Tackling Big Data with MATLAB

Yi Wang
Manager, Application Engineering
MathWorks, Inc.

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
  - Accessing 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
  - Learning new tools and rewriting algorithms can hurt productivity
- Time required to conduct analysis
  - Need to leverage parallel computing on desktop and cluster

Big Data workflow

1 ACCESS DATA

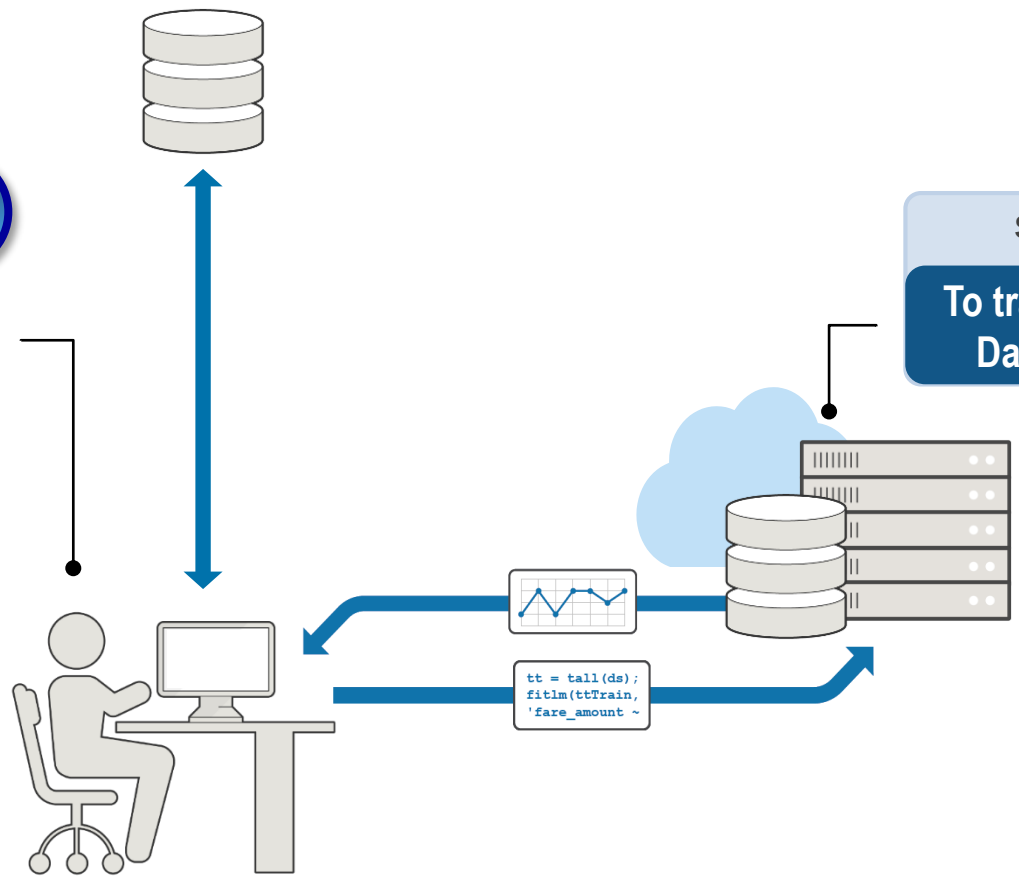
More data and collections of files than fit in memory

2 PROCESS ON THE DESKTOP

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

3 SCALE PROBLEM SIZE

To traditional clusters and Big Data systems like Hadoop



Big solutions

Wouldn’t it be nice if you could:

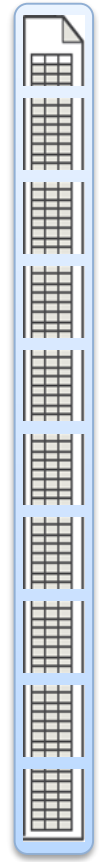
- Easily access data however it is stored
- Prototype algorithms quickly using small data sets
- Scale up to big data sets running on large clusters
- **Using the same intuitive MATLAB syntax you are used to**

Agenda

- New tall arrays for easily working with big data
- Distributed arrays for big matrix operations
- Other MATLAB tools for big data
- Summary

