Introduction to Big Data
with Apache Spark
This Lecture

The Structure Spectrum

Files: Formats and Performance

Tabular Data: Examples, Challenges, pySpark DataFrames

Log Files
Review: The Big Picture

Application Database

Extract Transform Load

Data Warehouse

Data Products

Business Intelligence

Analytics
Key Data Management Concepts

• A *data model* is a collection of concepts for describing data

• A *schema* is a description of a particular collection of data, using a given data model
The Structure Spectrum

Structured (schema-first)
- Relational Database
- Formatted Messages

Semi-Structured (schema-later)
- Documents XML
- Tagged Text/Media

Unstructured (schema-never)
- Plain Text
- Media
The Structure Spectrum

Structured (schema-first)
Relational Database
Formatted Messages

Semi-Structured (schema-later)
Documents
XML
JSON
Tagged Text/Media

Unstructured (schema-never)
Plain Text
Media

This lecture
Files

• What is a file?
  » A file is a named sequence of bytes
    • Typically stored as a collection of pages (or blocks)
  » A filesystem is a collection of files organized within an hierarchical namespace
    • Responsible for laying out those bytes on physical media
    • Stores file metadata
    • Provides an API for interaction with files
  » Standard operations
    • open() / close()
    • seek()
    • read() / write()

Files: Hierarchical Namespace

• On Linux, / is the root of a filesystem
• On Windows, \ is the root of a filesystem
• Files and and directories have associated permissions
• Files are not always arranged in a hierarchically
  » Content-addressable storage (CAS)
  » Often used for large multimedia collections
Considerations for a File Format

- Data model: tabular, hierarchical, array
- Physical layout
- Field units and validation
- Metadata: header, side file, specification, other?
- Plain text (ASCII, UTF-8, other) or binary
- Delimiters and escaping
- Compression, encryption, checksums?
- Schema evolution
Semi-Structured Tabular Data

• **One of the most common data formats**
• A **table** is a collection of **rows** and **columns**
• Each row has an **index** and each column has a **name**
• A **cell** is specified by an (index, name) **pair**
• A cell may or may not have a **value**
• A cell’s **type** is inferred from its value
Tabular Data Example

- Fortune 500 companies
  - Top 500 US closely held and public corporations by gross revenue

