Real-time Analytics on Medical Device Data

Brock Noland

brock@phdata.io
@phdatainc
phdata.io

phData

About phData

Managed Services Provider fully dedicated to Hadoop, Spark, and Kafka

analytics

Tackle your organization's most challenging analytical problems.

applications

Operationalize Hadoop and bring your Hadoop-driven application into production.

infrastructure

Deploy, develop and adopt Hadoop and its supporting technologies.

About Brock Noland

- Co-founder and Chief Architect for phData
- Co-founder of Apache Sentry, Vice President of Apache MRUnit, PMC on Apache Hive, Flume, Crunch, Parquet, Incubator, and Apache Member
- Apache mentor to Apache Kudu and Impala (incubating)
- Early Clouderan and earlier Hadoop user (Thomson Reuters)
- Trainer, Solution Architect, Engineer, and Engineering Manager
- Deep Spark experience
  - Hive on Spark
  - StreamSets

- **Current trends**
- Medical devices
- Infrastructure
- Outcome

phData

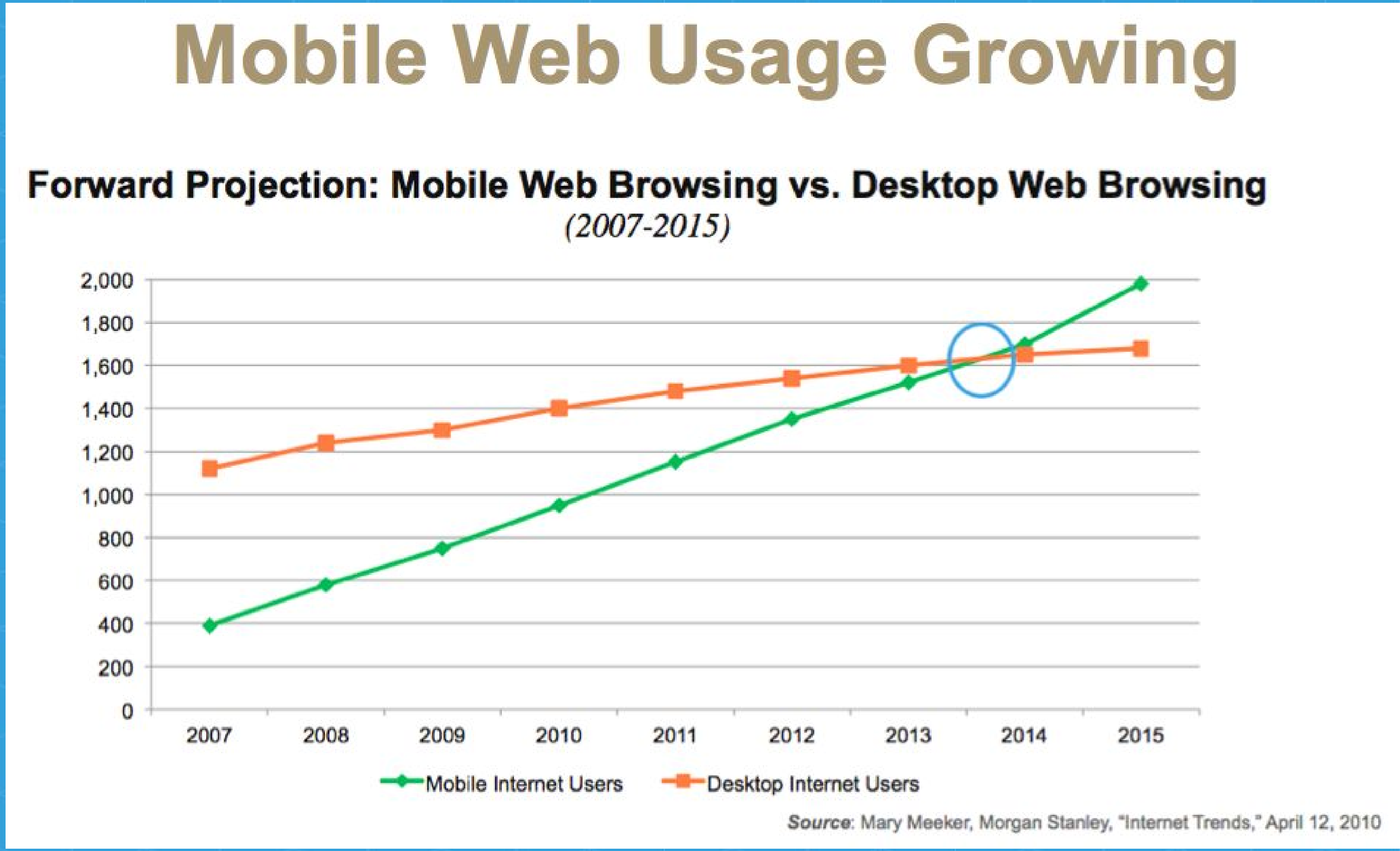
Current Trends

- Mobile technology miniaturization
- Always-on internet connections
- Big data technologies

phData

Mobile technology miniaturization

Always-on internet connections



Big data technologies

- Easily ingest, analyze and manage massive data volumes
- Previous generation technologies (RDBMS) capped out in low Terabytes
- Apache Hadoop/Spark installs start in low Terabytes

phData

Convergence

"More specifically, Rubin believes Internet-connected devices (smartphones, tablets, thermostats, smoke detectors, and cars, for example) will create massive amounts of data that will be analyzed by deep-learning technologies.

