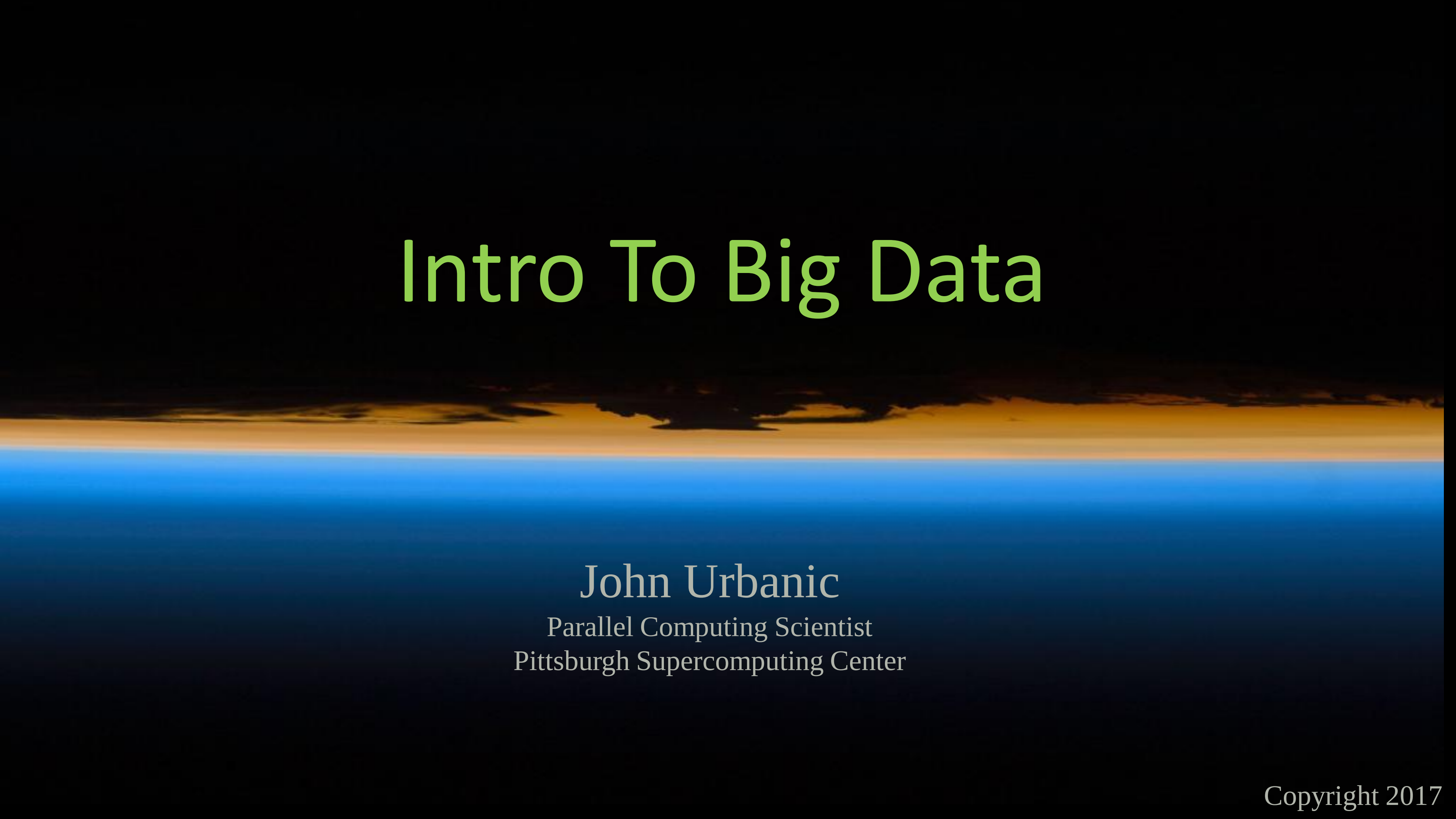


Intro To Big Data

John Urbanic
Parallel Computing Scientist
Pittsburgh Supercomputing Center

Big data is a broad term for data sets so large or complex that traditional data processing applications are inadequate.

—*Wikipedia*

Implications of “Big”

A classic amount of “small” data

Find a tasty appetizer – Easy!

Find something to use up these oranges – grumble…

What if….

Just need a little more sophistication…

Find all client names starting with "A"

Get all cases from 2007 - grumble

Get all cases from 2007…from Clinton County. – grumble, grumble.

Faster…

Even sophistication has its limits.

