()

DB db = c.getDb();

db.write(
    atom().name('temperature')
        .particles([location: 'lax'])
        .value(96.1)
        .timestamp(System.currentTimeMillis())
        .build()
)

Atom atom =  atom()
        .name('temperature')
        .particles([location: 'lax'])
        .value(96.1)
        .timestamp(System.currentTimeMillis())
        .build();

db.write(atom)
db.close()
c.close()
c.delete()
```

DLVR | continuum-core 0.1 API

file:///Users/zack/Code/dlvr/continuum/continuum-core/build/docs/javadoc/index.html

All Classes

Packages

com.dlvr.continuum
com.dlvr.continuum.atom
com.dlvr.continuum.core.atom
com.dlvr.continuum.core.db
com.dlvr.continuum.core.db.slice
com.dlvr.continuum.core.io.file
com.dlvr.continuum.db

com.dlvr.continuum.db

Interfaces

AtomID
DB
ID
Iterator
ParticlesID
Slab
SliceID

Method Detail

write

```
void write(Atom atom)
    throws java.lang.Exception
```

Write a datapoint to the time datastore

Parameters:

`atom - to write`

Throws:

`java.lang.Exception`

get

```
Atom get(AtomID id)
  throws java.lang.Exception
```

Retreive a single atom by id from the datastore

Parameters:

`id - to retrieve for`

Returns:

`atom for given id or null if not exist`

Throws:

`java.lang.Exception - on underlying slabs failure`

Read
GET /read
POST /write

Read

GET /read

Query time series or time key value data. Required fields are start, end, name. Optional values are tagname=tagvalue query parameters, interval for histogram responses, function aggregate to apply to the data.

example request

Get ten days worth of average temperature values in LAX in one day intervals.

```
$ curl -k -u dlvr:dlvr https://continuum.dlvr.com/api/1.0/read/temperature?
location=lax&start=0&end=10d&function=avg&interval=1d
```

response

```
{
  "data": [
    {12348567: 90.1},
    {12345567: 89.354678},
    {12346567: 92.9},
    {12347567: 90.0},