Andy Rubin: AI Is The Future Of Computing, Mobility - http://goo.gl/iJCnu8

- Current trends
- **Medical devices**
- Infrastructure
- Outcome

phData

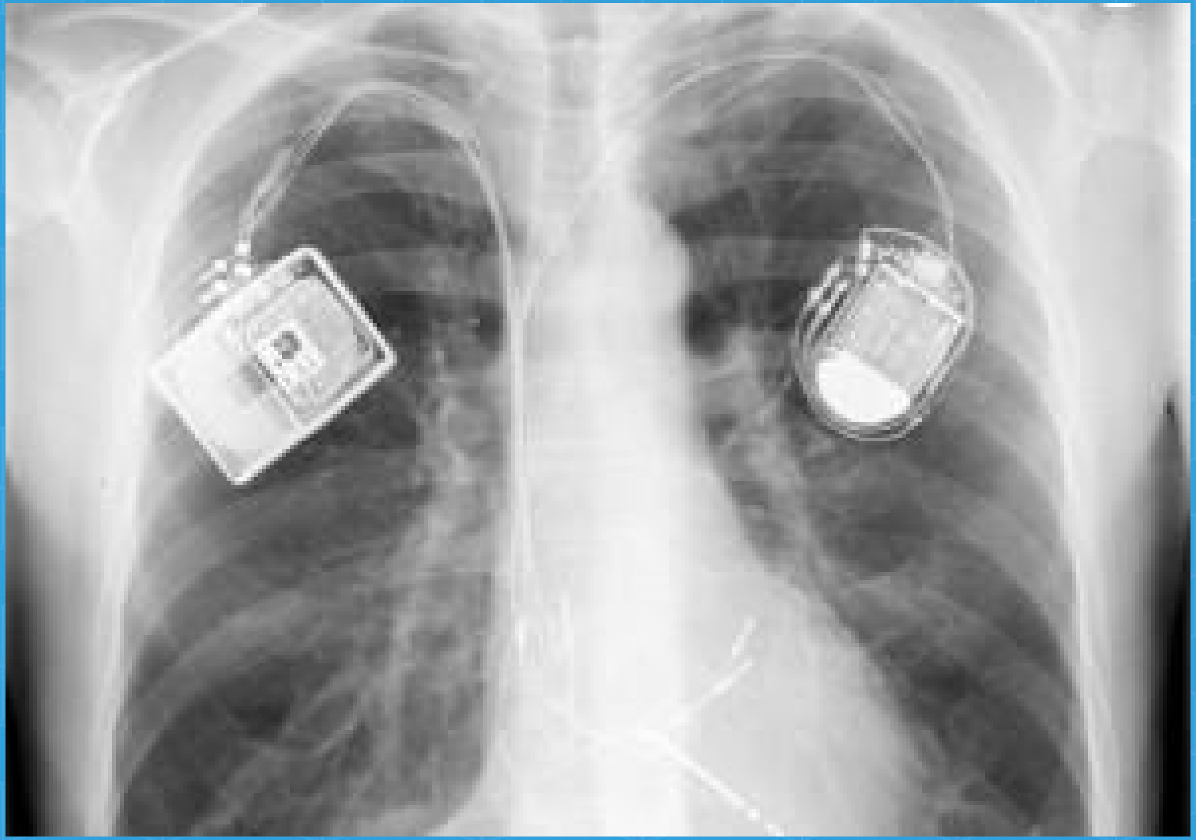
What does this mean for medical devices?

Medical devices

- Internal (Implantable)
- External

phData

Internal medical devices



phData

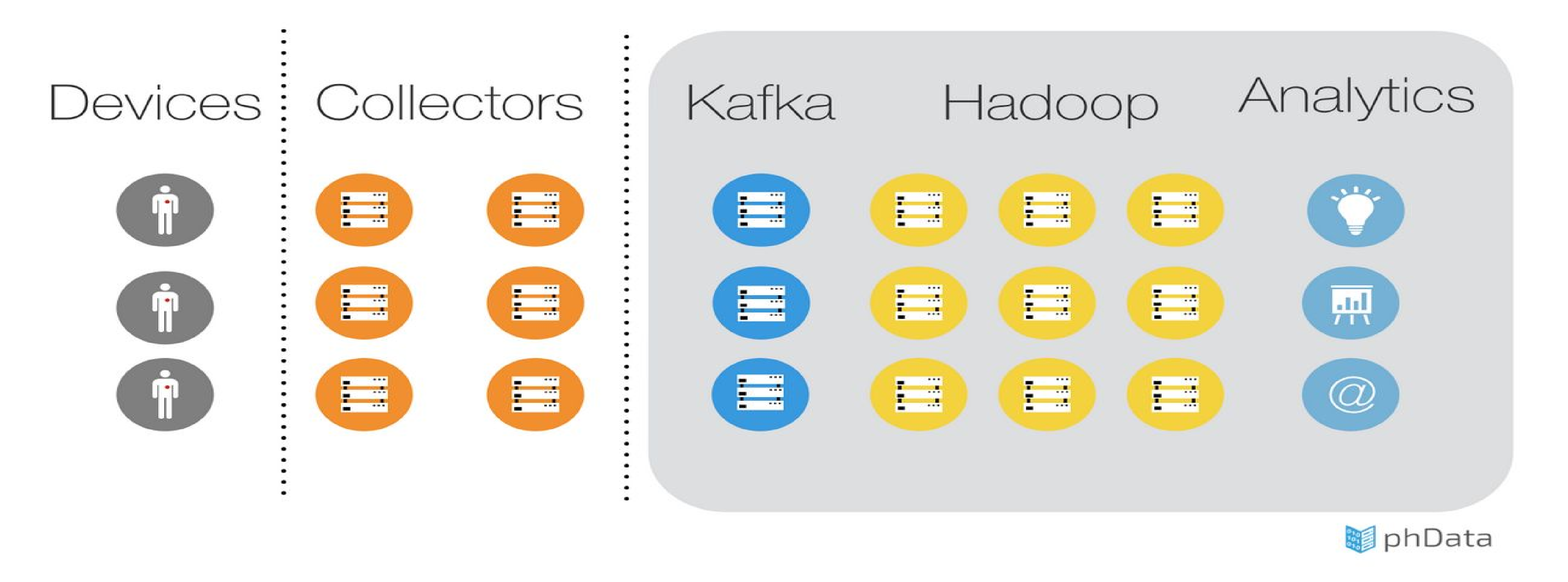
External medical devices

- e.g. Cardiac Event Recorders (CER)
- Rechargeable batteries
- Trivial to iterate
- Can be as simple as a watch