http://fortune.com/fortune500/

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>rank</td>
<td>company</td>
<td>ticker</td>
<td>sic</td>
<td>state_location</td>
<td>state_of_incorporation</td>
<td>revenues</td>
<td>profits</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Wal-Mart Stores</td>
<td>104169</td>
<td>WMT</td>
<td>5331</td>
<td>AR</td>
<td>DE</td>
<td>421649</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Exxon Mobil</td>
<td>34088</td>
<td>XOM</td>
<td>2911</td>
<td>TX</td>
<td>NJ</td>
<td>354674</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Chevron</td>
<td>93410</td>
<td>CVX</td>
<td>2911</td>
<td>CA</td>
<td>DE</td>
<td>196337</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>ConocoPhillips</td>
<td>1163165</td>
<td>COP</td>
<td>2911</td>
<td>TX</td>
<td>DE</td>
<td>184966</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>Fannie Mae</td>
<td>310522</td>
<td>FNM</td>
<td>6111</td>
<td>DC</td>
<td>DC</td>
<td>153935</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>General Electric</td>
<td>40545</td>
<td>GE</td>
<td>3600</td>
<td>CT</td>
<td>NY</td>
<td>151628</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>Berkshire Hathaway</td>
<td>1007983</td>
<td>BRKA</td>
<td>6331</td>
<td>NE</td>
<td>DE</td>
<td>136185</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>General Motors</td>
<td>1467858</td>
<td>GM</td>
<td>3711</td>
<td>MI</td>
<td>MI</td>
<td>135592</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>Bank of America Corp.</td>
<td>70858</td>
<td>BAC</td>
<td>6021</td>
<td>NC</td>
<td>DE</td>
<td>134194</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>Ford Motor</td>
<td>37996</td>
<td>F</td>
<td>3711</td>
<td>MI</td>
<td>DE</td>
<td>126954</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>Hewlett-Packard</td>
<td>47217</td>
<td>HPQ</td>
<td>3570</td>
<td>CA</td>
<td>DE</td>
<td>126033</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>AT&amp;T</td>
<td>732717</td>
<td>T</td>
<td>4813</td>
<td>TX</td>
<td>DE</td>
<td>124629</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>J.P. Morgan Chase &amp; Co.</td>
<td>10617</td>
<td>JPM</td>
<td>6021</td>
<td>NY</td>
<td>DE</td>
<td>115475</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>Citigroup</td>
<td>831001</td>
<td>C</td>
<td>6021</td>
<td>NY</td>
<td>DE</td>
<td>111056</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>McKesson</td>
<td>927653</td>
<td>MCK</td>
<td>5122</td>
<td>CA</td>
<td>DE</td>
<td>108702</td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>Verizon Communications</td>
<td>732712</td>
<td>VZ</td>
<td>4813</td>
<td>TX</td>
<td>DE</td>
<td>106566</td>
</tr>
<tr>
<td>18</td>
<td>17</td>
<td>American International Group</td>
<td>5272</td>
<td>AIG</td>
<td>5331</td>
<td>NY</td>
<td>DE</td>
<td>104417</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>International Business Machines</td>
<td>51143</td>
<td>IBM</td>
<td>3570</td>
<td>NY</td>
<td>NY</td>
<td>99870</td>
</tr>
<tr>
<td>20</td>
<td>19</td>
<td>Cardinal Health</td>
<td>721371</td>
<td>CAH</td>
<td>5122</td>
<td>OH</td>
<td>OH</td>
<td>986019</td>
</tr>
<tr>
<td>21</td>
<td>20</td>
<td>Freddie Mac</td>
<td>37786</td>
<td>FMC</td>
<td>2800</td>
<td>PA</td>
<td>DE</td>
<td>96368</td>
</tr>
</tbody>
</table>
Exporting Tabular Data

- US Fortune 500

Export as Comma Separated Values

http://fortune.com/fortune500/
### Tabular Data (CSV File)