    {12348567: 90.0004},
    {12349567: 87.123},
    {12350567: 94.555},
    {12351567: 92.5},
    {12352567: 81.0},
    {12353567: 80.999},
  ],
  "execution_time": 2,
```

Want to Make a Database?

one, small, simple, tailored, codebase

200,000 w/s/ssd forever on my Mac

200,000 w/s/ssd forever on my Mac

How?

How?

The read problem

Throw horizontal scale out the window

ACID compliance is not a goal, out of scope

Limited query / SQL ability

Key and Data Design

Time Series

identity												particles					timestamp								fields					quark				
8	9	0	1	2	3	4	5		.	.	.	0	1	.	.	.	0	1	2	3	4	5	6	7	0	1	.	.	.	min	max	count	sum	value

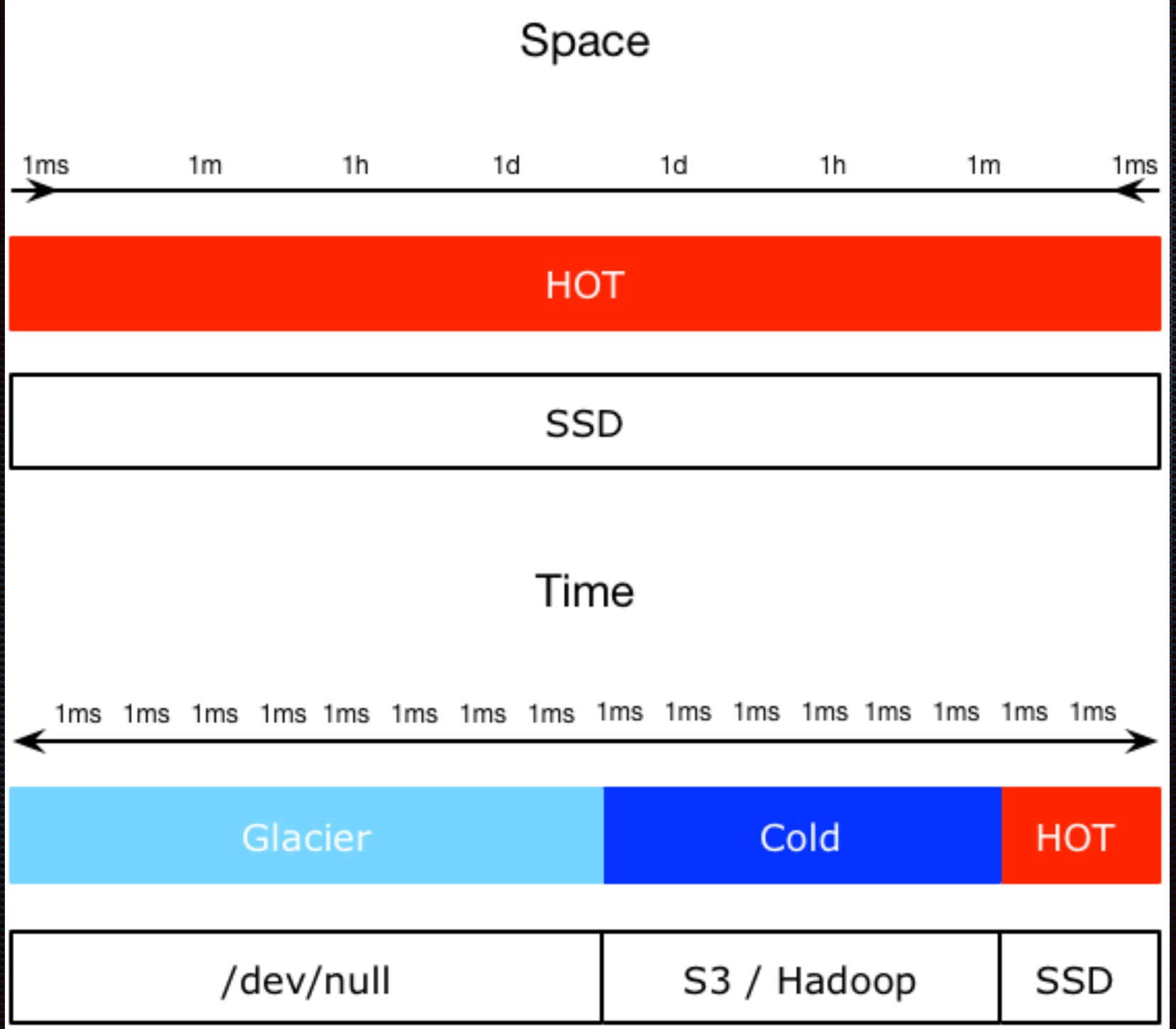
buffer_event.provider.network.limelight.231.1455570114342.[country=US,state=OR,client=ipad].2,2,1,2,2

Key and Data Design

Time Key ~~Value~~

timestamp								identity												particles					fields					quark				
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5		.	.	.	0	1	.	.	.	0	1	.	.	.	min	max	count	sum	value

1455570114342 : 942f7748-a531-11e5-842b-2e44683063cd.metrics.req_time.[prov=limelight, asn=7463, …].1337,1337,1,1337,1337


Space
1ms
1m
1h
1d
1d
1h
1m
1ms
HOT
SSD
Time
1ms 1ms 1ms 1ms 1ms 1ms 1ms 1ms 1ms 1ms 1ms 1ms 1ms 1ms 1ms 1ms
Glacier
Cold
HOT
/dev/null
S3 / Hadoop
SSD

vador.dlvr1.net:8080/api/1 | S3 Management Console | DLVR

https://console.aws.amazon.com/s3/home?region=us-west-2#&bucket=dlvr.prod&prefix=metric/2016/02/15/06/

AWS | Services | Edit | zack @ dlvr | Global | Support

Upload | Create Folder | Actions | Search by prefix | None | Properties | Transfers

All Buckets / dlvr.prod / metric / 2016 / 02 / 15 / 06

Name	Storage Class	Size	Last Modified
dal1-13.rdb1.zip	Standard - Infrequent Access	1.8 MB	Sun Feb 14 23:48:12 GMT-800 2016
iad1-13.rdb1.zip	Standard - Infrequent Access	1.5 MB	Sun Feb 14 23:48:24 GMT-800 2016
lax1-13.rdb1.zip	Standard - Infrequent Access	2.4 MB	Sun Feb 14 23:48:09 GMT-800 2016