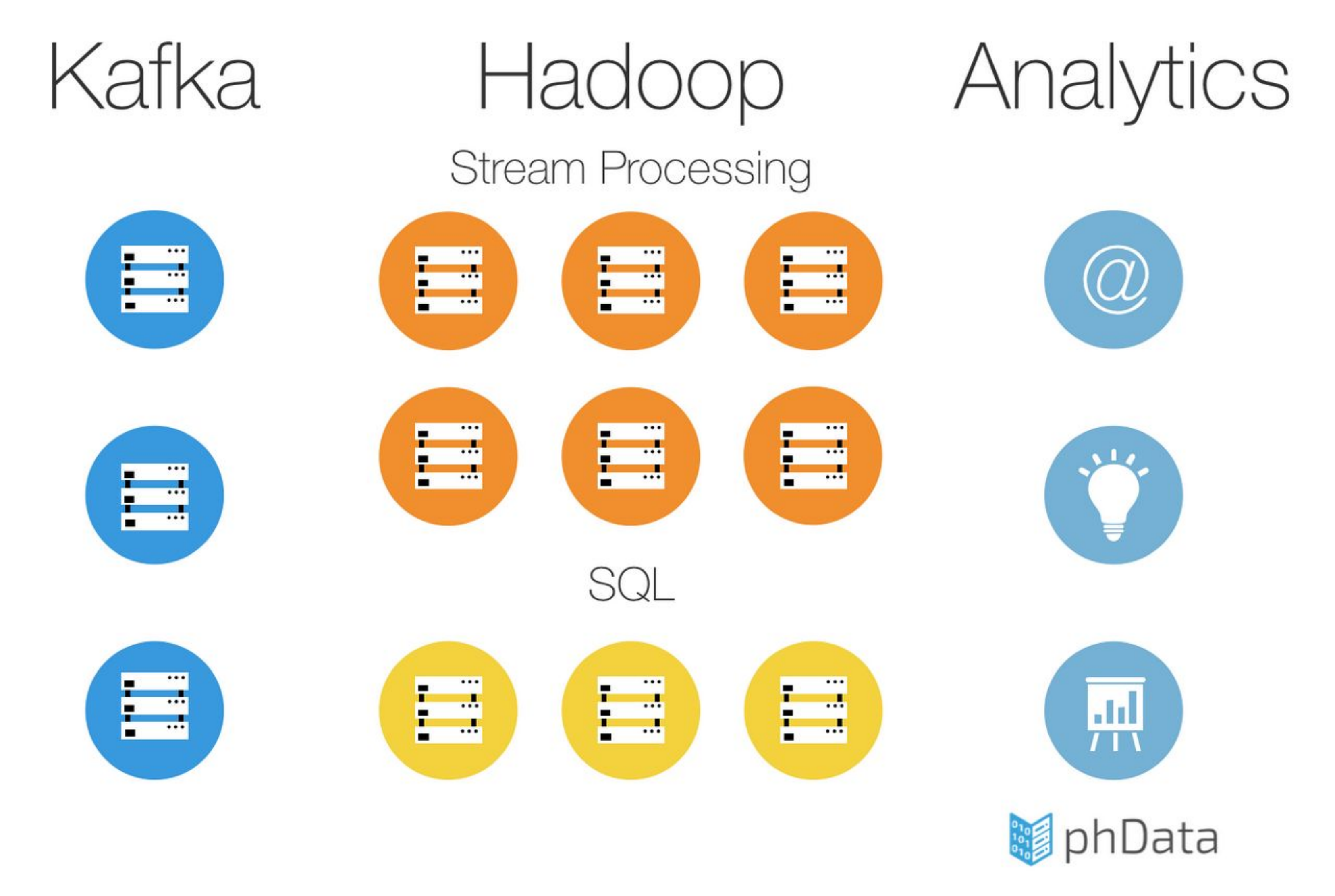
phData

- Current trends
- Medical devices
- **Infrastructure**
- Outcome

phData



phData


Kafka
Hadoop
Analytics
Stream Processing
SQL
phData

Apache Kafka

- Open source, scalable message system
- Important for real-time IoT use cases
- 1.1 trillion messages per day at LinkedIn
  http://goo.gl/Wqb72H

phData

Apache Hadoop

- Open source, scalable **ecosystem**
- Important for storing large historical data sets
- 455 petabytes at Yahoo
  http://goo.gl/utc4qM

phData

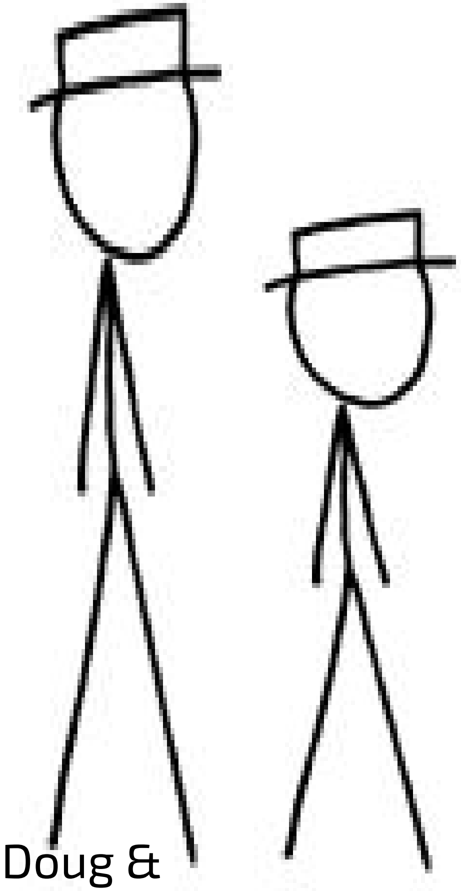
What is the Hadoop ecosystem?

- Apache
  - Atlas
  - Ambari
  - Accumulo
  - Avro
  - Beam
  - Bigtop
  - Chukwa
  - Crunch
  - Eagle
  - Falcon
  - Flink
  - Flume
  - Giraph
  - Hadoop
    - HDFS
    - MapReduce
    - YARN
  - HBase
  - Hama
  - Hive
  - Impala
  - Incubator

- Apache
  - Kylin
  - Kudu
  - Kafka
  - Knox
  - Mesos
  - Myriad
  - Metron
  - Mahout
  - Nifi
  - Oozie
  - ORC
  - Oozie
  - Parquet
  - Pig
  - REEF
  - Ranger
  - Sentry
  - Storm
  - Samza
  - Slider
  - Solr
  - Spark

- Apache
  - Spot
  - Sqoop
  - SystemML
  - Twill
  - Thrift
  - Whir
  - Zeppelin
  - Zookeeper
- AtScale
- Alluxio/Tachyon
- Cask
- DataFu
- Dremel
- Kylin
- Hue
- BigTable
- Deeplearning4j
- H20
- Oryx
- StreamSets
- Sense

HDFS



Doug &
Mike

- Started building a web search engine in 2002
  - Crawl entire internet
  - Generate large files
  - Process and index huge volumes of data
- Discovered Google's GFS paper in 2002, MapReduce in 2004

HDFS Design Assumptions

- Files are append-only
- Files are large (GBs to TBs)
- Accesses are large and sequential

HDFS Today

- HDFS is great at:
  - Large scans
  - Unstructured data
  - Batch ingest
  - Ingest/access of large files

- HDFS is NOT great at:
  - Random read/write
  - Real-time ingest
  - Updates

HBase

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.

HBase Design

- Need OLTP-like storage
  - Low latency
  - Keys are indexed
  - Provide fast random read/write access
  - Mutable

- Same properties apply to Cassandra

Apache Spark

- Open source, scalable processing engine
- Important for scalable stream processing (among other things)
- Processing all daily rides at Uber
  http://goo.gl/dclAub

