Big data foundations
Big data in real-world

Big data in the movies

Big data in the sports

Big data in the hospitals

• Q: What would be volume and financial impact be if we were to hire another cardiovascular surgeon?
• Q: What are re-admission patterns for heart failure patients?
• Q: For a specific diagnosis, what are core interventions that improves the outcomes?
Big data in real-world

Government
- “Pillbox” project in US -> reduce expenses of 50 million USD per year
- Customized employment using Big data in Germany -> reduce 10 billion euro per 3 years
- Open competition by NIH -> detect geographical epidemic diseases via twitter analysis

Industry
- Google: predict geographical epidemic flu (trajectory) via search engine log analysis
- Google: provide real time road traffic service
- Volvo: find initial faulty of newly released vehicles via SNS and blog analysis (prevent recall of 50 thousand vehicles)
- Hertz: review customers’ evaluations by Big data analysis
- Posco: determine the purchase time and price of raw materials
- Watcha: recommend movies via taste analysis (no 1., larger than Naver movie)
- Xerox: recruit via SNS analysis

Prompt response to commercial condition changes, improve credibility and image, reduce expenses, improve productivity, facilitate administration, etc.
Big data as proper noun

*what is big data?*

Extremely large data (Wikipedia, Mckinsey)

- Too large to store, manage, and analyze in existing ways using existing storage and existing DBMS SWs

Government and Industry

- Information technology to predict trends and respond proactively.
- Technology to collect, store, manage, search, and analyze large scale data.
Evolution of science


• Before 1600: empirical science
• 1600~1950: theoretical science
• 1950~1990: computational science
• 1990~: data science <= DB, DW (data warehouse)

• 2010~ : Mobile Computing and Big Data
  • SNS, UCC, RFID, sensors, ...
  • Twitter 1TB/day, Facebook 15TB/day, ...
  • Size of data accumulated for the last 2 years > that of the previous 10 years
  • 80 exabyte at 2009 -> 40% increase every year -> 35 zetabyte at 2020

• (2010~ : Deep learning and AI)
History of Big data

Relational database management

Data warehousing

Data mining

Big data
Business intelligence

- Increasing potential to support business decisions
- Decision Making
  - Data Presentation
    - Visualization Techniques
  - Data Mining
    - Information Discovery
  - Data Exploration
    - Statistical Summary, Querying, and Reporting
  - Data Preprocessing/Integration, Data Warehouses
  - Data Sources
    - Paper, Files, Web documents, Scientific experiments, Database Systems

- End User
- Business Analyst
- Data Analyst
- DBA
All sciences are data sciences!


“The necessity of grappling with Big Data, and the desirability of unlocking the information hidden within it, is now a key theme in all the sciences – arguably the key scientific theme of our times.”

Francis X. Diebold
Paul F. and Warren S. Miller Professor of Economics
School of Arts and Sciences, University of Pennsylvania
Big data demands

KDnuggets report, 2017

- Data scientist is selected as the sexiest job on 21st century by Harvard business review
- From 2014 to 2024, the data scientist career path is expected to grow by 11%–14% faster than for all occupations.
• If you’re a DBA, you need to learn to deal with unstructured data

• If you’re a statistician, you need to learn to deal with data that does not fit in memory

• If you’re a software engineer, you need to learn statistical modeling and how to communicate results.
New challenges

task: Scaling up to Billion-Nodes Network using Map-Reduce

Very Hard!

Something is easily parallelized does NOT mean it can be easily “map-reduced”.

Big data processing $\neq$ Parallel data processing

How different?
Structured Data: RDBMS, DW SQL

**3V:** Data Volume, Variety, Velocity increase
=> Storage (DAS, NAS, SAN) cost increase,
Analysis is hard (unstructured >> structured)

Enterprise DBMS

Big Data Analysis System

Structured + Unstructured Data

- App
- Hive, Pig, R
- Hadoop
- HDFS, Swift
- HBase

Scale-out cluster
Big Data Analysis System

Structured + Unstructured Data

- Hive, Pig, R
- App

Hadoop

- HDFS, Swift
- HBase

App

DB or Data Access

High level Language

Distributed File System

Storage

Scale-out cluster
Network, distributed file system

- Proprietary, Highly reliable HW
  => Scale-up: Expensive
  => Fast data transfer
- Commodity HW
  => Scale-out: Inexpensive
  => Slow data transfer
  => Need new programming model!