lga1-13.rdb1.zip	Standard - Infrequent Access	1.7 MB	Sun Feb 14 23:47:44 GMT-800 2016
ord1-13.rdb1.zip	Standard - Infrequent Access	1.7 MB	Sun Feb 14 23:48:06 GMT-800 2016
sjc1-13.rdb1.zip	Standard - Infrequent Access	2 MB	Sun Feb 14 23:13:17 GMT-800 2016

Continuum Contracts

ID	Particles	Quark				
		min	max	count	sum	value
AAPL	EXCH:NYSE	289.10	289.10	1	289.10	289.10
		min	max	count	sum	value
AAPL	EXCH:NYSE	301.12	301.12	1	301.12	301.12
		min	max	count	sum	value
AAPL	EXCH:NYSE	347.10	347.10	1	347.10	347.10
		min	max	count	sum	value
AAPL	EXCH:NYSE	321.71	321.71	1	321.71	321.71
Continuum Contracts						
		min	max	count	sum	value
AAPL	EXCH:NYSE	289.10	347.10	4	1259.03	314.7575

Space

Constant storage space requirement

1455570114342 : 942f7748-a531-11e5-842b-2e44683063cd.metrics.req_time.[prov=limelight, asn=7463, …] .1337,1337,1,1337,1337

1455570134342 : 942f7748-a531-11e5-842b-2e44683063cd.metrics.req_time.[prov=limelight, asn=7463, …] .1338,1338,1,1338,1338

1455570154342 : 942f7748-a531-11e5-842b-2e44683063cd.metrics.req_time.[prov=limelight, asn=7463, …] .1336,1336,1,1336,1336

Fusion

1455570114342 : 942f7748-a531-11e5-842b-2e44683063cd.metrics.req_time.[prov=limelight, asn=7463, …] .1336,1338,3,4011,1337

Fusion

```
class BufferEventFusion implements Fusion {
        Atom fuse(Atoms atoms)
}
```

Fission

```
class KeynameCounter implements Fission {
        Atoms fiz(Atom atom)
}
```

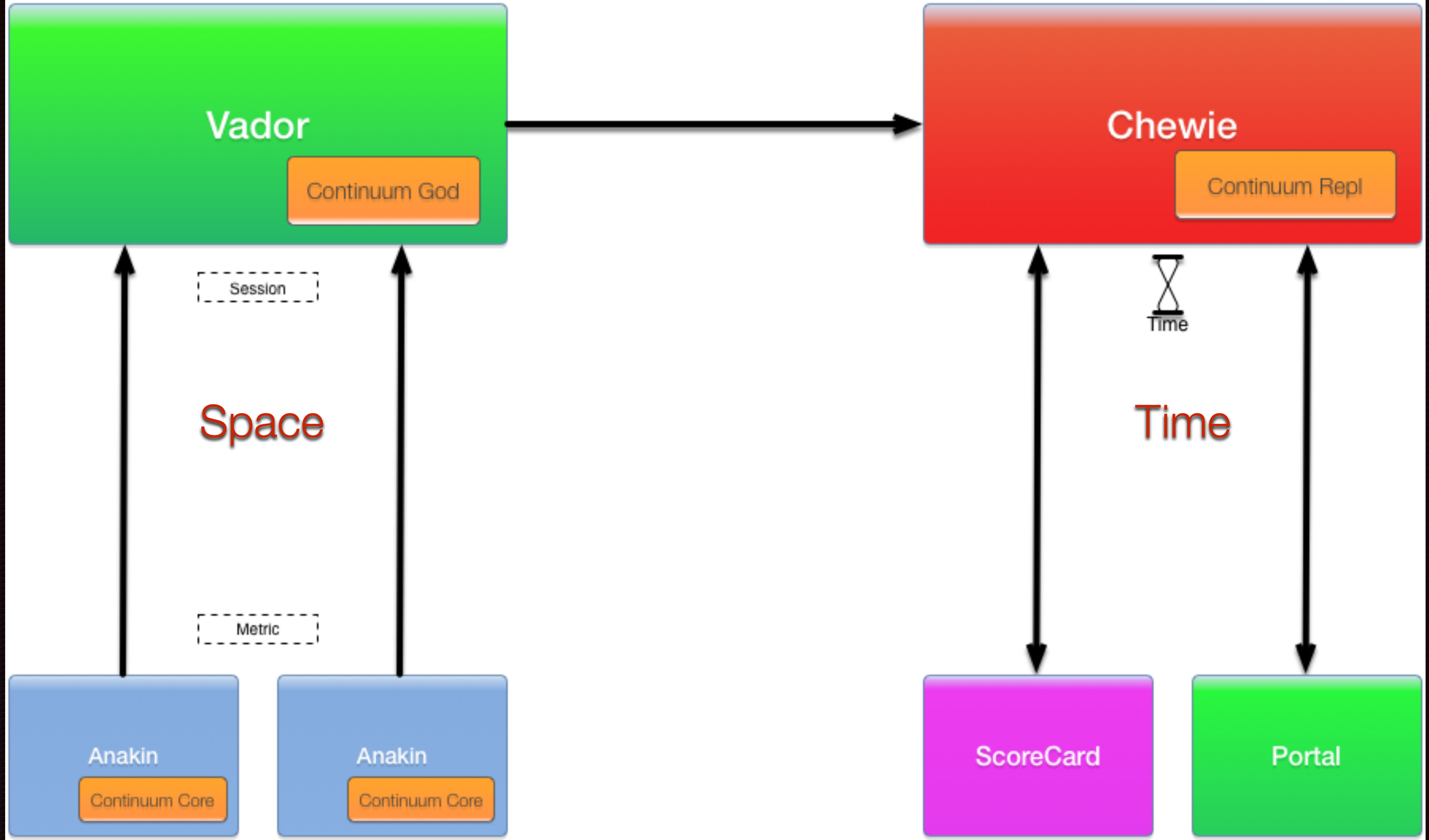
Scheduling

```
while (true) {
    s = query_sessions(end)
    m = query_metrics(s)
    rollup(s, m)
    sleep(1)
}
```

blocking :(

Scheduling

```
while (true) {
    s = query_sessions(end)
    m = query_metrics(s)
    roll_up(s, m)
    sleep(1)
}
```


Vador
Continuum God
Chewie
Continuum Repl
Session
Time
Space
Time
Metric
Anakin
Continuum Core
Anakin
Continuum Core
ScoreCard
Portal

1M / sec

so what?