Find books on Modern Physics (DD# 539)

Find books by Wheeler

where he isn't the first author – grumble…

Your only hope…

Less sophisticated, but sometimes better…

“Chronologically” or “geologically” organized.
Familiar to some of you at tax time.

Get all articles from 2007.

Get all papers on “fault tolerance”
– grumble and cough

Indexing will determine your individual performance. Teamwork can scale that up.

How much is Big Data?

1 LOC

Library of Congress

= 10 TB *

**Actually, a silly estimate. The original reference actually mentions a more accurate 208TB, and in 2013 the digital collection alone was 3PB.*

A better sense of biggish

Size

- 1000 Genomes Project
    - AWS hosted
    - 260TB
- Common Crawl
    - Soon to be hosted on Bridges
    - 300-800TB+

Throughput

- Square Kilometer Array
    - Building now
    - Exabyte of raw data/day – compressed to 10PB
- Internet of Things (IoT) / motes
    - Endless streaming

Records

- GDELT (Global Database of Events, Language, and Tone)
    - 250M rows and 59 fields (BigTable)
    - *"during periods with relatively little content, maximal translation accuracy can be achieved, with accuracy linearly degraded as needed to cope with increases in volume in order to ensure that translation always finishes within the 15 minute window…. and prioritizes the highest quality material, accepting that lower-quality material may have a lower-quality translation to stay within the available time window."*

If it is all about the queries vs. the data, what options do we have?

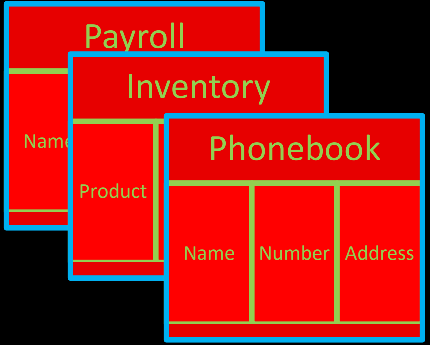
Let’s map out a few provinces:

- SQL
- No SQL
  - Key/Value
  - Document
  - Column
  - Graph
- Analysis / Machine Learning

Good Ol' SQL

MySQL, Postgres, Oracle, etc.

```
SELECT  NAME, NUMBER, FROM PHONEBOOK
```



Why it *wasn't* fashionable:

- Schemas set in stone:
  - Need to define before we can add data
  - Not a fit for "agile development"
- Queries often require accessing multiple indexes and joining and sorting multiple tables
- Sharding isn't trivial
- Caching is tough
  - ACID (Atomicity,Consistency,Isolation,Durability) in a *Transaction* is costly.

NoSQL

- Everything Else?
- Not Only SQL
- Was effectively *NoACID*
- Now maybe *NoRelational*

Key-Value

Redis, Memcached, Amazon DynamoDB, Riak, Ehcache

```
GET foo
```

- Certainly agile (no schema)
- Certainly scalable (linear in most ways: hardware, storage, cost)
- Good hash might deliver fast lookup
- Sharding, backup, etc. could be simple
- Often used for "session" information: online games, shopping carts

```
GET cart:joe:15~4~7~0723
```

foo	bar
2	fast
6	0
9	0
0	9
text	pic
1055	stuff
bar	foo

Document

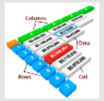
Cassandra, CouchDB, MongoDB

```
GET foo
```

- Value must be an object the DB can understand
- Common are: XML, JSON, Binary JSON and nested thereof
- This allows server side operations on the data

```
GET plant=daisy
```

- Can be quite complex: Linq query, JavaScript function
- Different DB's have different update/staleness paradigms

foo	
2	`<CATALOG> <PLANT> <COMMON>Bloodroot</COMMON> <BOTANICAL>Sanguinaria canadensis</BOTANICAL> <ZONE>4</ZONE> <LIGHT>Mostly Shady</LIGHT> <PRICE>$2.44</PRICE> <AVAILABILITY>031599</AVAILABILITY> </PLANT> <PLANT> <COMMON>Columbine</COMMON> <BOTANICAL>Aquilegia canadensis</BOTANICAL> <ZONE>3</ZONE> <LIGHT>Mostly Shady</LIGHT> <PRICE>$9.37</PRICE> <AVAILABILITY>030699</AVAILABILITY> </PLANT>` . .
6	JSON
9	XML
0	Binary JSON
bar	JSON XML
12	XML XML

Wide Column Stores

Google BigTable, Cassandra, HBase

```
SELECT Name, Occupation FROM People WHERE key IN (199, 200, 207);
```