Big data => Need scalability

Centralized storage: SAN, NAS

Distributed storage

Network, distributed file system
Data Trend (Big Data)

Storage Trend (Distributed): Inexpensive Scale-out, but Expensive Data Transfer!

Need New Programming Model to Minimize Data Transfer

Move operations instead of data!

MapReduce by Google

Hadoop and many subprojects
Design Tips

• Lower the work of reduce
  • Use combine if possible
• Compression of map’s output helps decreasing network overhead
• Minimize iterations and broadcasting
  • Sharing information is minimized
• Use bulk reading
  • Too many invocation of map may incur too many function calls
• Design algorithm to have enough reduce functions
  • Having only a single reduce will not speed up
  • ...

MapReduce Principles

• Run operation on data nodes: Move operations to Data
• Minimize data transfer

Programming is Hard!!!

A straightforward extension of parallel IPA algorithm produce too many iterations and heavy data transfer from map to reduce

* Independent Path Algorithm (IPA) in SNS influence analysis
Big data subprojects

• Big data programming framework
  • MapReduce (Batch): HDFS & Hadoop, Dryad
  • MapReduce (Iterative): HaLoop, Twister
  • MapReduce (Streaming): Storm (Twitter), S4 (Yahoo), InfoSphere Streams (IBM), HStreaming

• NoSQL DB
  • HBase (Master, slaves), Cassandra (P2P, “Gossip”, no master server), Dynamo (Amazon), MongoDB (for text)

• Graph processing engine
  • Pregel, Giraph, Trinity, Neo4J, TurboGraph

• IoT platform
  • NoSQL DB + Analytics solutions
  • Allseen, Predix
Big Data subprojects: MapReduce, NoSQL DB

• Minimize Data Transfer
• Which platform?
• Generalization
• Feasible? Approximate?
• Storage-aware mining

• Minimize Data Transfer
• Tasks: Search, Recommendation, ..
• Data: Text, Graph, Multimedia, ..
• Processing: Batch, Streaming
• Storage-aware platform

• Scalability
• Scale-out cost
• Energy efficiency
• Load balancing
• Heterogeneous storage
Reality

• Big data system is complex and slow.
• Big data is rare.
• Active data is small.
### What is data?

**Line P Cruise - GeomICS (May 2012)**

Nutrients sampled on board Thompson by members of the Ingalls and Deveau Labs (Laura Trulio, Davey Deveau, Katherine Hea)

1. Water sampled from CTD Niskin bottles unless otherwise indicated

<table>
<thead>
<tr>
<th>station#</th>
<th>Nutrient</th>
<th>depth (m)</th>
<th>Conc NO2 (nm)</th>
<th>Conc NH4 (nm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td>8</td>
<td>125</td>
<td>181.60</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>10</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>11</td>
<td>165</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>12</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>13</td>
<td>260</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>14</td>
<td>445</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>16</td>
<td>115</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>17</td>
<td>300</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>18</td>
<td>455</td>
<td>16.00</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>station#</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
</tr>
</tbody>
</table>