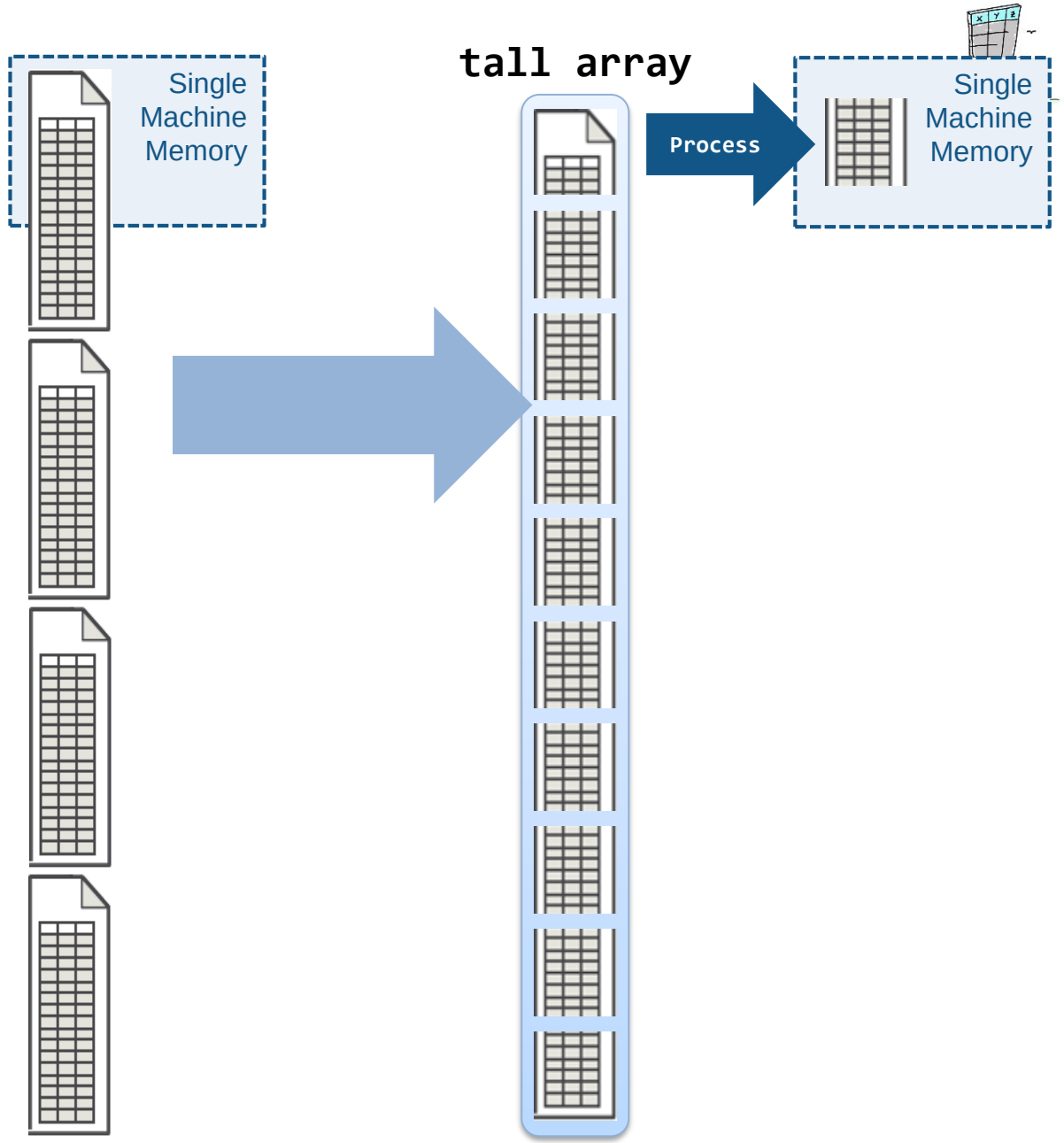
`tall` arrays R2016b

- New data type designed 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 several hundred functions for basic math, stats, indexing, etc.
  - **Statistics and Machine Learning Toolbox** support (clustering, classification, etc.)



