MATLAB EXPO 2016
Navigating Big Data with MATLAB
Isaac Noh
Application Engineer
How big is big?
What does “Big Data” even mean?

“Big data is a term for data sets that are so large or complex that traditional data processing applications are inadequate to deal with them.”

Wikipedia
So, what’s the (big) problem?

- Traditional tools and approaches won’t work
  - **Getting** the data is hard; **processing** it is even harder
  - Need to learn **new tools** and **new coding styles**
  - Have to rewrite algorithms, often at a lower level of abstraction

- Quality of your results can be impacted
  - e.g., by being forced to work on a subset of your data
Big Data workflow

ACCESS
More data and collections of files than fit in memory

PROCESS AND ANALYZE
Adapt traditional processing tools or learn new tools to work with Big Data

SCALE
To Big Data systems like Hadoop

MATLAB EXPO 2016
Big solutions

Wouldn’t it be nice if you could:

- Easily access data however it is stored
- Prototype algorithms quickly using a local workstation
- Scale up to big data sets running on large clusters
- **Using the same intuitive MATLAB syntax you are used to**
tall arrays R2016b

- For data that doesn’t fit into memory
- Lots of observations (hence “tall”)
- Looks like a normal MATLAB array
  - Supports numeric types, tables, datetimes, strings, etc…
  - Supports basic math, stats, indexing, etc.
  - Statistics and Machine Learning Toolbox support
    (clustering, classification, etc.)
tall arrays  R2016b

- Data is in one or more files
- Typically tabular data
- Files stacked vertically
- Data doesn’t fit into memory (even cluster memory)
tall arrays R2016b

- Automatically breaks data up into small “chunks” that fit in memory
**tall arrays R2016b**

- “Chunk” processing is handled automatically
- Processing code for tall arrays is the same as ordinary arrays
### tall arrays **R2016b**

- With Parallel Computing Toolbox, process several “chunks” at once
- Can scale up to clusters with MATLAB Distributed Computing Server
Example: Working with Big Data in MATLAB

- **Objective:** Create a model to predict the cost of a taxi ride in New York City

- **Inputs:**
  - Monthly taxi ride log files
  - The local data set is **small** (~2 MB)
  - The full data set is **big** (~25 GB)

- **Approach:**
  - Preprocess and explore data
  - Develop and validate predictive model (linear fit)
    - Work with subset of data for prototyping
    - Scale to full data set on HDFS
### Example: Prototyping

**Preview Data**

#### Description
- **Location:** New York City
- **Date(s):** (Partial) January 2015
- **Data size:** “small data” 13,693 rows / ~2 MB

```matlab
>> ds = datastore('taxidataNYC_1_2015.csv');
>> preview(ds)
```

<table>
<thead>
<tr>
<th>VendorID</th>
<th>tep_pickup_datetime</th>
<th>tep_dropoff_datetime</th>
<th>passenger_count</th>
<th>trip_distance</th>
<th>pickup_long</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2015-01-09 02:53:26</td>
<td>2015-01-09 03:01:26</td>
<td>1</td>
<td>1.43</td>
<td>-74.004</td>
</tr>
<tr>
<td>2</td>
<td>2015-01-25 05:29:56</td>
<td>2015-01-25 06:03:40</td>
<td>1</td>
<td>10.74</td>
<td>-73.998</td>
</tr>
<tr>
<td>1</td>
<td>2015-01-05 13:00:31</td>
<td>2015-01-05 13:03:45</td>
<td>2</td>
<td>0.5</td>
<td>-74.007</td>
</tr>
<tr>
<td>1</td>
<td>2015-01-26 11:47:23</td>
<td>2015-01-26 11:51:02</td>
<td>1</td>
<td>0.5</td>
<td>-73.997</td>
</tr>
<tr>
<td>2</td>
<td>2015-01-17 22:49:44</td>
<td>2015-01-17 22:57:01</td>
<td>2</td>
<td>1.3</td>
<td>-73.979</td>
</tr>
<tr>
<td>2</td>
<td>2015-01-19 06:01:36</td>
<td>2015-01-19 06:34:16</td>
<td>1</td>
<td>20.32</td>
<td>-73.975</td>
</tr>
</tbody>
</table>
## Example: Prototyping