**#query**

<table>
<thead>
<tr>
<th>query</th>
<th>length</th>
<th>COGid</th>
<th>e-value</th>
<th>identity</th>
<th>score</th>
<th>hitlength</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr_4(480001-580000).287</td>
<td>4500</td>
<td>CG6457</td>
<td>2.00E-04</td>
<td>19</td>
<td>44.6</td>
<td>620</td>
<td>Cobalamin biosynthesis protein</td>
</tr>
<tr>
<td>chr_4(560001-660000).1</td>
<td>3556</td>
<td>CG6457</td>
<td>2.00E-04</td>
<td>38</td>
<td>43.9</td>
<td>1001</td>
<td>Nucleosome binding factor SPN</td>
</tr>
<tr>
<td>chr_9(400001-500000).503</td>
<td>4211</td>
<td>CG5406</td>
<td>5.00E-05</td>
<td>18</td>
<td>46.2</td>
<td>620</td>
<td>Cobalamin biosynthesis protein</td>
</tr>
<tr>
<td>chr_9(320001-420000).548</td>
<td>2833</td>
<td>CG5406</td>
<td>5.00E-05</td>
<td>17</td>
<td>46.2</td>
<td>777</td>
<td>RNA-binding protein of the Puf</td>
</tr>
<tr>
<td>chr_27(320001-404298).20</td>
<td>3991</td>
<td>CG5406</td>
<td>5.00E-05</td>
<td>17</td>
<td>46.2</td>
<td>777</td>
<td>RNA-binding protein of the Puf</td>
</tr>
<tr>
<td>chr_26(320001-420000).378</td>
<td>3963</td>
<td>CG5099</td>
<td>5.00E-05</td>
<td>17</td>
<td>46.2</td>
<td>777</td>
<td>RNA-binding protein of the Puf</td>
</tr>
<tr>
<td>chr_26(400001-441226).106</td>
<td>2949</td>
<td>CG5099</td>
<td>2.00E-04</td>
<td>17</td>
<td>43.9</td>
<td>777</td>
<td>RNA-binding protein of the Puf</td>
</tr>
<tr>
<td>chr_24(160001-260000).65</td>
<td>3542</td>
<td>CG5099</td>
<td>4.00E-09</td>
<td>20</td>
<td>59.3</td>
<td>777</td>
<td>RNA-binding protein of the Puf</td>
</tr>
<tr>
<td>chr_5(720001-820000).339</td>
<td>3141</td>
<td>CG5099</td>
<td>1.00E-09</td>
<td>26</td>
<td>114</td>
<td>1089</td>
<td>Ubiquitin carboxyl-terminal hyd</td>
</tr>
<tr>
<td>chr_9(160001-260000).243</td>
<td>3002</td>
<td>CG5077</td>
<td>1.00E-25</td>
<td>26</td>
<td>60.5</td>
<td>2105</td>
<td>Phosphatidylinositol kinase and</td>
</tr>
<tr>
<td>chr_12(720001-820000).86</td>
<td>2895</td>
<td>CG5032</td>
<td>1.00E-09</td>
<td>30</td>
<td>60.1</td>
<td>2105</td>
<td>Phosphatidylinositol kinase and</td>
</tr>
<tr>
<td>chr_12(800001-900000).109</td>
<td>1463</td>
<td>CG5032</td>
<td>1.00E-09</td>
<td>30</td>
<td>60.1</td>
<td>2105</td>
<td>Phosphatidylinositol kinase and</td>
</tr>
<tr>
<td>chr_11[1-100000].70</td>
<td>2886</td>
<td>CG5032</td>
<td>1.00E-09</td>
<td>30</td>
<td>60.1</td>
<td>2105</td>
<td>Phosphatidylinositol kinase and</td>
</tr>
<tr>
<td>chr_11(800001-180000).100</td>
<td>1523</td>
<td>CG5032</td>
<td>1.00E-09</td>
<td>30</td>
<td>60.1</td>
<td>2105</td>
<td>Phosphatidylinositol kinase and</td>
</tr>
</tbody>
</table>
Where to store data?
Data type and representation

1. Table and record
   • Relational database, transaction data
   • Matrix, cross table
   • Text documents as term-frequency vector

2. Graph and network
   • World Wide Web
   • Social or information networks
   • Molecular structures

3. Ordered data or sequence
   • Time-series, temporal data, sequence data
   • Data streams, sensor data
   • Natural language and text data

4. Spatial, Multimedia
   • Spatial data (map), spatiotemporal data
   • Multimedia: Image, video

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Coke, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Beer, Bread</td>
</tr>
<tr>
<td>3</td>
<td>Beer, Coke, Diaper, Milk</td>
</tr>
<tr>
<td>4</td>
<td>Beer, Bread, Diaper, Milk</td>
</tr>
<tr>
<td>5</td>
<td>Coke, Diaper, Milk</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>team</th>
<th>coach</th>
<th>year</th>
<th>play</th>
<th>bull</th>
<th>score</th>
<th>game</th>
<th>W</th>
<th>L</th>
<th>lost</th>
<th>timeout</th>
<th>season</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc 1</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Doc 2</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Doc 3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
What is data model?

Three components of data model

1. Structures
   - rows and columns?
   - nodes and edges?
   - key-value pairs?
   - a sequence of bytes?

2. Constraints
   - all rows must have the same number of columns
   - all values in one column must have the same type
   - a child cannot have two parents

3. Operations
   - find the value of key x
   - find the rows where column “lastname” is “Jordan”
   - get the next N bytes
What is *database*?