phData

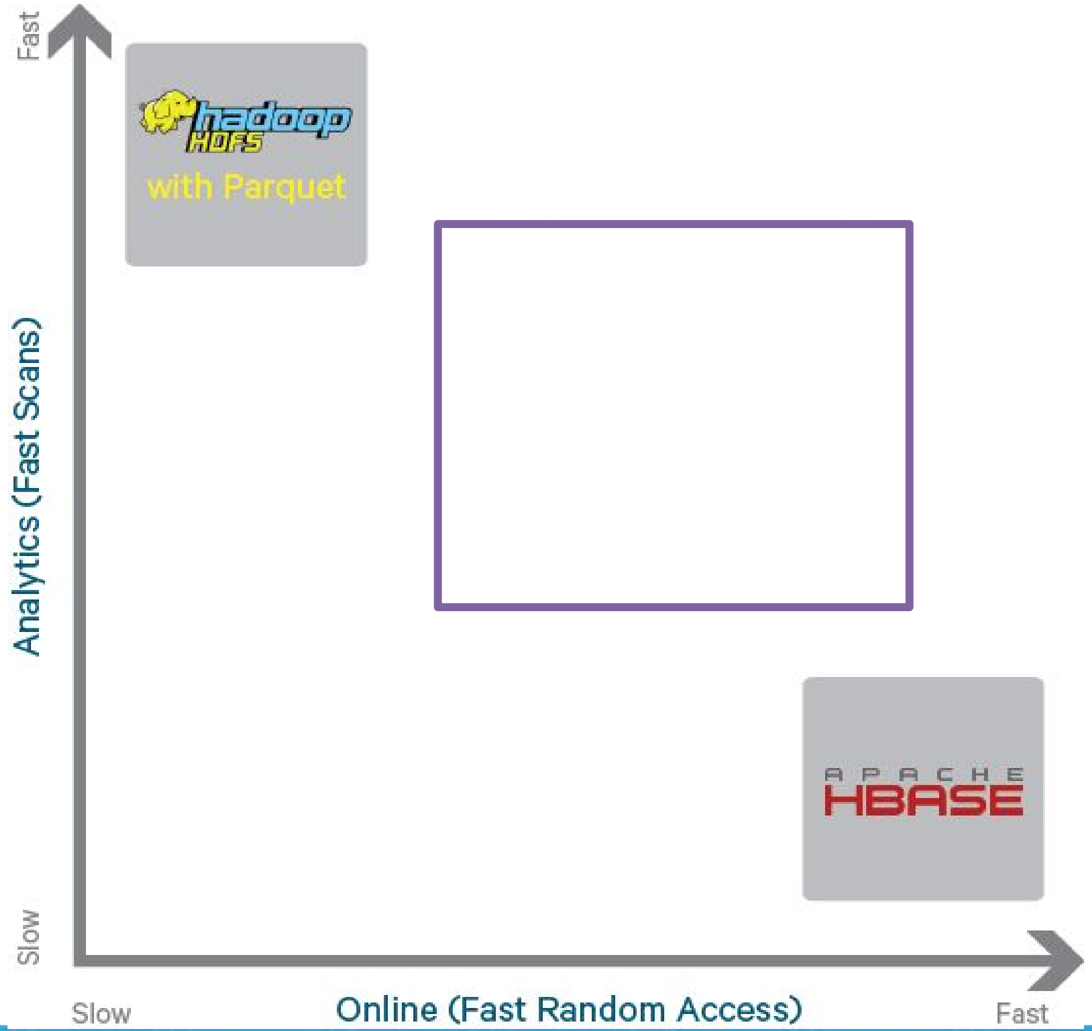
SQL

- Many projects providing SQL
  - Apache Hive
  - Apache SparkSQL
  - Cloudera Implala

- Important for post hoc analytics

phData

Previous storage landscape of the Hadoop ecosystem



HDFS (GFS) excels at:

- Batch ingest only (eg hourly)
- Efficiently scanning large amounts of data (analytics)

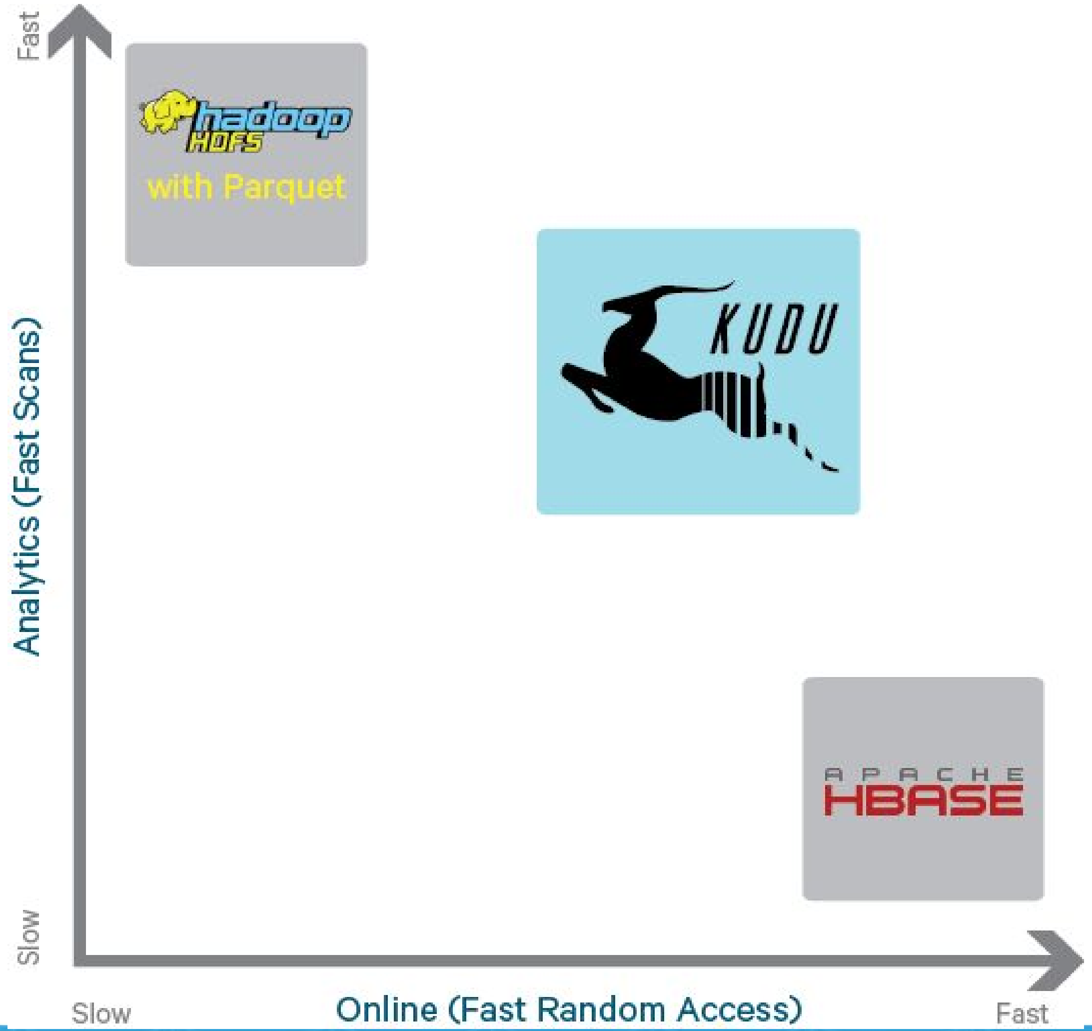
HBase (BigTable) excels at:

- Efficiently finding and writing individual rows
- Making data mutable

Gaps exist when these properties are needed *simultaneously*

- HDFS allows for fast writes and scans, but updates are slow and cumbersome
- HBase is fast for updates and inserts at the expense of data scans (i.e. analytics)

Kudu design goals



- **High throughput** for big scans
  Goal: Within 2x of Parquet
- **Low-latency** for short accesses
  Goal: 1ms read/write on SSD
- **Database-like** semantics
  (initially single-row ACID)
- **Relational data model**
  - SQL queries are easy
  - "NoSQL" style scan/insert/update (Java/C++ client)