Simpsons Already Did It

vador.dlvr1.net:8080/api/1. × 1 million writes per second ×

https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=1%20million%20writes%20per%20second

Google 1 million writes per second

All Shopping Images News Videos More Search tools

About 3,020,000 results (0.37 seconds)

The Netflix Tech Blog: Revisiting 1 Million Writes per second
techblog.netflix.com/.../revisiting-**1-million-writes-per-second**.ht... Netflix
Jul 25, 2014 - In an article we posted in November 2011, Benchmarking Cassandra Scalability on AWS - Over a **million writes per second**, we showed how ...

One Million Writes / Sec | DataStax
www.datastax.com/**1-million-writes** DataStax
One Million Writes / Sec. Excerpt from "Benchmarking Cassandra Scalability on AWS – Over a million writes **per second**" by Adrian Cockcroft and Denis ...

But

vador.dlvr1.net:8080/api/1. | The Netflix Tech Blog: Revi | DLVR

techblog.netflix.com/2014/07/revisiting-1-million-writes-per-second.html

Node Count, Software Versions & Configuration

The C* Cluster

WTF??

The Cassandra cluster ran Datastax Enterprise 3.2.5, which incorporates C* 1.2.15.1. The C* cluster had 285 nodes. The instance type used was i2.xlarge. We ran JVM 1.7.40_b43 and set the heap to 12GB. The OS is Ubuntu 12.04 LTS. Data and logs are in the same mount point. The mount point is EXT3.

You will notice that in the previous test we used m1.xlarge instances for the test. Although we could have had similar write throughput results with this less powerful instance type, in Production, for the majority of our clusters, we read more than we write. The choice of i2.xlarge (an SSD backed instance type) is more realistic and will better showcase read throughput and latencies.

The full schema follows:

```
create keyspace Keyspace1
  with placement_strategy = 'NetworkTopologyStrategy'
  and strategy_options = {us-east : 3}
  and durable_writes = true;
```

Netflix ISP Speed Index
Open positions at Netflix
Netflix Website
Facebook Netflix Page
Netflix UI Engineering
RSS Feed

About the Netflix Tech Blog

This is a Netflix blog focused on technology and technology issues. We'll share our perspectives, decisions and challenges regarding the software we build and use to create the Netflix service.

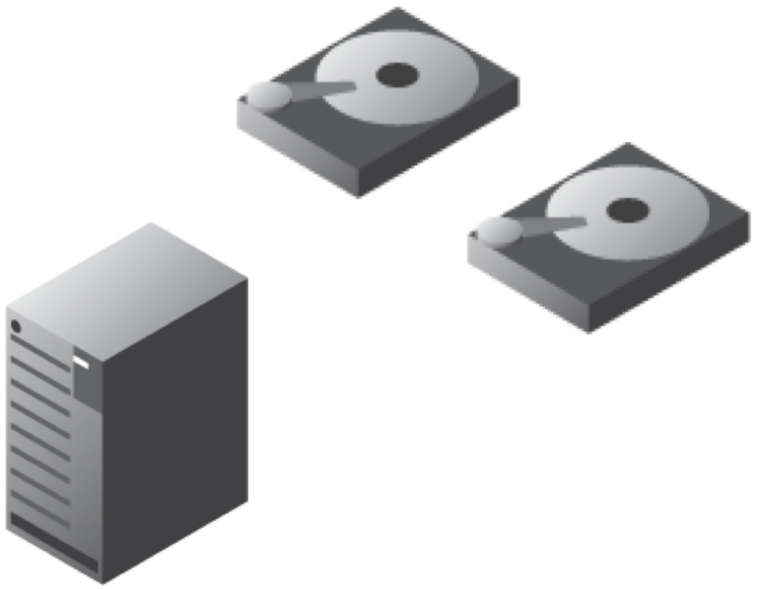
$243.105 / hour

i2.xlarge

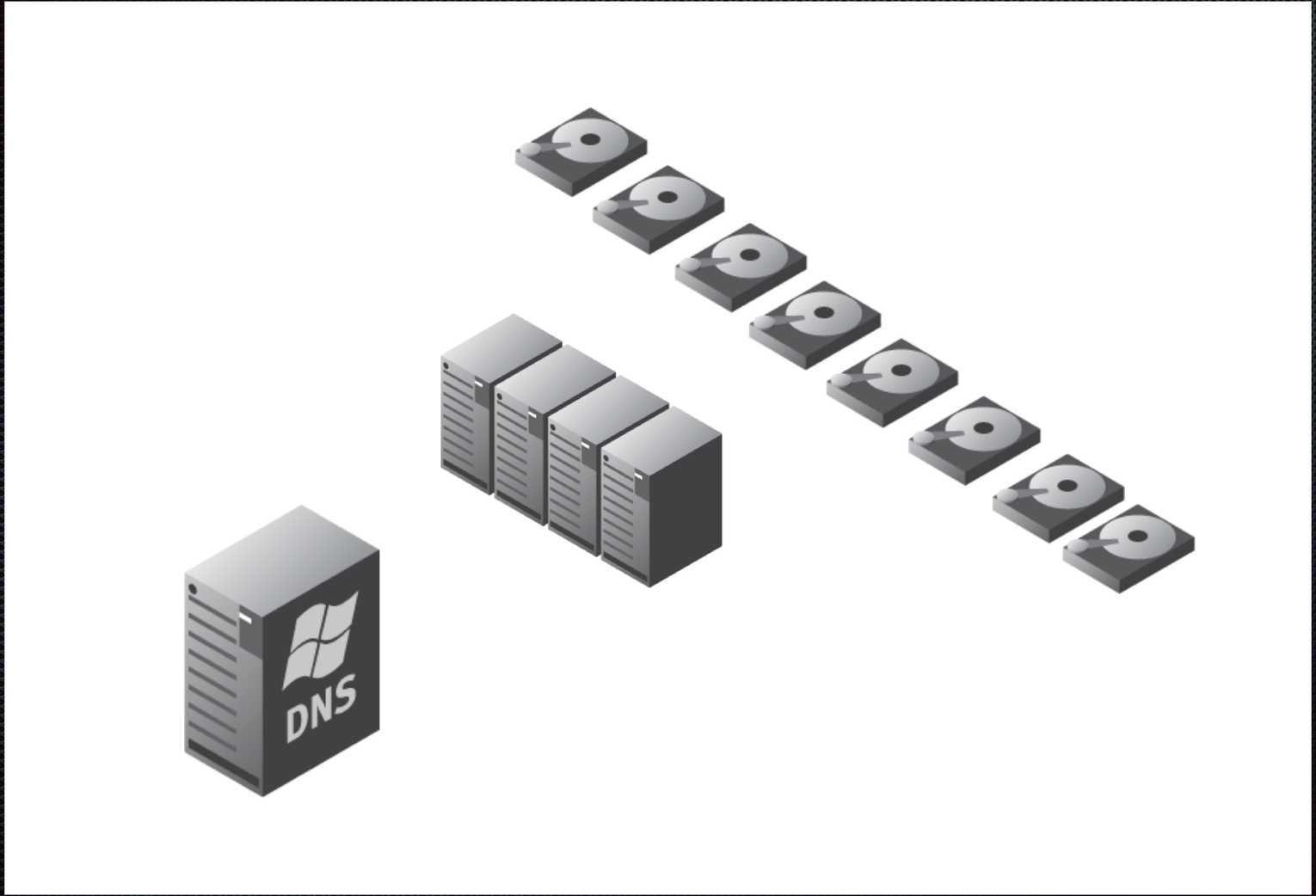