* A database is a collection of information organized to provide efficient retrieval.

[http://www.usg.edu/galileo/skills/unit04/primer04_01.phtml](http://www.usg.edu/galileo/skills/unit04/primer04_01.phtml)
Why do we want a *database*?

What problems do they solve?

1. **Sharing**  
   • Support concurrent access by multiple readers and writers

2. **Data model enforcement**  
   • Make sure all applications see clean, organized data

3. **Scalability**  
   • Work with datasets too large to fit in memory

4. **Flexibility**  
   • Use the data in new, unanticipated ways
Questions to consider

• How is the data physically organized on disk?

• What kinds of queries are efficiently supported by this organization and what kinds are not?

• How hard is it to update the data or add new data?

• What happens when I encounter new queries that I didn’t anticipate? Do I reorganize the data? How hard is that?
Historical example: network database
Historical example: hierarchical database

- Works great if you want to find all orders for a particular customer.
- What if you want to find all customers who ordered a Nail?
Relational database (Codd 1970)

“Relational Database Management Systems were invented to let you use one set of data in multiple ways, including ways that are unforeseen at the time the database is built and the 1st applications are written.” (Curt Monash, analyst/blogger)
Relational database (Codd 1970)

- Data is represented as a table.
- A database is represented as a set of tables.
- Every row in a table has the same columns.
- Relationships between tables are implicit: no pointers
- Processing is equivalent for
  - “find names registered for CSE344”
  - “find courses that Jane registered”
- Row: record, tuple, instance, object, ...
- Column: attribute, field, feature, dimension, ...

<table>
<thead>
<tr>
<th>Course</th>
<th>Student Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSE 344</td>
<td>223...</td>
</tr>
<tr>
<td>CSE 344</td>
<td>244...</td>
</tr>
<tr>
<td>CSE 514</td>
<td>255..</td>
</tr>
<tr>
<td>CSE 514</td>
<td>244...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Student Id</th>
<th>Student Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>223...</td>
<td>Jane</td>
</tr>
<tr>
<td>244...</td>
<td>Joe</td>
</tr>
<tr>
<td>255..</td>
<td>Susan</td>
</tr>
</tbody>
</table>
Attribute type

Attribute types are

- Nominal (or Categorical), e.g. Type of car, Color name
- Binary, e.g. Gender, Whether to have car or not
- Ordinal, e.g. Grade
- Numerical, e.g. Height, Temperature

Numerical could be

- Discrete, e.g. Integer
- Continuous, e.g. Real
Relational database in practice

- Pre-Relational: if your data changed, your application broke.
- Early RDBMS were buggy and slow (and often reviled), but required only 5% of the application code.

“Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed.” (Codd 1979)

- Key Ideas: Programs that manipulate tabular data exhibit an algebraic structure allowing reasoning and manipulation independently of physical data representation
Key idea: “Physical data independence”

```sql
SELECT seq
FROM ncbi_sequences
WHERE seq = 'GATTACGATATTA';
```

```c
f = fopen('table_file');
fseek(10030440);
while (True) {
    fread(&buf, 1, 8192, f);
    if (buf == GATTACGATATTA){
        ...
    }
}
```
Size of data

R, Matlab, SAS, Excel, ...

SQLite, MySQL, ...

Hadoop, Spark, NoSQL, SPARK, ...
What does “scalable” mean?

Operationally:
• In the past: “Works even if data doesn’t fit in main memory”
• Now: “Can make use of 1000s of cheap computers”

Algorithmically:
• In the past: “If you have $N$ data items, you must do no more than $N^m$ operations” -- polynomial time algorithms
• Now: “If you have $N$ data items, you must do no more than $N^m/k$ operations”, for some large $k$
  • Polynomial-time algorithms must be parallelized
• Soon: “If you have $N$ data items, you should do no more than $N \log N$ operations”
  • As data sizes go up, you may only get one pass at the data
  • The data is streaming -- you better make that one pass count
  • Ex: Large Synoptic Survey Telescope (30TB / night)
Relational database

• Databases are especially effective at “finding needle in haystack” by using indexes.

CREATE INDEX seq_index ON sequence(seq);

• Indexes are easily built and automatically used when appropriate.