What Kudu is ***NOT***

Not a SQL interface itself

- ✓ It's a storage layer

Not an application that runs on HDFS

- ✓ It's an alternative, native Hadoop storage engine

Not a replacement for HDFS or HBase

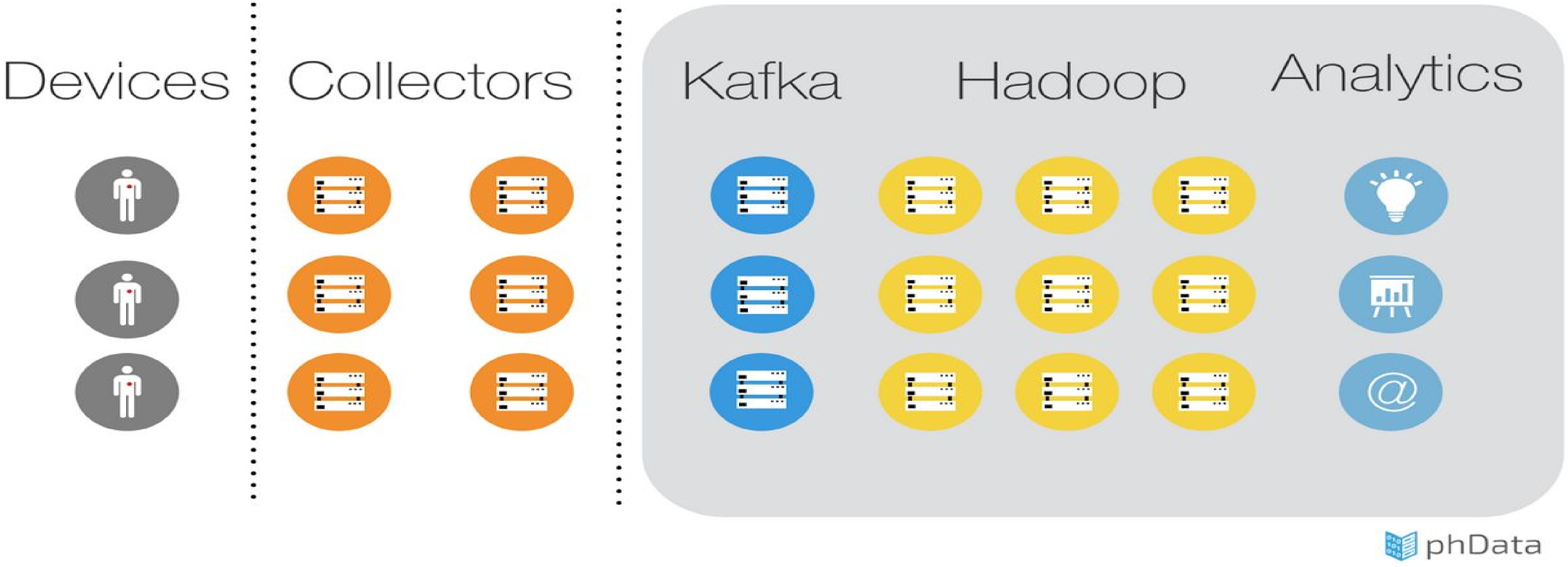
- ✓ Select the right storage for the right use case
- ✓ Cloudera will continue to support and invest in all three

Common Kudu use cases

Kudu is best for use cases requiring a simultaneous combination of sequential and random reads and writes

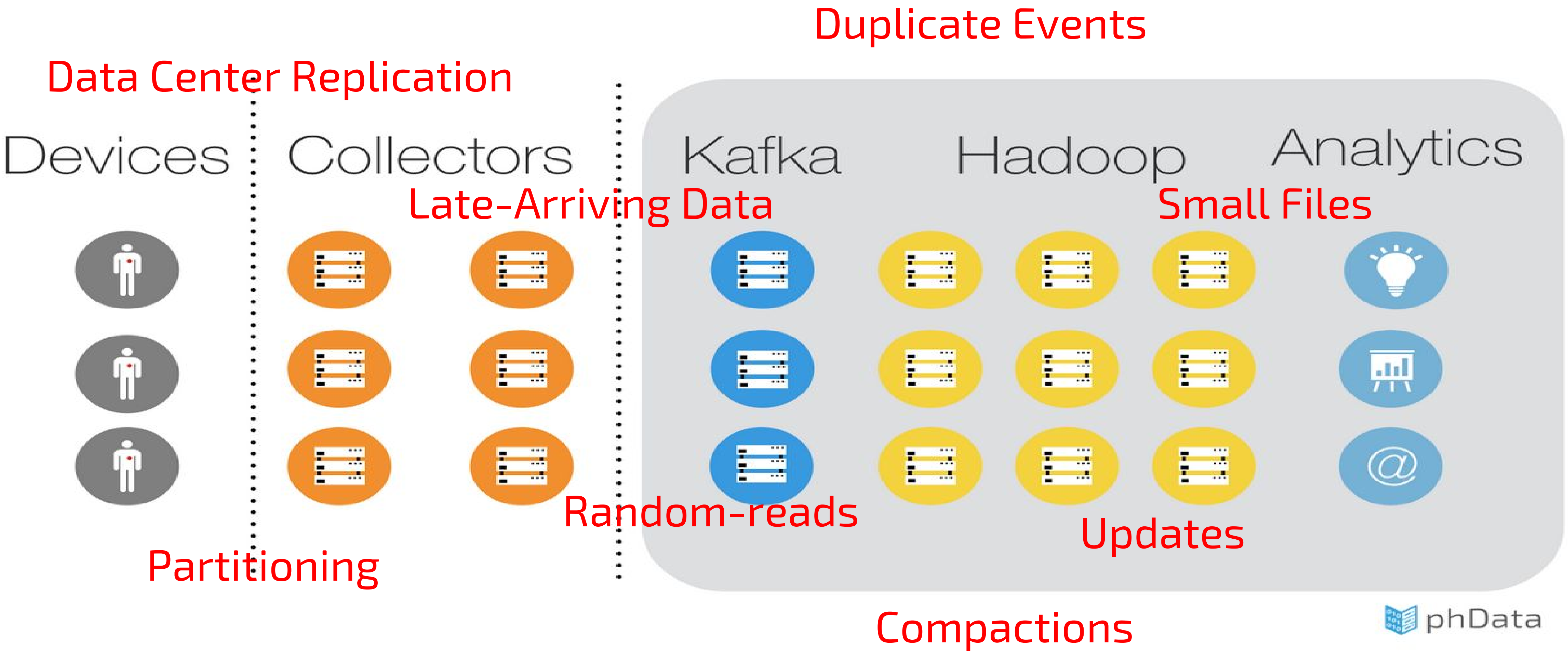
- **Time series**
  - Examples: Streaming market data, fraud detection / prevention, risk monitoring
  - Workload: Insert, updates, scans, lookups
- **Machine data analytics**
  - Example: Network threat detection
  - Workload: Inserts, scans, lookups
- **Online reporting**
  - Example: Operational data store (ODS)
  - Workload: Inserts, updates, scans, lookups
- **Potentially anything where the data is not append only** (or needs scale, parallelism, …)