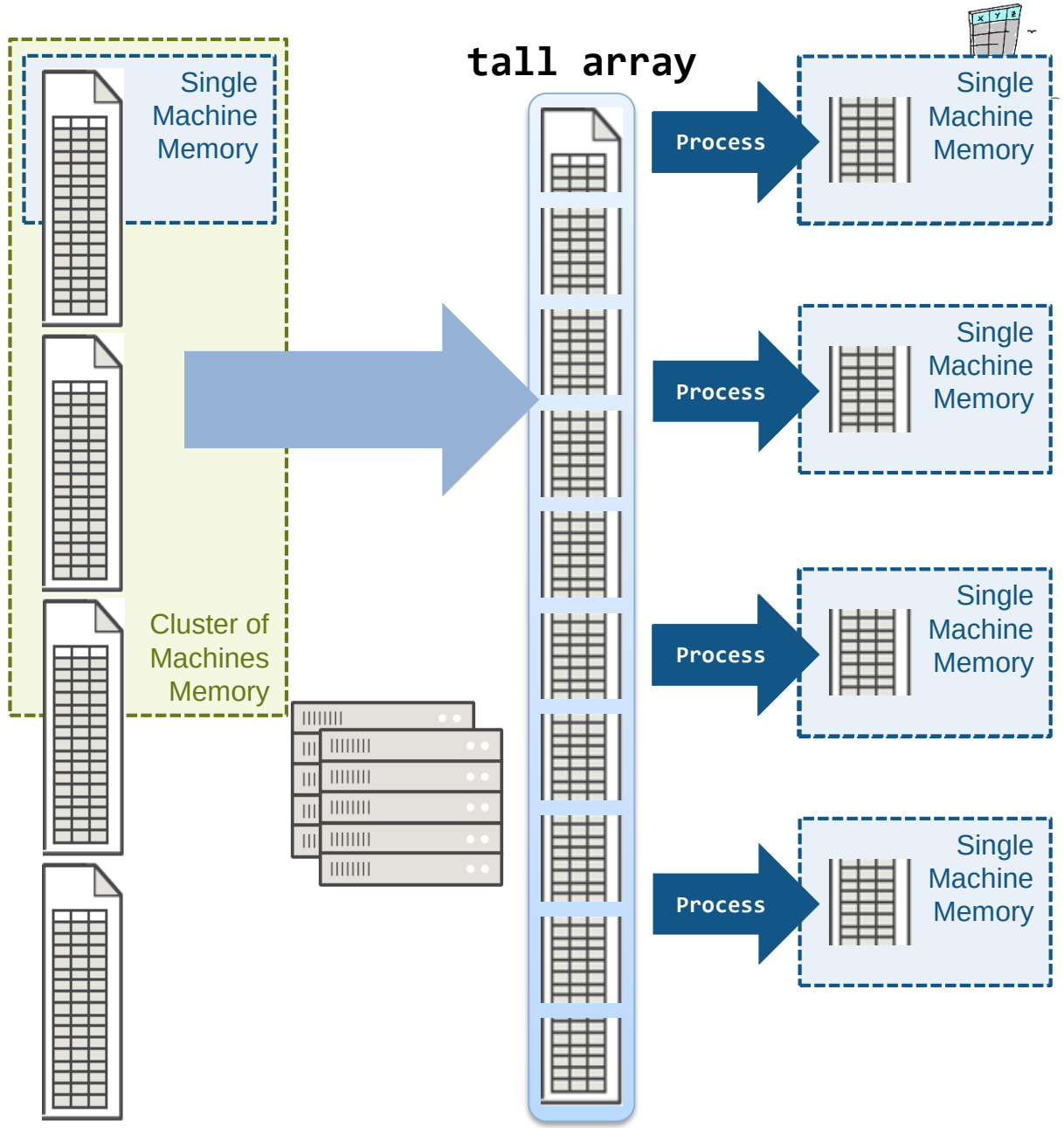
`tall` arrays R2016b

- Automatically breaks data up into small “chunks” that fit in memory
- Tall arrays scan through the dataset one “chunk” at a time
- 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** (~20 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 a cluster