#### US Fortune 500

<table>
<thead>
<tr>
<th>Rank</th>
<th>Company</th>
<th>Ticker</th>
<th>Industry</th>
<th>State Location</th>
<th>State of Incorporation</th>
<th>Revenue</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Wal-Mart Stores</td>
<td>WMT</td>
<td>Retail</td>
<td>AR</td>
<td>DE</td>
<td>128349</td>
<td>4919</td>
</tr>
<tr>
<td>2</td>
<td>Exxon Mobil</td>
<td>XOM</td>
<td>Oil</td>
<td>TX</td>
<td>NJ</td>
<td>36474</td>
<td>4365</td>
</tr>
<tr>
<td>3</td>
<td>Chevron</td>
<td>CVX</td>
<td>Oil</td>
<td>CA</td>
<td>DE</td>
<td>35237</td>
<td>1402</td>
</tr>
<tr>
<td>4</td>
<td>ConocoPhillips</td>
<td>COP</td>
<td>Oil</td>
<td>TX</td>
<td>DE</td>
<td>13966</td>
<td>1358</td>
</tr>
<tr>
<td>5</td>
<td>Fortnite Game</td>
<td>FMV</td>
<td>Entertainment</td>
<td>DE</td>
<td>DE</td>
<td>83582</td>
<td>1481</td>
</tr>
<tr>
<td>6</td>
<td>General Electric</td>
<td>GE</td>
<td>Manufacturing</td>
<td>CT</td>
<td>NY</td>
<td>60592</td>
<td>1644</td>
</tr>
<tr>
<td>7</td>
<td>Berkshire Hathaway</td>
<td>BRK</td>
<td>Finance</td>
<td>NE</td>
<td>DE</td>
<td>151565</td>
<td>12967</td>
</tr>
<tr>
<td>8</td>
<td>General Motors</td>
<td>GM</td>
<td>Automotive</td>
<td>MI</td>
<td>MI</td>
<td>135952</td>
<td>6172</td>
</tr>
<tr>
<td>9</td>
<td>Bank of America Corp</td>
<td>BOA</td>
<td>Banking</td>
<td>CA</td>
<td>CA</td>
<td>131944</td>
<td>2206</td>
</tr>
<tr>
<td>10</td>
<td>Ford Motor</td>
<td>F</td>
<td>Automotive</td>
<td>MI</td>
<td>DE</td>
<td>129954</td>
<td>5061</td>
</tr>
<tr>
<td>11</td>
<td>Hewlett-Packard</td>
<td>HPQ</td>
<td>Technology</td>
<td>CA</td>
<td>CA</td>
<td>126563</td>
<td>9761</td>
</tr>
<tr>
<td>12</td>
<td>AT&amp;T</td>
<td>T</td>
<td>Communications</td>
<td>TX</td>
<td>DE</td>
<td>124620</td>
<td>1984</td>
</tr>
<tr>
<td>13</td>
<td>J.P. Morgan Chase &amp; Co</td>
<td>JPM</td>
<td>Banking</td>
<td>NY</td>
<td>DE</td>
<td>115475</td>
<td>17570</td>
</tr>
<tr>
<td>14</td>
<td>Citigroup</td>
<td>C</td>
<td>Banking</td>
<td>NY</td>
<td>DE</td>
<td>111065</td>
<td>18662</td>
</tr>
<tr>
<td>15</td>
<td>McKesson</td>
<td>MCK</td>
<td>Healthcare</td>
<td>CA</td>
<td>CA</td>
<td>108722</td>
<td>1263</td>
</tr>
<tr>
<td>16</td>
<td>Verizon Communications</td>
<td>VZ</td>
<td>Communications</td>
<td>NY</td>
<td>DE</td>
<td>106566</td>
<td>2643</td>
</tr>
<tr>
<td>17</td>
<td>American International Group</td>
<td>AIG</td>
<td>Insurance</td>
<td>NY</td>
<td>DE</td>
<td>104417</td>
<td>7785</td>
</tr>
<tr>
<td>18</td>
<td>International Business Machines</td>
<td>IBM</td>
<td>Technology</td>
<td>NY</td>
<td>NY</td>
<td>99378</td>
<td>14833</td>
</tr>
<tr>
<td>19</td>
<td>Cardinal Health</td>
<td>CAH</td>
<td>Healthcare</td>
<td>OH</td>
<td>OH</td>
<td>96019</td>
<td>642</td>
</tr>
<tr>
<td>20</td>
<td>Freddie Mac</td>
<td>FMCC</td>
<td>Finance</td>
<td>VA</td>
<td>DE</td>
<td>90950</td>
<td>14025</td>
</tr>
<tr>
<td>21</td>
<td>CVS Caremark</td>
<td>CVS</td>
<td>Healthcare</td>
<td>NJ</td>
<td>DE</td>
<td>96413</td>
<td>3427</td>
</tr>
<tr>
<td>22</td>
<td>UnitedHealthGroup</td>
<td>UHC</td>
<td>Healthcare</td>
<td>OH</td>
<td>OH</td>
<td>6324</td>
<td>4634</td>
</tr>
<tr>
<td>23</td>
<td>Wells Fargo</td>
<td>WFC</td>
<td>Banking</td>
<td>CA</td>
<td>DE</td>
<td>03249</td>
<td>22062</td>
</tr>
<tr>
<td>24</td>
<td>Valero Energy</td>
<td>VVR</td>
<td>Oil</td>
<td>TX</td>
<td>DE</td>
<td>68064</td>
<td>324</td>
</tr>
<tr>
<td>25</td>
<td>Kroger</td>
<td>KR</td>
<td>Retail</td>
<td>OH</td>
<td>OH</td>
<td>82160</td>
<td>1118</td>
</tr>
<tr>
<td>26</td>
<td>Procter &amp; Gamble</td>
<td>PG</td>
<td>Consumer Goods</td>
<td>OH</td>
<td>OH</td>
<td>7368</td>
<td>12736</td>
</tr>
<tr>
<td>27</td>
<td>AmerisourceBergen</td>
<td>ASB</td>
<td>Healthcare</td>
<td>PA</td>
<td>DE</td>
<td>77954</td>
<td>636</td>
</tr>
<tr>
<td>28</td>
<td>Costco Wholesale</td>
<td>COST</td>
<td>Retail</td>
<td>WA</td>
<td>WA</td>
<td>77945</td>
<td>1388</td>
</tr>
<tr>
<td>29</td>
<td>Nordstrom</td>
<td>JWN</td>
<td>Retail</td>
<td>WA</td>
<td>WA</td>
<td>69113</td>
<td>2506</td>
</tr>
<tr>
<td>30</td>
<td>Home Depot</td>
<td>HD</td>
<td>Retail</td>
<td>DC</td>
<td>DE</td>
<td>57957</td>
<td>3336</td>
</tr>
</tbody>
</table>