1-YEAR TERM					
Payment Option	**Upfront**	**Monthly***	**Effective Hourly****	**Savings over On-Demand**	**On-Demand Hourly**
No Upfront	$0	$309.52	$0.424	50%	$0.853 per Hour
Partial Upfront	$1820	$113.15	$0.3628	58%	
All Upfront	$3114	$0	$0.3555	58%	

3-YEAR TERM					
Payment Option	**Upfront**	**Monthly***	**Effective Hourly****	**Savings over On-Demand**	**On-Demand Hourly**
Partial Upfront	$2740	$88.33	$0.2253	74%	$0.853 per Hour
All Upfront	$5564	$0	$0.2117	75%	

“Horizontal” Scale



“Horizontal” Scale



“Horizontal” Scale

Traditional Application Level Sharding

Java?

(long story…)

Java?

Cheap

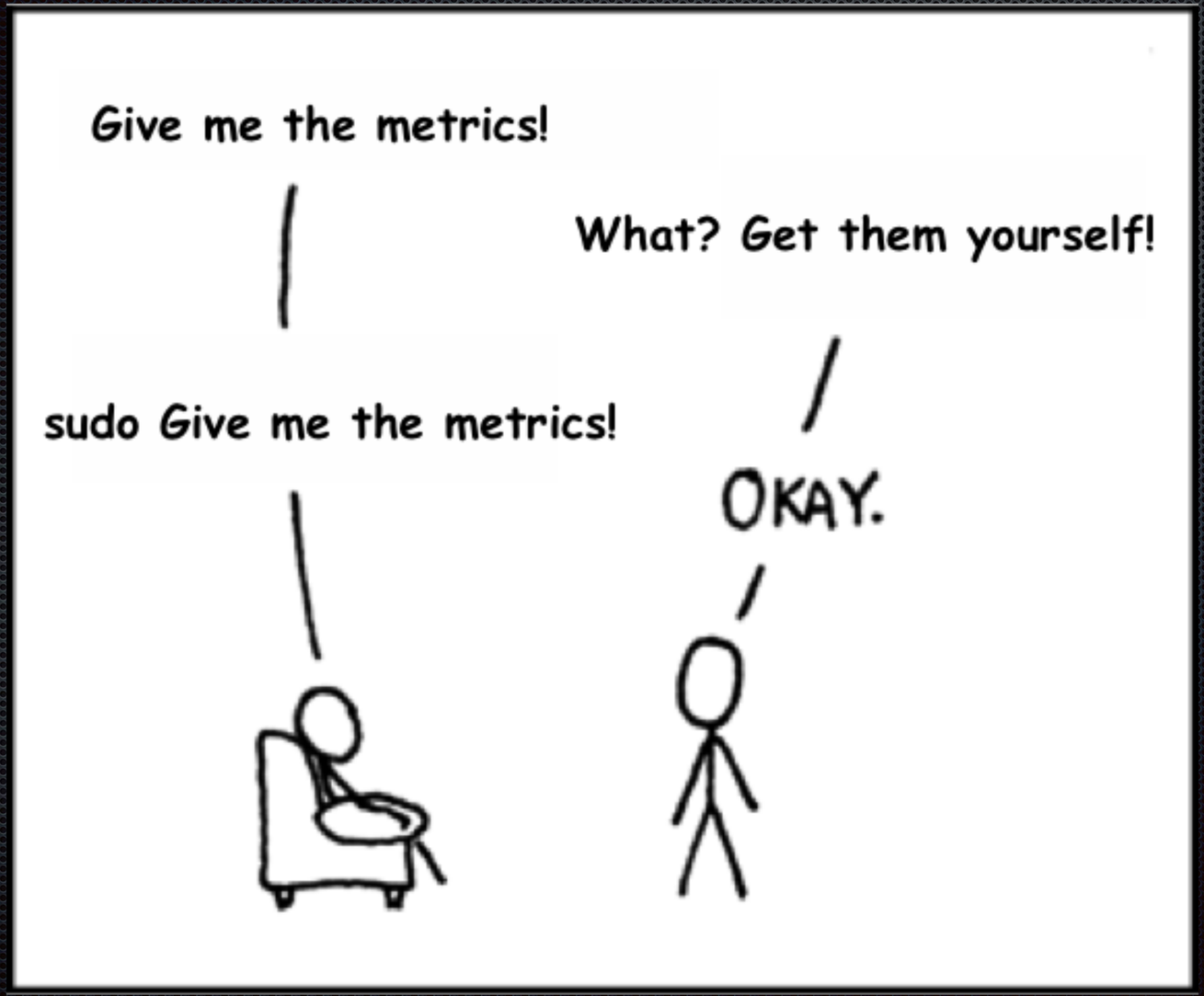
The Metrics Monster

The read problem

The Metrics Monster

read performance :(

The Metrics Monster



The Metrics Monster

while (true) {

}

The Metrics Monster

The Metrics Monster

The Metrics Monster

The Metrics Monster

We’ll get to that

almost done I promise

The Best Part

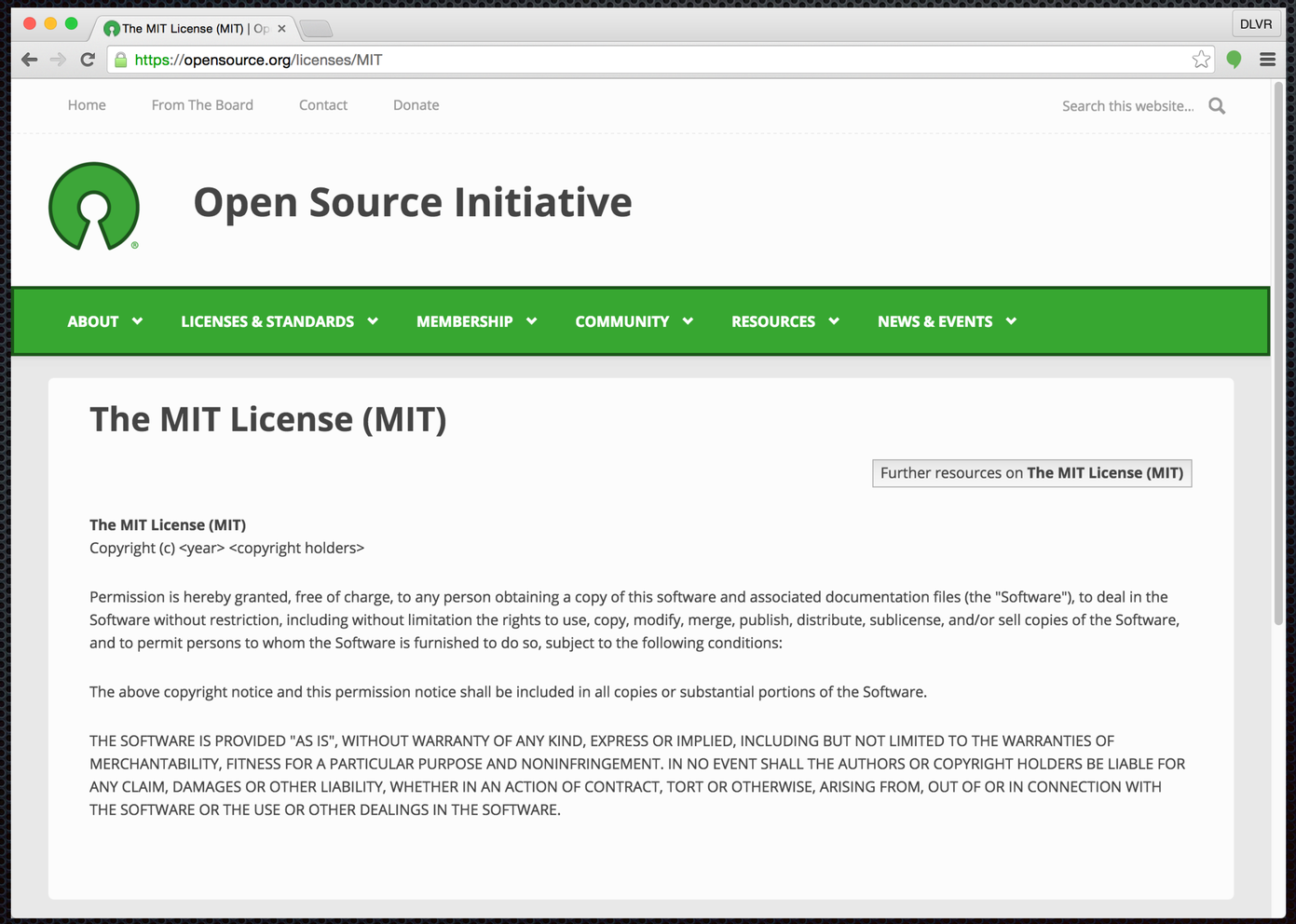
Continuum Core is Free

Continuum Core is Free

Under the MIT License

Continuum Core is Free

(pretty please? :)



The Best Part

Why struggle to hire when the best in the world will do it for free?