How would we build any Streaming or IOT Analytics System Today?



phData

What makes this hard?



phData

Blog

Real-time Analytics on Medical Device Data – Part 1 – Introduction

BY: BROCK NOLAND | JULY 21, 2015 | HADOOP, INGEST, KAFKA, MEDICAL, SQL

This is the first post in a series of posts on real-time analytics on medical device data. Read post two on the required infrastructure and post three on the SQL schema.

phData believes the Internet of Things (IoT) and Big Data technology such as Apache Hadoop will result in big changes in health care. Dramatic advances in **Telemedicine** will be made possible by the constant miniaturization of electronics, ever present internet connections and the ability to store and analyze large volumes of data.

phData

Common Hive schema

Raw Data

date	t1	patient_id	a	...	z
20150101	00:00:01:43	12345	1.4	...	0.04
20150101	00:00:01:12	98273	1.7	...	0.12
20150101	00:00:02:43	12345	1.4	...	0.06
20150102	00:00:03:43	12345	1.3	...	0.07
20150102	00:00:01:37	58796	2.1	...	0.07
...	...	...	...	...	...
20151231	11:59:59:43	12345	1.6	...	0.04
20151231	11:59:59:12	98273	1.1	...	0.11

365*24*60*10 million records per year

Patient ID: 12345

date	t1	a	...	z
20150101	00:00:01:43	1.4	...	0.04
20150101	00:00:02:43	1.4	...	0.06
20150102	00:00:03:43	1.3	...	0.07
...	...	...	...	...
20151231	11:59:59:43	1.6	...	0.04

525,600 records per patient per year

Partition by Date

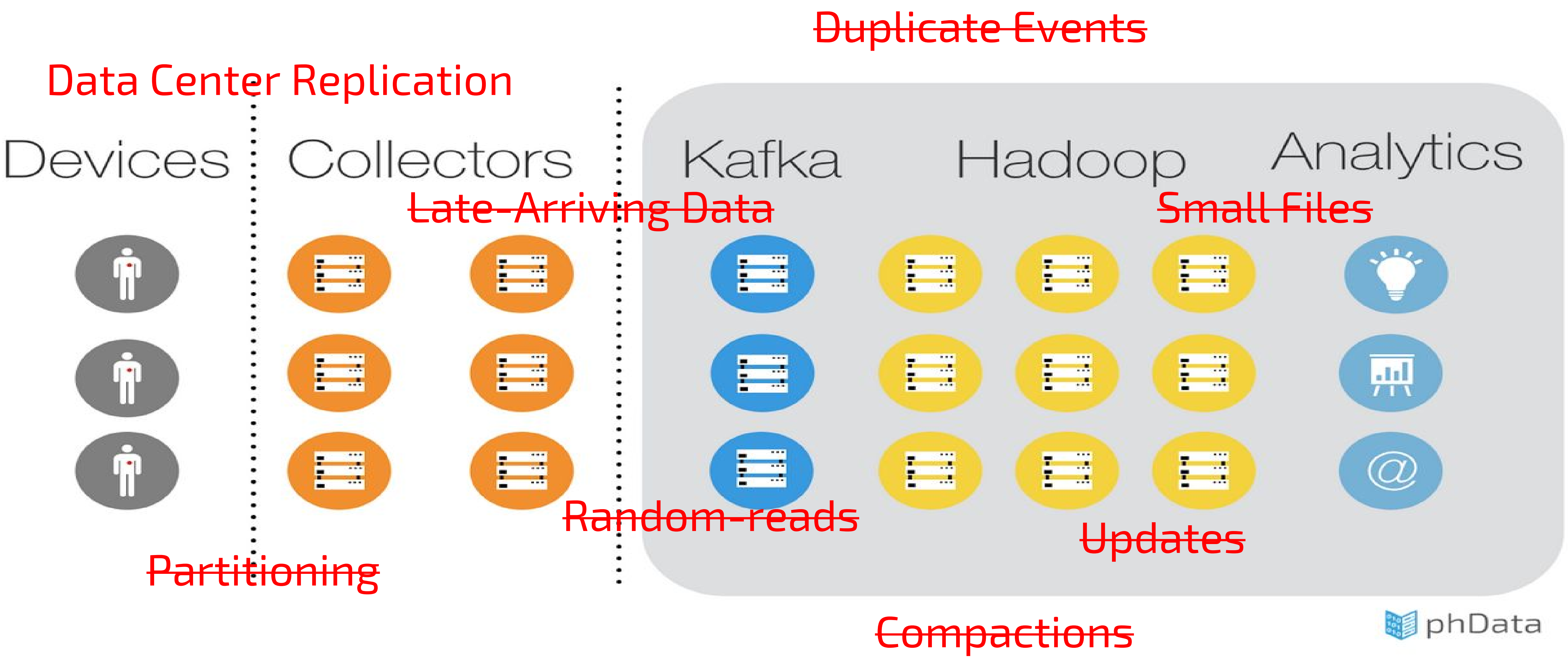
date	t1	patient_id	a	...	z
20150101	00:00:01:43	12345	1.4	...	0.04
20150101	00:00:02:43	12345	1.4	...	0.06
...	...	...	...	...	...
20150101	11:59:59:43	12345	1.3	...	0.04

20150102	00:00:01:43	12345	1.4	...	0.05
20150102	00:00:02:43	12345	1.5	...	0.05
...	...	...	...	...	...
20150102	11:59:59:43	12345	1.6	...	0.06

⋮

20151231	00:00:01:43	12345	1.3	...	0.06
20151231	00:00:02:43	12345	1.3	...	0.05
...	...	...	...	...	...
20151231	11:59:59:43	12345	1.4	...	0.05

Which problems go away with Kudu?



Changing hardware landscape

- **Spinning disk** -> **solid state storage**
  - **NAND flash**: Up to 450k read 250k write iops, about 2GB/sec read and 1.5GB/sec write throughput, at a price of less than $3/GB and dropping
  - **3D XPoint memory** (1000x faster than NAND, cheaper than RAM)

- **RAM** is cheaper and more abundant:
  - 64->128->256GB over last few years

- ***Takeaway***: The **next bottleneck is CPU**, and current storage systems weren't designed with CPU efficiency in mind.