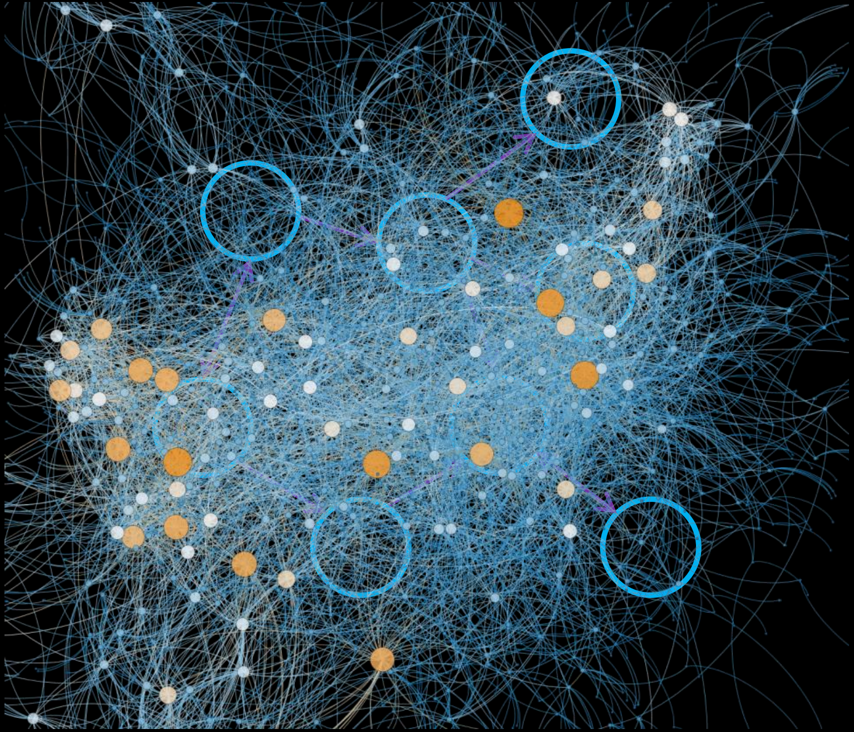
- No predefined schema
- Can think of this as a 2-D key-value store: the value may be a key-value store itself
- Different databases aggregate data differently on disk with different optimizations

Key			
Joe	**Email:** joe@gmail	**Web:** www.joe.com	
Fred	**Phone:** 412-555-3412	**Email:** fred@yahoo.com	**Address:** 200 S. Main Street
Julia	**Email:** julia@apple.com		
Mac	**Phone:** 214-555-5847		

Graph

Neo4J, Titan, GEMS

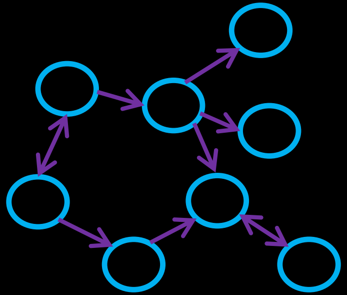
- Great for semantic web
- Great for graphs
- Can be hard to visualize
- Serialization can be difficult
- Queries more complicated



From *PDX Graph Meetup*

Queries

SPARQL, Cypher



SPARQL (W3C Standard)

- Uses Resource Description Framework format (triple store)
- RDF Limitations
  - No named graphs
  - No quantifiers or general statements
    - “Every page was created by some author”
    - “Cats meow”
- Requires a schema (RDFS) to define rules
    - "The object of ‘homepage’ must be a Document.“

```
SELECT ?name ?email
WHERE {
          ?person a foaf:Person.
          ?person foaf:name ?name.
          ?person foaf:mbox ?email.
}
```

Cypher (Neo4J only)

- No longer proprietary
- Stores whole graph, not just triples
- Allows for named graphs
- …and general Property Graphs (edges and nodes may have values)

```
SMATCH (Jack:Person
  { name:‘Jack Nicolson’})-[:ACTED_IN]-(movie:Movie)
RETURN movie
```

Hadoop & Spark

~~What kind of databases are they?~~

Frameworks for data

These are both frameworks for distributing and retrieving data. Hadoop is focused on disk based data and a basic map-reduce scheme, and Spark attempts evolves that in several directions that we will get in to. Both can accommodate multiple types of databases and *achieve their performance gains by using parallel workers*. You are about to learn a lot more, but here are a few concrete examples:

Hadoop

- HBASE: modeled after BigTable and a natural fit
- HIVE: SQL-like HiveQL converts to the underlying map/reduce (often slow)
- Now very passe

Spark

- Spark SQL (moved to star billing as DataFrames!)
- GraphX
- MLlib: stats, clustering, optimization, regression, etc.