SELECT seq,
FROM sequence
WHERE seq = ‘GATTACGATATTA’;
New task: Read trimming

- Given a set of DNA sequences
- Trim the final $n$ bps of each sequence
- Generate a new dataset
You are given short “reads”: genomic sequences about 35-75 characters each

Distribute the reads among $k$ computers

$f$ is a function to trim a read; apply it to every item

Now we have a big distributed set of trimmed reads
Abridged Declaration of Independence

A Declaration By the Representatives of the United States of America, in General Congress Assembled.

When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.

We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying its foundation on such principles and organizing its power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.
You have millions of documents

Distribute the documents among k computers

For each document f returns a set of (word, freq) pairs

Now we have a big distributed list of sets of word freqs

**New task: Compute word frequency of 5M documents**
Map function

There’s a pattern here...

• A function that *maps* a read to a trimmed read
• A function that *maps* a TIFF image to a PNG image
• A function that *maps* a set of parameters to a simulation result
• A function that *maps* a document to its most common word
• A function that *maps* a document to a histogram of word frequencies
What if we want to compute word frequency across all documents?
New task: Compute word frequency across 5M documents

You have millions of documents

Distribute the documents among $k$ computers

For each document, return a set of (word, freq) pairs

Now what?

But we don’t want a bunch of little histograms – we want one big histogram.

How can we make sure that a single computer has access to every occurrence of a given word regardless of which document it appeared in?

Condition: We have to avoid bottleneck as much as possible!
Compute word frequency across 5M documents

Distribute the documents among \( k \) computers

For each document, return a set of \((\text{word}, \text{freq})\) pairs

Now we have a big distributed list of sets of word freqs.

Now just count the occurrences of each word

We have our distributed histogram
MapReduce: A distributed algorithm framework

Map

*(Shuffle)*

Reduce

*split-apply-combine strategy*
Taxonomy of parallel architecture

*communication is through messages only; communication bottleneck

Scales to 1000s of computers

a) shared nothing
b) shared disc
c) shared memory

Fig. 3.1 Logical multi-processor database designs (diagram after [DEWI92])
Cluster computing

- Large number of commodity servers, connected by commodity network
- Rack: holds a small number of servers
- Data center: holds many racks
- Massive parallelism:
  - 100s, 1000s, or 10,000s servers
- Failure:
  - If mean-time-between-failure is 1 year,
  - then, 10,000 servers have one failure per hour
Distributed file system (DFS)

- For very large files: TBs, PBs
- Each file is partitioned into chunks, typically 64MB
- Each chunk is replicated several times (>=3) on different racks for fault tolerance
- Implementations:
  - Google’s DFS: GFS, proprietary
  - Hadoop’s DFS: HDFS, open source
Many tasks process big data, produce big data

Want to use hundreds or thousands of CPUs
  ... but this needs to be easy
  Parallel databases exist, but they are expensive, difficult to set up, and do not necessarily scale to hundreds of nodes.

MapReduce is a lightweight framework, providing:
  Automatic parallelization and distribution
  Fault-tolerance
  Status and monitoring
NoSQL: distributed data management system

No ACID but *eventual consistency*

- In absence of updates, all replicas converge towards identical copies
- What the application sees in the meantime is sensitive to replication mechanics and difficult to predict

*ACID (Atomicity, Consistency, Isolation, Durability): transaction reliability

*specific form of weak consistency: the storage system guarantees that if no new updates are made to the object, eventually all accesses will return the last updated value*
## Eventual consistency example

<table>
<thead>
<tr>
<th>Write</th>
<th>Update Sue’s status. Who sees the new status, and who sees the old one?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDBMS</td>
<td>“Everyone MUST see the same thing, either old or new, no matter how long it takes.”</td>
</tr>
<tr>
<td>NoSQL</td>
<td>“For large applications, we can’t afford to wait that long, and maybe it doesn’t matter anyway”</td>
</tr>
</tbody>
</table>
NoSQL: pros and cons

For whom?

• “I started with MySQL, but had a hard time scaling out in a distributed environment”
• “My data doesn’t conform to a rigid schema”

Cons:

• No ACID, thus screwing up mission-critical data is no!
• Low-level query language is hard to maintain.
• Distributed system is hard to maintain.
• NoSQL means no standards!
• A typical large enterprise has thousands of databases!