ZABBIX

OSL
OPEN SOURCE LAB

One more thing…

One more thing…

The read problem

One more thing…

The read problem

One more thing…

timestamp								identity												particles					fields					quark				
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5		.	.	.	0	1	.	.	.	0	1	.	.	.	min	max	count	sum	value

identity												particles					timestamp								fields					quark				
8	9	0	1	2	3	4	5		.	.	.	0	1	.	.	.	0	1	2	3	4	5	6	7	0	1	.	.	.	min	max	count	sum	value

One more thing…

timestamp								identity												particles					fields					quark				
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5		.	.	.	0	1	.	.	.	0	1	.	.	.	min	max	count	sum	value

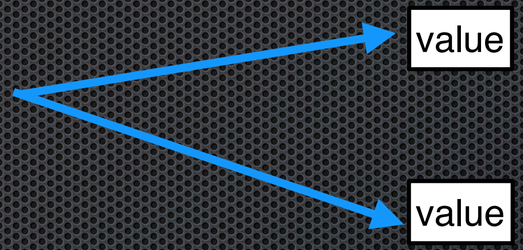
the last 64 bits

identity												particles					timestamp								fields					quark				
8	9	0	1	2	3	4	5		.	.	.	0	1	.	.	.	0	1	2	3	4	5	6	7	0	1	.	.	.	min	max	count	sum	value

One more thing…

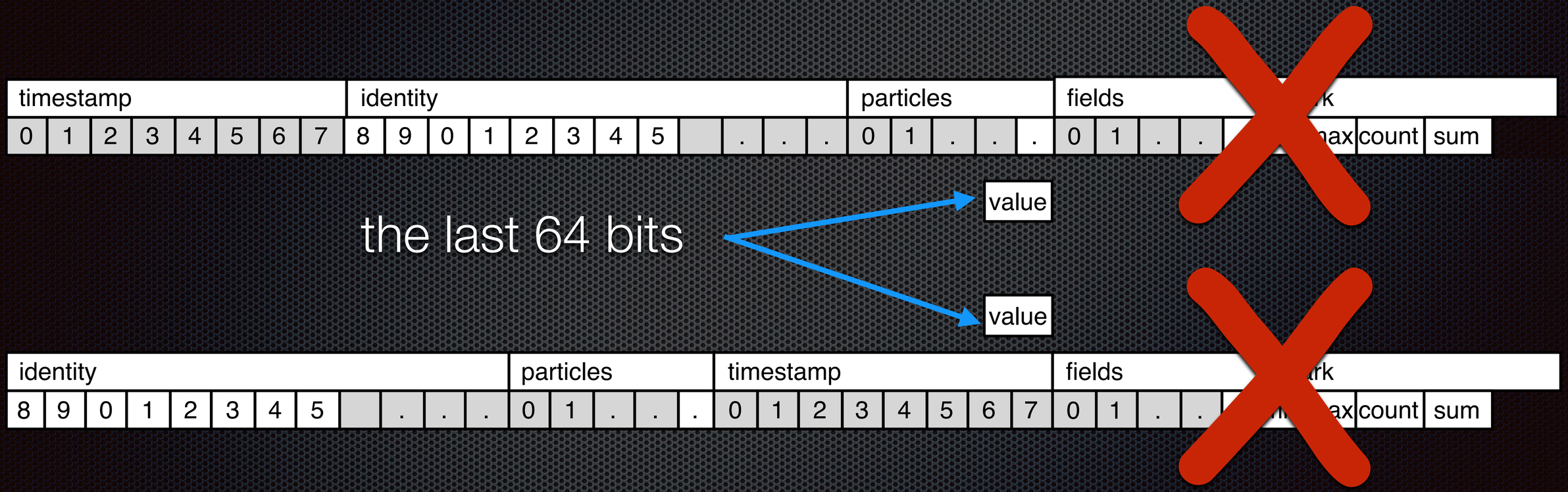
timestamp	identity	particles	fields	quark
0 1 2 3 4 5 6 7	8 9 0 1 2 3 4 5	0 1 . . .	0 1 . . .	min max count sum

the last 64 bits



identity	particles	timestamp	fields	quark
8 9 0 1 2 3 4 5	0 1 . . .	0 1 2 3 4 5 6 7	0 1 . . .	min max count sum

One more thing…

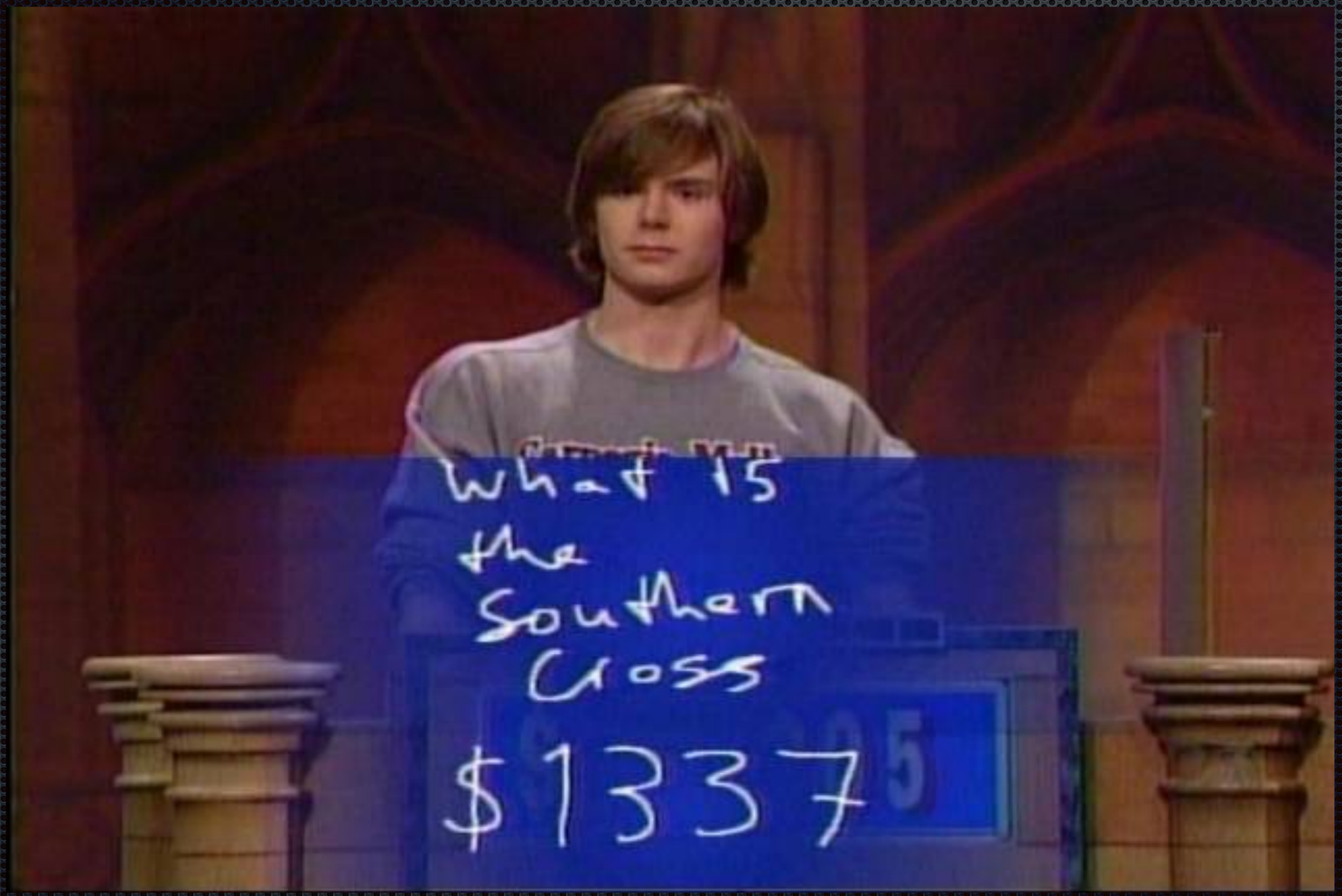


One more thing…

Time Series - Single measurement over time

Time Key ~~Value~~ - Related measurements over time

Questions?



Bibliography

TODO

~ : string = byte
q : quark = {~,…}
p : particles = {(~,...), (~,…)}
ID: collection of strings identifying attributes
a : atom = (ID * p + {~,…} + q) = datapoint
T : time
S : space

<-> = fusion/fission = Sum({Nand, …}) (-, +, /, *)