AN IMPROVED MODEL OF THE HUMAN APOPTOSOME

MOL_ID: 1;
MOLECULE: APOPTOTIC PROTEASE-ACTIVATING FACTOR 1;
CHAIN: A, B, C, D, E, F, G;
SYNONYM: APAF-1;
ENGINEERED: YES;
MOL_ID: 2;
MOLECULE: CYTOCHROME C;
CHAIN: H, I, J, K, L, M, N

ORGANISM_SCIENTIFIC: HOMO SAPIENS;
ORGANISM_COMMON: HUMAN;
ORGANISM_TAXID: 9606;
GENE: APAF-1, APAF1, KIAA0413;
EXPRESSION_SYSTEM: SPODOPTERA FRUGIPERDA;
EXPRESSION_SYSTEM_COMMON: FALL ARMYWORM;
APOPTOSIS PROTEASE ACTIVATING FACTOR-1, APAF-1, CYTOCHROME C;
APOPTOSIS

ELECTRON MICROSCOPY
S.YUAN,M.TOPF,C.W.AKEY
17-APR-13 3J2T 1 JRNL
10-APR-13 3J2T 0

http://www.rcsb.org/pdb/files/3J2T.pdb
Tabular Data

Several Challenges

• Format not well-defined (may be missing data values)
• Types may be incorrectly inferred ("2" versus "2.0")
• No support for versioning of format
• …
Tabular Data from Multiple Sources

Several Challenges

• May be missing fields (not every source provides same data)
• Inconsistent data types (one file has $ values another has £)
• Inconsistent values for same entity (Wal-Mart versus WalMart)
• …
Tabular Data from Sensors

Several Challenges

- May be missing fields (a given sensor may not produce all types)
- Sensor may be damaged (permanently or intermittently)
- Timestamps may not be accurate
- Other metadata (sensor location, ID) may have errors
- Sensor may go offline for a while
- …
**pandas**: Python Data Analysis Library

- Open source data analysis and modeling library
  - An alternative to using R

- pandas **DataFrame**: a table with named columns
  - The most commonly used pandas object
  - Represented as a Python **Dict** (column_name ➝ Series)
  - Each pandas **Series** object represents a column
    - 1-D labeled array capable of holding any data type
  - R has a similar **data frame** type
Semi-Structured Data in pySpark

- **DataFrames** introduced in Spark 1.3 as extension to RDDs
- **Distributed** collection of data organized into named columns
  » Equivalent to Pandas and R DataFrame, but distributed
- Types of columns inferred from values
pySpark and pandas DataFrames