Create a Tall Array

```matlab
>> tt = tall(ds)
tt =
   Mx19 tall table
   _______    __________    ___________    _______    _______    _______    _______
   VendorID tpep_pickup_datetime tpep_dropoff_datetime passenger_count trip_distance pickup_long
             ___________ ___________ ___________ _______ _______ _______ ___________ ___________
   2        2015-01-09 02:53:26 2015-01-09 03:01:26 1    1.43    -74.004
   2        2015-01-25 05:29:56 2015-01-25 06:03:40 1    10.74   -73.998
   1        2015-01-05 13:00:31 2015-01-05 13:03:45 2    0.5     -74.007
   1        2015-01-14 11:47:23 2015-01-14 11:51:02 1    0.5     -73.997
   2        2015-01-17 12:17 2015-01-17 22:57:01 2    1.3     -73.979
   2        2015-01-19 06:34:16 2015-01-19 06:34:16 1    20.32   -73.975
   2        2015-01-26 16:34:16 2015-01-26 16:03:06 5    4.48    -73.966
```

Input data is tabular – result is a tall table

Number of rows is unknown until all the data has been read

Only the first few rows are displayed
Example: Prototyping
Calling Functions with a Tall Array

Once the tall table is created, can process much like an ordinary table

```
% Calculate average trip duration
mnTrip = mean(tt.trip_minutes,'omitnan')

mnTrip =
    tall double
?
Preview deferred. Learn more.

% Execute commands and gather results into workspace
mn = gather(mnTrip)

Evaluating tall expression using the Local MATLAB Session:
  - Pass 1 of 1: Completed in 4 sec
  Evaluation completed in 5 sec

mn =
    15.2648
```

- Most results are evaluated only when explicitly requested (e.g., `gather`)
- MATLAB automatically optimizes queued calculations to minimize the number of passes through the data
Example: Prototyping
Preprocess, clean, and explore data

% Remove some bad data
tt.speed_mph = tt.trip_distance ./ (tt.trip_minutes ./ 60);
ignore = tt.trip_minutes <= 1 | ... % really short
    tt.trip_minutes >= 60 * 12 | ... % unfeasibly long
    tt.trip_distance <= 1 | ... % really short
    tt.trip_distance >= 12 * 55 | ... % unfeasibly far
    tt.speed_mph > 55 | ... % unfeasibly fast
    tt.total_amount < 0 | ... % negative fares?!
    tt.total_amount > 10000; % unfeasibly large fares
tt(ignore, :) = []; % Explore data
figure
    histogram(tt.trip_distance,'BinLimits',[0 30])
    title('Trip Distance')

Evaluating tall expression using the Local MATLAB Session:
- Pass 1 of 2: Completed in 6 sec
- Pass 2 of 2: Completed in 6 sec
Evaluation completed in 12 sec
Example: Prototyping

Fit predictive model

```matlab
% Fit predictive model
model = fitlm(ttTrain,'fare_amount ~ 1 + hr_of_day + trip_distance*trip_minutes')
```

Evaluating tall expression using the Local MATLAB Session:
- Pass 1 of 1: Completed in 7 sec
Evaluation completed in 8 sec

```matlab
model =
Compact linear regression model:
  fare_amount ~ 1 + hr_of_day + trip_distance*trip_minutes
```

Estimated Coefficients:

<table>
<thead>
<tr>
<th></th>
<th>Estimate</th>
<th>SE</th>
<th>tStat</th>
<th>pValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>2.8167</td>
<td>0.038002</td>
<td>74.12</td>
<td>0</td>
</tr>
<tr>
<td>trip_distance</td>
<td>2.2207</td>
<td>0.006166</td>
<td>360.16</td>
<td>0</td>
</tr>
<tr>
<td>hr_of_day</td>
<td>0.001222</td>
<td>0.0019124</td>
<td>0.63901</td>
<td>0.52282</td>
</tr>
<tr>
<td>trip_minutes</td>
<td>0.24528</td>
<td>0.001793</td>
<td>136.79</td>
<td>0</td>
</tr>
<tr>
<td>trip_distance:trip_minutes</td>
<td>-0.00053185</td>
<td>0.00012339</td>
<td>-4.3102</td>
<td>1.6336e-05</td>
</tr>
</tbody>
</table>

Number of observations: 58793, Error degrees of freedom: 58788
Root Mean Squared Error: 3.06
R-squared: 0.927, Adjusted R-Squared 0.927
F-statistic vs. constant model: 1.86e+05, p-value = 0
Example: Prototyping
Predict and validate model

```matlab
% Predict and validate
yPred = predict(model,ttValidation);
residuals = yPred - ttValidation.fare_amount;
figure
histogram(residuals,'Normalization','pdf','BinLimits',[-10 10])
```