Kudu CPU efficiency

- C++ implementation designed for modern CPUs
  - Use SSE instructions, optimize for deep CPU cache hierarchy
  - Embed Impala's LLVM-based query fragment execution for SQL access
  - Avoid GC and other Java pitfalls
- File formats designed for efficient deserialization
  - Type-specific compression much faster than generic (LZO)
  - Trade-offs: 10% space increase is worth it, if 100% faster to decode

Kudu Basic Design

- Basic Construct: Tables
  - Tables broken down into a storage mechanism named Tablets (roughly equivalent to regions or partitions)
  - Columnar storage
  - Flexible data partitioning
- Typed storage
  - Enables compression
- Next generation consistency and replication
  - Multi-Version Concurrency Control (MVCC)
  - Raft Consensus to replicate *operations*
  - Architecture supports geographically disparate, active/active systems
    - o <u>*Not in the initial implementation*</u>

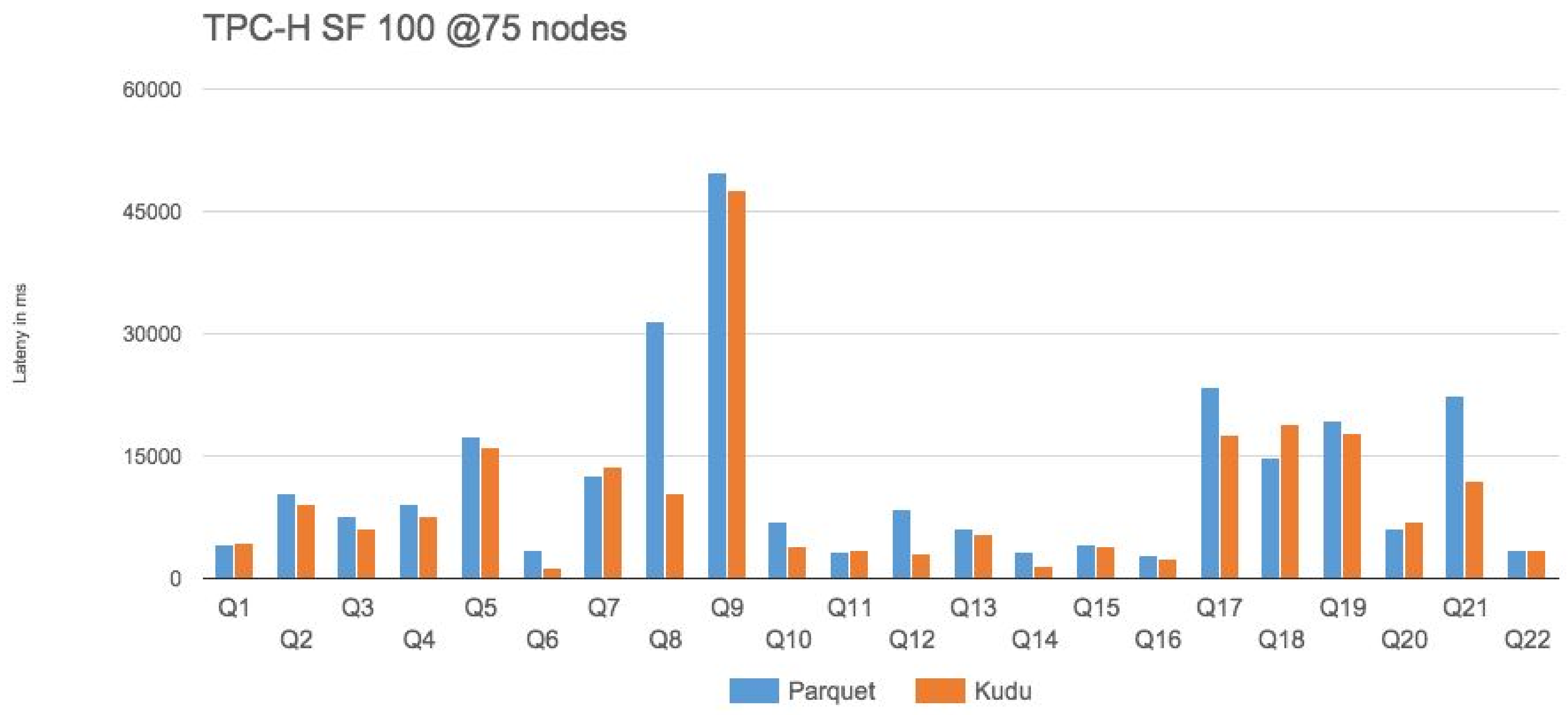
Kudu: Scalable and fast tabular storage

- **Scalable**
  - Tested up to 275 nodes (~3PB cluster)
  - Designed to scale to **1000s of nodes, tens of PBs**
- **Fast**
  - **Millions** of read/write operations per second across cluster
  - **Multiple GB/second** read throughput per node
- **Tabular**
  - Store tables like a normal database (can support SQL, Spark, etc)
  - Individual record-level access to **100+ billion row tables** (Java/C++/Python APIs)

TPC-H (analytics benchmark)

- 75 server cluster
  - 12 (spinning) disks each, enough RAM to fit dataset
  - TPC-H Scale Factor 100 (100GB)
- Example query:

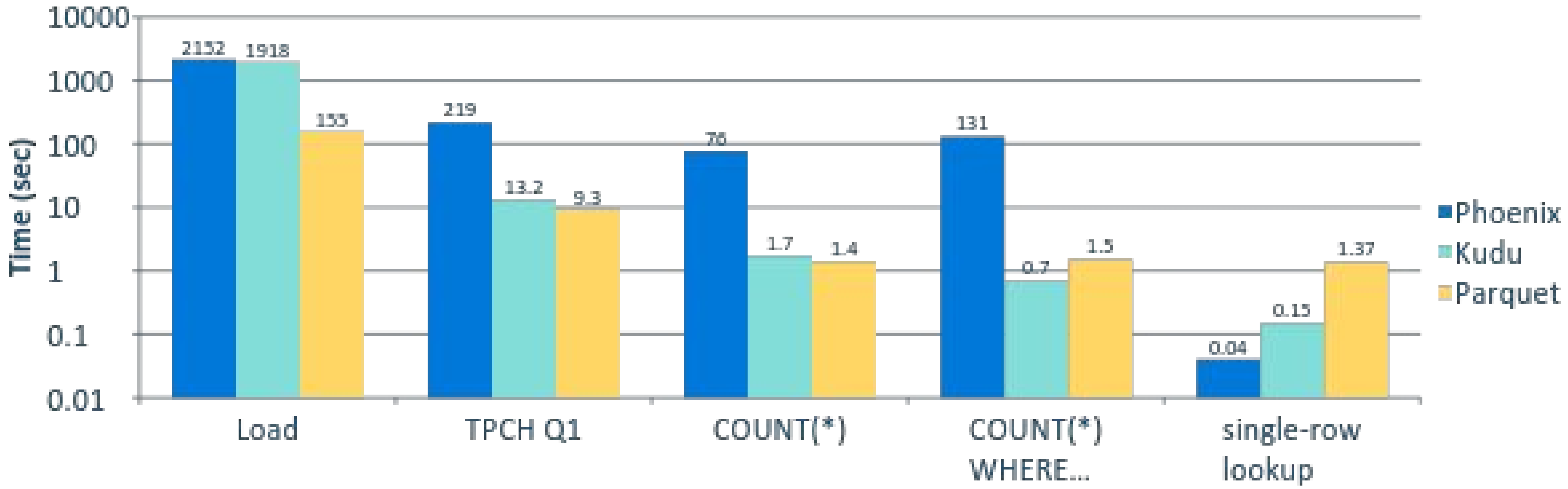
```
SELECT n_name, sum(l_extendedprice * (1 - l_discount)) as revenue
FROM customer, orders, lineitem, supplier, nation, region
WHERE c_custkey = o_custkey AND l_orderkey = o_orderkey
   AND l_suppkey = s_suppkey AND c_nationkey = s_nationkey
   AND s_nationkey = n_nationkey
   AND n_regionkey = r_regionkey
   AND r_name = 'ASIA'
   AND o_orderdate >= date '1994-01-01'
   AND o_orderdate < '1995-01-01’
GROUP BY n_name ORDER BY revenue desc;
```