• Easy to convert between Pandas and pySpark
  » Note: pandas DataFrame must fit in driver

# Convert Spark DataFrame to Pandas
pandas_df = spark_df.toPandas()

# Create a Spark DataFrame from Pandas
spark_df = context.createDataFrame(pandas_df)
pySpark DataFrame Performance

- Almost 5x pySpark performance on a single machine

https://databricks.com/blog/2015/02/17/introducing-dataframes-in-spark-for-large-scale-data-science.html
Semi-Structured Log Files

• Created by `printf` statements in server processes:
  » Web, database, network file servers, operating system components

• Human-readable text format files
  » Very rarely actually read by a human
  » Can store/archive in binary or compressed format

• Format published or “defined” by code
  » Can be very difficult to parse
Recall: Apache Web Server Log

uplherc.upl.com - - [01/Aug/1995:00:00:07 -0400] "GET / HTTP/1.0" 304 0
uplherc.upl.com - - [01/Aug/1995:00:00:08 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 304 0
uplherc.upl.com - - [01/Aug/1995:00:00:08 -0400] "GET /images/MOSAIC-logosmall.gif HTTP/1.0" 304 0
uplherc.upl.com - - [01/Aug/1995:00:00:08 -0400] "GET /images/USA-logosmall.gif HTTP/1.0" 304 0
ix-esc-ca2-07.ix.netcom.com - - [01/Aug/1995:00:00:09 -0400] "GET /images/launch-logo.gif HTTP/1.0" 200 1713
uplherc.upl.com - - [01/Aug/1995:00:00:10 -0400] "GET /images/WORLD-logosmall.gif HTTP/1.0" 304 0
slppp6.intermind.net - - [01/Aug/1995:00:00:10 -0400] "GET /history/skylab/skylab.html HTTP/1.0" 200 1687
piweba4y.prodigy.com - - [01/Aug/1995:00:00:10 -0400] "GET /images/launchmedium.gif HTTP/1.0" 200 11853
Apache Web Server Log Format

• **Apache Common Log Format** specifies log file format

• Example line from log file:
  ```
  127.0.0.1 - - [01/Aug/1995:00:00:01 -0400] "GET /images/launch-logo.gif HTTP/1.0"
  200 1839
  ```

• Components:
  » **127.0.0.1** Client IP address
Apache Web Server Log Format

- **Apache Common Log Format** specifies log file format

Example line from log file:

```plaintext
127.0.0.1 - - [01/Aug/1995:00:00:01 -0400] "GET /images/launch-logo.gif HTTP/1.0" 200 1839
```

- **Components:**
  - *User identity from remote machine*  
    (hyphen means not available)
  - *User identity from local logon*  
    (hyphen means not available)
Apache Web Server Log Format

- **Apache Common Log Format** specifies log file format

- Example line from log file:
  » 127.0.0.1 - - [01/Aug/1995:00:00:01 -0400] "GET /images/launch-logo.gif HTTP/1.0" 200 1839

- Components:
  » [01/Aug/1995:00:00:01 -0400]  
    *Request time*
Apache Web Server Log Format

- **Apache Common Log Format** specifies log file format

- Example line from log file:
  » 127.0.0.1 - - [01/Aug/1995:00:00:01 -0400] "GET /images/launch-logo.gif HTTP/1.0"
  » 200 1839

- **Components:**
  » "GET /images/launch-logo.gif HTTP/1.0"
    
    *Client request*
    
    - Request method (e.g., GET, POST, etc.)
    - Endpoint (a Uniform Resource Identifier)
    - Client protocol version
Apache Web Server Log Format

- **Apache Common Log Format** specifies log file format

- Example line from log file:
  
  » 127.0.0.1 - - [01/Aug/1995:00:00:01 -0400] "GET /images/launch-logo.gif HTTP/1.0" 200 1839