Evaluating tall expression using the Local MATLAB Session:
- Pass 1 of 2: Completed in 8 sec
- Pass 2 of 2: Completed in 5 sec
Evaluation completed in 15 sec
# Scale to the Entire Data Set

## Description

- **Location:** New York City
- **Date(s):** All of 2015
- **Data size:** "Big Data" 150,000,000 rows / ~25 GB
Example: “small data” processing vs. Big Data processing

% Access the data
ds = datastore('taxiDataNYC_1_2015.csv');
tt = tall(ds);

% Access the data
ds = datastore('taxiData\*.csv');
tt = tall(ds);

% Calculate average trip duration
mnTrip = mean(tt.trip_minutes,'omitnan')

% Calculate average trip duration
mnTrip = mean(tt.trip_minutes,'omitnan')

% Execute commands and gather results into workspace
mn = gather(mnTrip)

% Execute commands and gather results into workspace
mn = gather(mnTrip)

% Remove some bad data
tt.trip_minutes = minutes(tt.tpep_dropoff_datetime - tt.tpep_pickup_datetime);
tt.speed_mph = tt.trip_distance ./ (tt.trip_minutes ./ 60);
ignore = tt.trip_minutes <= 1 | ... % really short
  tt.trip_minutes >= 60 * 12 | ... % unfeasibly long
  tt.trip_distance <= 1 | ... % really short
  tt.trip_distance >= 12 * 55 | ... % unfeasibly far
  tt.speed_mph > 55 | ... % unfeasibly fast
  tt.total_amount < 0 | ... % negative fares?! tt.total_amount > 10000; % unfeasibly large fares

% Remove some bad data
tt.trip_minutes = minutes(tt.tpep_dropoff_datetime - tt.tpep_pickup_datetime);
tt.speed_mph = tt.trip_distance ./ (tt.trip_minutes ./ 60);
ignore = tt.trip_minutes <= 1 | ... % really short
  tt.trip_minutes >= 60 * 12 | ... % unfeasibly long
  tt.trip_distance <= 1 | ... % really short
  tt.trip_distance >= 12 * 55 | ... % unfeasibly far
  tt.speed_mph > 55 | ... % unfeasibly fast
  tt.total_amount < 0 | ... % negative fares?! tt.total_amount > 10000; % unfeasibly large fares

tt(ignore,:)=[];

MATLAB EXPO 2016
Example: Running on Spark + Hadoop

```matlab
% Hadoop/Spark Cluster
numWorkers = 16;

setenv('HADOOP_HOME', '/dev_env/cluster/hadoop');
setenv('SPARK_HOME', '/dev_env/cluster/spark');

cluster = parallel.cluster.Hadoop;
cluster.SparkProperties('spark.executor.instances') = num2str(numWorkers);
mr = mapreduce(cluster);

% Access the data
ds = datastore('hdfs://hadoop01:54310/datasets/taxiData/*.csv');
tt = tall(ds);
```
Summary for tall arrays

Process out-of-memory data on your Desktop to explore, analyze, gain insights and to develop analytics

Use Parallel Computing Toolbox for increased performance

Run on Compute Clusters or Spark + Hadoop (HDFS), for large scale analysis

MATLAB Distributed Computing Server, Spark+Hadoop

MATLAB EXPO 2016
Big Data capabilities in MATLAB

**ACCESS**
Access data and collections of files that do not fit in memory

**Datastores**
- Images
- Spreadsheets
- Tabular Text
- Custom Files
- SQL
- Hadoop (HDFS)

**SCALE**
Scale to compute clusters and Hadoop/Spark for data stored in HDFS

- Tall Arrays
  - Math
  - Statistics
  - Visualization
  - Machine Learning

- GPU Arrays
  - Matrix Math
  - Image Processing

- Deep Learning
  - Image Classification

**PROCESS AND ANALYZE**
Purpose-built capabilities for domain experts to work with big data locally

MATLAB EXPO 2016
Summary

- MATLAB makes it easy, convenient, and scalable to work with big data
  - **Access** any kind of big data from any file system
  - Use tall arrays to **process and analyze** that data on your desktop, clusters, or on Hadoop/Spark

There’s no need to learn big data programming or out-of-memory techniques -- simply use the same code and syntax you're already used to.
For more information

- Advanced Data Analytics with MATLAB kiosk
- Website: https://www.mathworks.com/solutions/big-data-matlab
- Web search for: “Big Data MATLAB”