- Kudu outperforms Parquet by 31% (geometric mean) for RAM-resident data

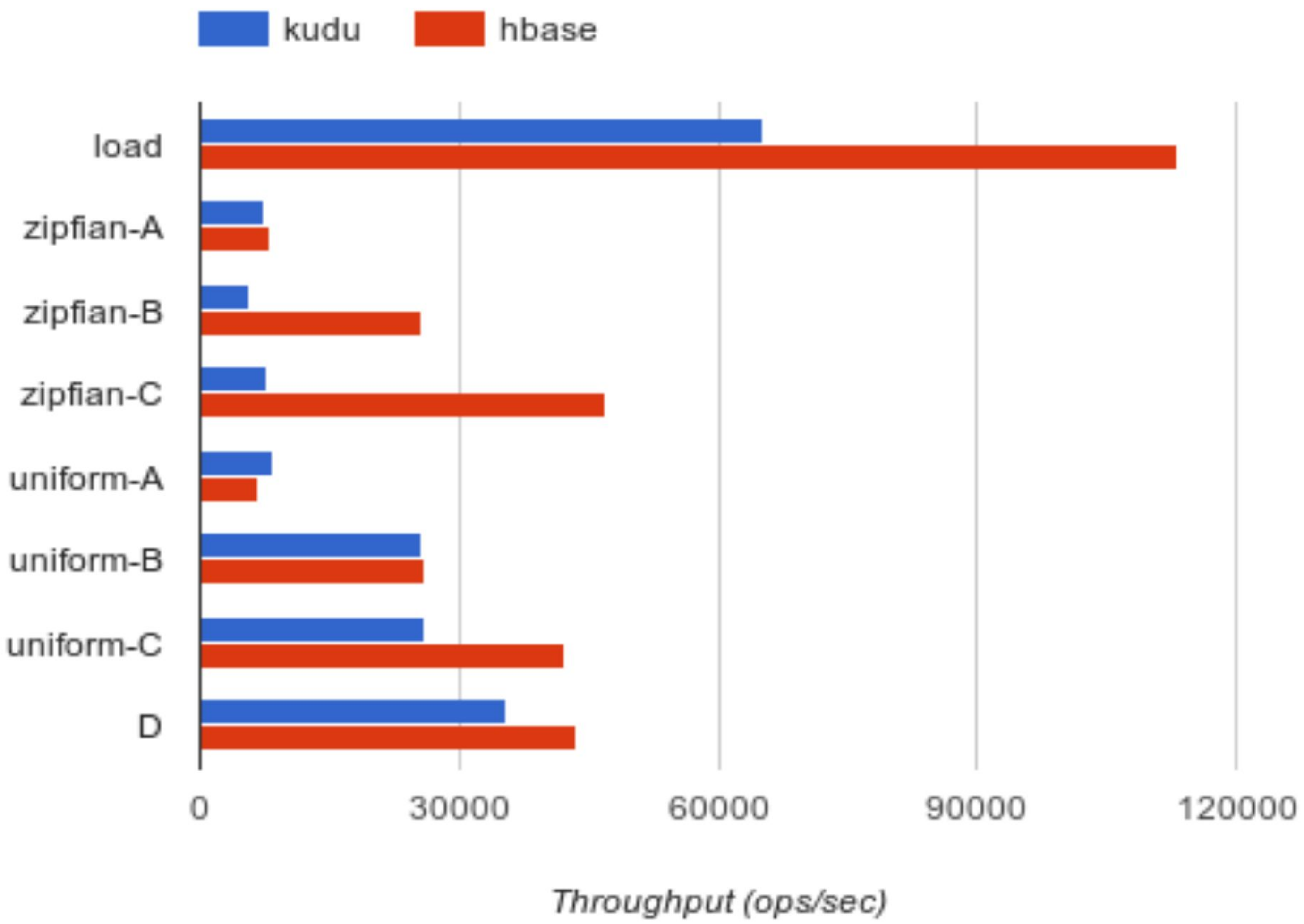
Versus other NoSQL storage

- **Apache Phoenix: OLTP SQL engine built on HBase**
- 10 node cluster (9 worker, 1 master)
- TPC-H LINEITEM table only (6B rows)



What about NoSQL-style random access? (YCSB)

- **YCSB** 0.5.0-snapshot
- 10 node cluster (9 worker, 1 master)
- 100M row data set
- 10M operations each workload



- Current trends
- Medical devices
- Infrastructure
- **Outcome**

phData

Parkinson Disease

- 60,000 Americans are diagnosed with Parkinson's every year
- No cure, several therapies with varying tradeoffs
- No objective measurement on therapy impact

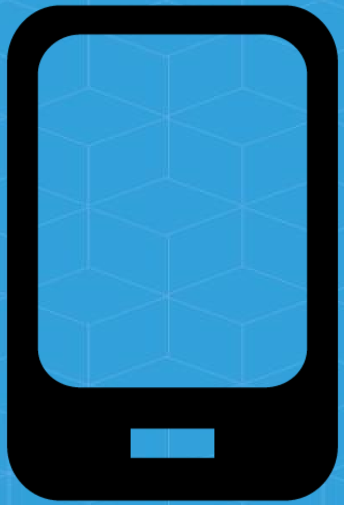
phData

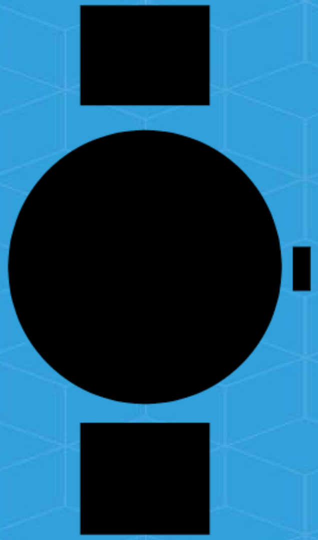
Therapies

- Drugs are minimally invasive, but can make symptoms worse and cause seizures
- Surgery to remove working portions of the brain
- Implantable medical devices are invasive and carry their own set of risks

phData

Solution





Data driven disease management

- Device continually uploads tremor data
- Useful for multiple
  - Clinician
  - Researcher
  - Pharmaceutical & Medical Device
- Intel and Michael J. Fox Foundation
  - http://goo.gl/Eccu5x
  - http://goo.gl/8nVPuU

phData

Summary

- Mobile technology
- Medical devices
- Combined with open source big data technologies
  - Hadoop
  - Spark
  - Kafka
  - Kudu
- Results in healthier population and better care

phData

Brock Noland

brock@phdata.io
@phdatainc
phdata.io

phData