- Components:
  
  » **200** Status code the server sent back to the client
    
    • OK response (2xx), others: 3xx, 4xx, 5xx
  
  » **1839** Size of the object returned to client
    
    "-" if no content returned, or sometimes 0
Lab: Explore Web Server Access Log

• NASA HTTP server access log

• Log covers 21 days (1 Aug, 3 Aug – 22 Aug 1995)

• Log includes 1,043,177 requests

• Log partially cleaned for you
  » Removed some very hard to parse requests
Some Log Analysis Questions

• Overall:
  » What are the statistics for content being returned? Sizes, statuses?
  » What are the types of return codes?
  » How many 404 (page not found) errors are there?

• Temporal:
  » How many unique hosts per day?
  » How many requests per day?
  » On average, how many requests per host?
  » How many 404 errors per day?
dhcp-47-129:CS100_1> syslog -w 10
Feb 3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -
[EDAMAuthenticationResult read::]: unexpected field ID 6 with type 11. Skipping.
Feb 3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -
[EDAMAuthenticationResult read::]: unexpected field ID 7 with type 11. Skipping.
Feb 3 15:18:11 dhcp-47-129 Evernote[1140] <Warning>: -
[EDAMAccounting read::]: unexpected field ID 19 with type 8. Skipping.
Data Mining (“Splunking”)

- Collect files from many machines
  » Application and system event logs

- Check for unusual events:
  » Disk errors
  » Network congestion
  » Security attacks

- Monitor resources
  » Network, memory, disk, CPU, application queues

- Visualize with a dashboard
  » Splunk Free License

http://www.splunk.com
Considerations for a File Format

- Data model: tabular, hierarchical, array
- Physical layout
- Field units and validation
- Metadata: header, side file, specification, other?
- Plain text (ASCII, UTF-8, other) or binary
- Delimiters and escaping
- Compression, encryption, checksums?
- Schema evolution
File Performance Considerations

- Read versus write performance
- Plain text versus binary format
- Environment: Pandas (Python) versus Scala/Java
- Uncompressed versus compressed
## File Performance

*626 MB text file*

*787 MB binary file*

<table>
<thead>
<tr>
<th></th>
<th>Read Time (Text)</th>
<th>Write Time (Text)</th>
<th>Read Time (Binary)</th>
<th>Write Time (Binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pandas (Python)</td>
<td>36 secs</td>
<td>45 secs</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Scala/Java</td>
<td>18 secs</td>
<td>21 secs</td>
<td>1-6* secs</td>
<td>1-6* secs</td>
</tr>
</tbody>
</table>

** Read-Write Times Comparable**

** Pandas doesn't have a binary file I/O library
(Python performance depends on library you use)

* 6 seconds is the time for sustained read/write
(often faster due to system caching)
## File Performance

<table>
<thead>
<tr>
<th>File Type</th>
<th>Read Time (Text)</th>
<th>Write Time (Text)</th>
<th>Read Time (Binary)</th>
<th>Write Time (Binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pandas (Python)</td>
<td>36 secs</td>
<td>45 secs</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Scala/Java</td>
<td>18 secs</td>
<td>21 secs</td>
<td>1-6* secs</td>
<td>1-6* secs</td>
</tr>
</tbody>
</table>

** Binary I/O much faster than text

** Pandas doesn't have a binary file I/O library
(Python performance depends on library you use)

* 6 seconds is the time for sustained read/write
(often faster due to system caching)
## File Performance - Compression

### Binary File

<table>
<thead>
<tr>
<th>Compression Level</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (Java default)</td>
<td>4 secs</td>
<td>75 secs</td>
<td>286 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>4 secs</td>
<td>20 secs</td>
<td>313 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>4 secs</td>
<td>14 secs</td>
<td>328 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>2 secs</td>
<td>4 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw binary file</td>
<td>1-6 secs</td>
<td>1-6 secs</td>
<td>787 MB</td>
</tr>
</tbody>
</table>

### Text File

<table>
<thead>
<tr>
<th>Compression Level</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (default)</td>
<td>26 secs</td>
<td>98 secs</td>
<td>243 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>25 secs</td>
<td>46 secs</td>
<td>259 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>25 secs</td>
<td>33 secs</td>
<td>281 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>22 secs</td>
<td>24 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw text file</td>
<td>18 secs</td>
<td>21 secs</td>
<td>626 MB</td>
</tr>
</tbody>
</table>

*Write times much larger than read*

---

*Scala/Java language*
## File Performance - Compression

### Binary File

<table>
<thead>
<tr>
<th>Compression</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (Java default)</td>
<td>4 secs</td>
<td>75 secs</td>
<td>286 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>4 secs</td>
<td>20 secs</td>
<td>313 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>4 secs</td>
<td>14 secs</td>
<td>328 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>2 secs</td>
<td>4 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw binary file</td>
<td>1-6 secs</td>
<td>1-6 secs</td>
<td>787 MB</td>
</tr>
</tbody>
</table>

### Text File

<table>
<thead>
<tr>
<th>Compression</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (default)</td>
<td>26 secs</td>
<td>98 secs</td>
<td>243 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>25 secs</td>
<td>46 secs</td>
<td>259 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>25 secs</td>
<td>33 secs</td>
<td>281 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>22 secs</td>
<td>24 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw text file</td>
<td>18 secs</td>
<td>21 secs</td>
<td>626 MB</td>
</tr>
</tbody>
</table>

*Scala/Java language*

Large range of compression times
# File Performance - Compression

## Binary File

<table>
<thead>
<tr>
<th>Compression Type</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (Java default)</td>
<td>4 secs</td>
<td>75 secs</td>
<td>286 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>4 secs</td>
<td>20 secs</td>
<td>313 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>4 secs</td>
<td>14 secs</td>
<td>328 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>2 secs</td>
<td>4 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw binary file</td>
<td>1-6 secs</td>
<td>1-6 secs</td>
<td>787 MB</td>
</tr>
</tbody>
</table>

## Text File

<table>
<thead>
<tr>
<th>Compression Type</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (default)</td>
<td>26 secs</td>
<td>98 secs</td>
<td>243 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>25 secs</td>
<td>46 secs</td>
<td>259 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>25 secs</td>
<td>33 secs</td>
<td>281 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>22 secs</td>
<td>24 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw text file</td>
<td>18 secs</td>
<td>21 secs</td>
<td>626 MB</td>
</tr>
</tbody>
</table>

- **Large range of compression times**
- Scala/Java language
## File Performance - Compression

<table>
<thead>
<tr>
<th>Binary File</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (Java default)</td>
<td>4 secs</td>
<td>75 secs</td>
<td>286 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>4 secs</td>
<td>20 secs</td>
<td>313 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>4 secs</td>
<td>14 secs</td>
<td>328 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>2 secs</td>
<td>4 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw binary file</td>
<td>1-6 secs</td>
<td>1-6 secs</td>
<td>787 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text File</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (default)</td>
<td>26 secs</td>
<td>98 secs</td>
<td>243 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>25 secs</td>
<td>46 secs</td>
<td>259 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>25 secs</td>
<td>33 secs</td>
<td>281 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>22 secs</td>
<td>24 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw text file</td>
<td>18 secs</td>
<td>21 secs</td>
<td>626 MB</td>
</tr>
</tbody>
</table>

*Small range (15%) of compressed file sizes*
## File Performance - Compression

<table>
<thead>
<tr>
<th>Binary File</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (Java default)</td>
<td>4 secs</td>
<td>75 secs</td>
<td>286 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>4 secs</td>
<td>20 secs</td>
<td>313 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>4 secs</td>
<td>14 secs</td>
<td>328 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>2 secs</td>
<td>4 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw binary file</td>
<td>1-6 secs</td>
<td>1-6 secs</td>
<td>787 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text File</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (default)</td>
<td>26 secs</td>
<td>98 secs</td>
<td>243 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>25 secs</td>
<td>46 secs</td>
<td>259 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>25 secs</td>
<td>33 secs</td>
<td>281 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>22 secs</td>
<td>24 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw text file</td>
<td>18 secs</td>
<td>21 secs</td>
<td>626 MB</td>
</tr>
</tbody>
</table>

Scala/Java language

Binary I/O still much faster than text
## File Performance - Compression

<table>
<thead>
<tr>
<th>Binary File</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (Java default)</td>
<td>4 secs</td>
<td>75 secs</td>
<td>286 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>4 secs</td>
<td>20 secs</td>
<td>313 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>4 secs</td>
<td>14 secs</td>
<td>328 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>2 secs</td>
<td>4 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw binary file</td>
<td>1-6 secs</td>
<td>1-6 secs</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text File</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (default)</td>
<td>26 secs</td>
<td>98 secs</td>
<td>243 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>25 secs</td>
<td>46 secs</td>
<td>259 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>25 secs</td>
<td>33 secs</td>
<td>281 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>22 secs</td>
<td>24 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw text file</td>
<td>18 secs</td>
<td>21 secs</td>
<td>626 MB</td>
</tr>
</tbody>
</table>

*Scala/Java language*

Binary I/O still much faster than text.
# File Performance - Compression

<table>
<thead>
<tr>
<th>Binary File</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (Java default)</td>
<td>4 secs</td>
<td>75 secs</td>
<td>286 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>4 secs</td>
<td>20 secs</td>
<td>313 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>4 secs</td>
<td>14 secs</td>
<td>328 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>2 secs</td>
<td>4 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw binary file</td>
<td>1-6 secs</td>
<td>1-6 secs</td>
<td>787 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text File</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (default)</td>
<td>26 secs</td>
<td>98 secs</td>
<td>243 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>25 secs</td>
<td>46 secs</td>
<td>259 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>25 secs</td>
<td>33 secs</td>
<td>281 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>22 secs</td>
<td>24 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw text file</td>
<td>18 secs</td>
<td>21 secs</td>
<td>626 MB</td>
</tr>
</tbody>
</table>

LZ4 compression ≈ raw I/O speed

Scala/Java language
## File Performance - Compression

<table>
<thead>
<tr>
<th>Binary File</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (Java default)</td>
<td>4 secs</td>
<td>75 secs</td>
<td>286 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>4 secs</td>
<td>20 secs</td>
<td>313 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>4 secs</td>
<td>14 secs</td>
<td>328 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>2 secs</td>
<td>4 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw binary file</td>
<td>1-6 secs</td>
<td>1-6 secs</td>
<td>787 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text File</th>
<th>Read Time</th>
<th>Write Time</th>
<th>File Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gzip level 6 (default)</td>
<td>26 secs</td>
<td>98 secs</td>
<td>243 MB</td>
</tr>
<tr>
<td>Gzip level 3</td>
<td>25 secs</td>
<td>46 secs</td>
<td>259 MB</td>
</tr>
<tr>
<td>Gzip level 1</td>
<td>25 secs</td>
<td>33 secs</td>
<td>281 MB</td>
</tr>
<tr>
<td>LZ4 fast</td>
<td>22 secs</td>
<td>24 secs</td>
<td>423 MB</td>
</tr>
<tr>
<td>Raw text file</td>
<td>18 secs</td>
<td>21 secs</td>
<td>626 MB</td>
</tr>
</tbody>
</table>

*Scala/Java language*

LZ4 compression ≈ raw I/O speed
File Performance - Summary

• Uncompressed read and write times are comparable

• Binary I/O is much faster than text I/O

• Compressed reads much faster than compressed writes
  » LZ4 is better than gzip
  » LZ4 compression times approach raw I/O times