Example: Running on Spark + Hadoop

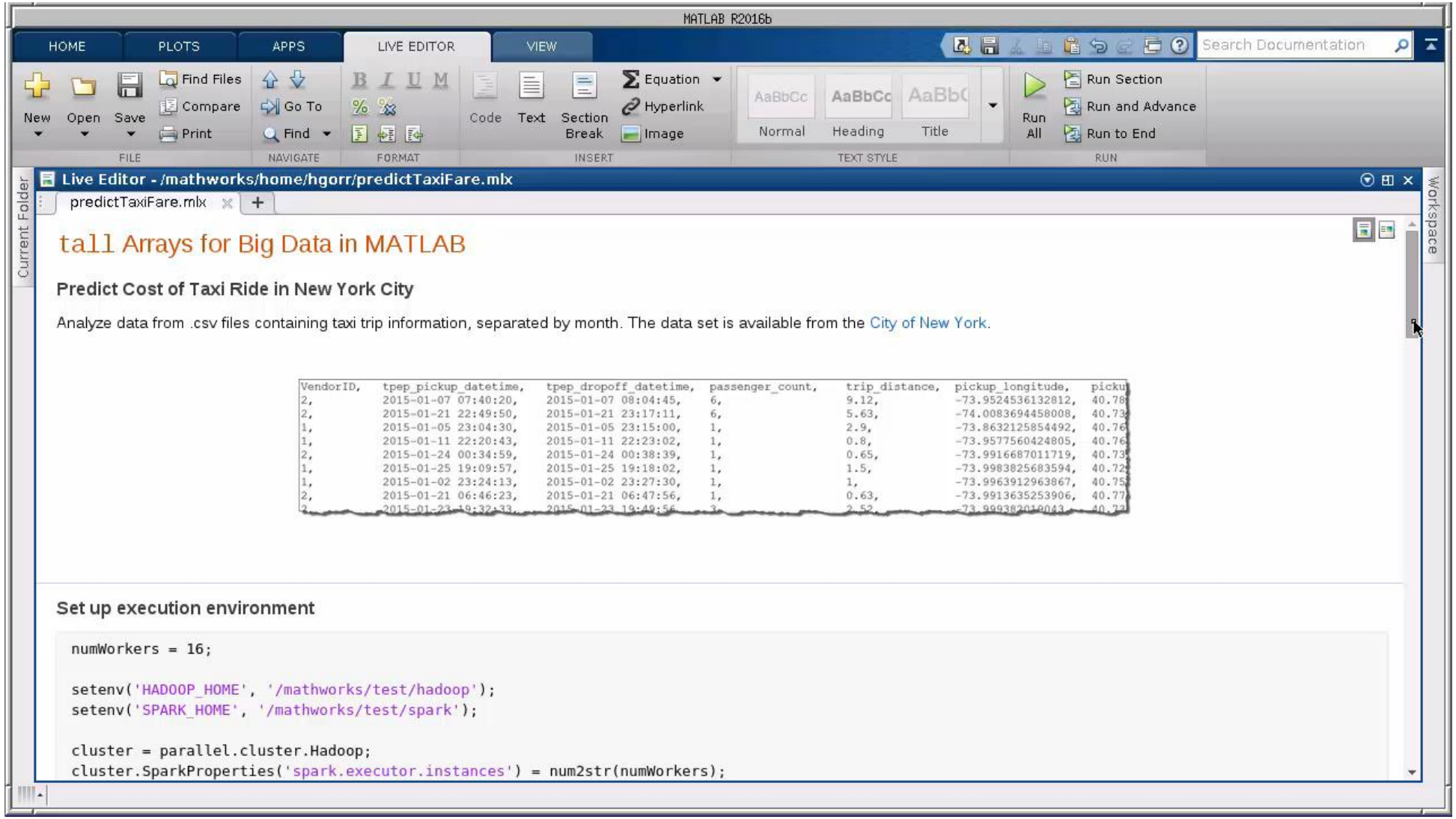
```
% Hadoop/Spark Cluster
numWorkers = 16;

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 = mapreducer(cluster);


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

Example: Running on Spark + Hadoop



Summary of `tall` Array Capabilities

Provides purpose-built functions for use with data that does not fit in memory

Data Access

ASCII File
Database (SQL)
Spreadsheet
Custom Files

- table
- cell
- numeric
- cellstr & string
- Date & Time
- categorical

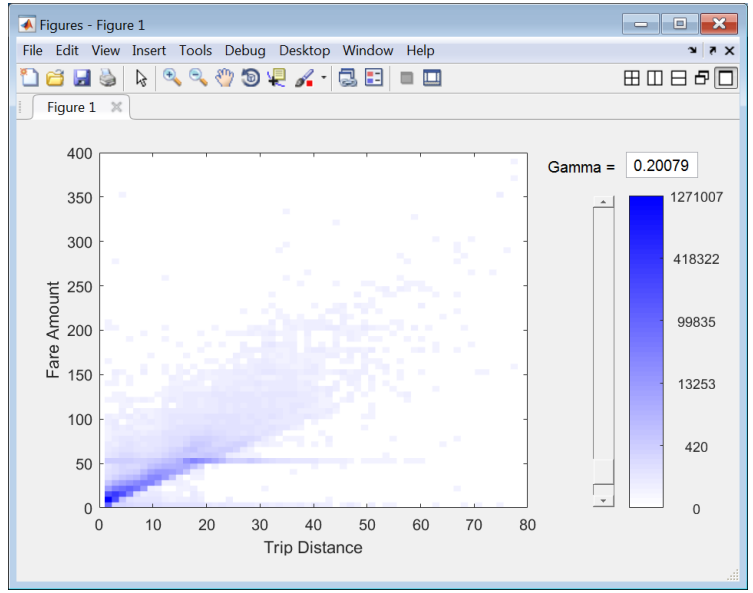
Data Munging

(100's of functions)

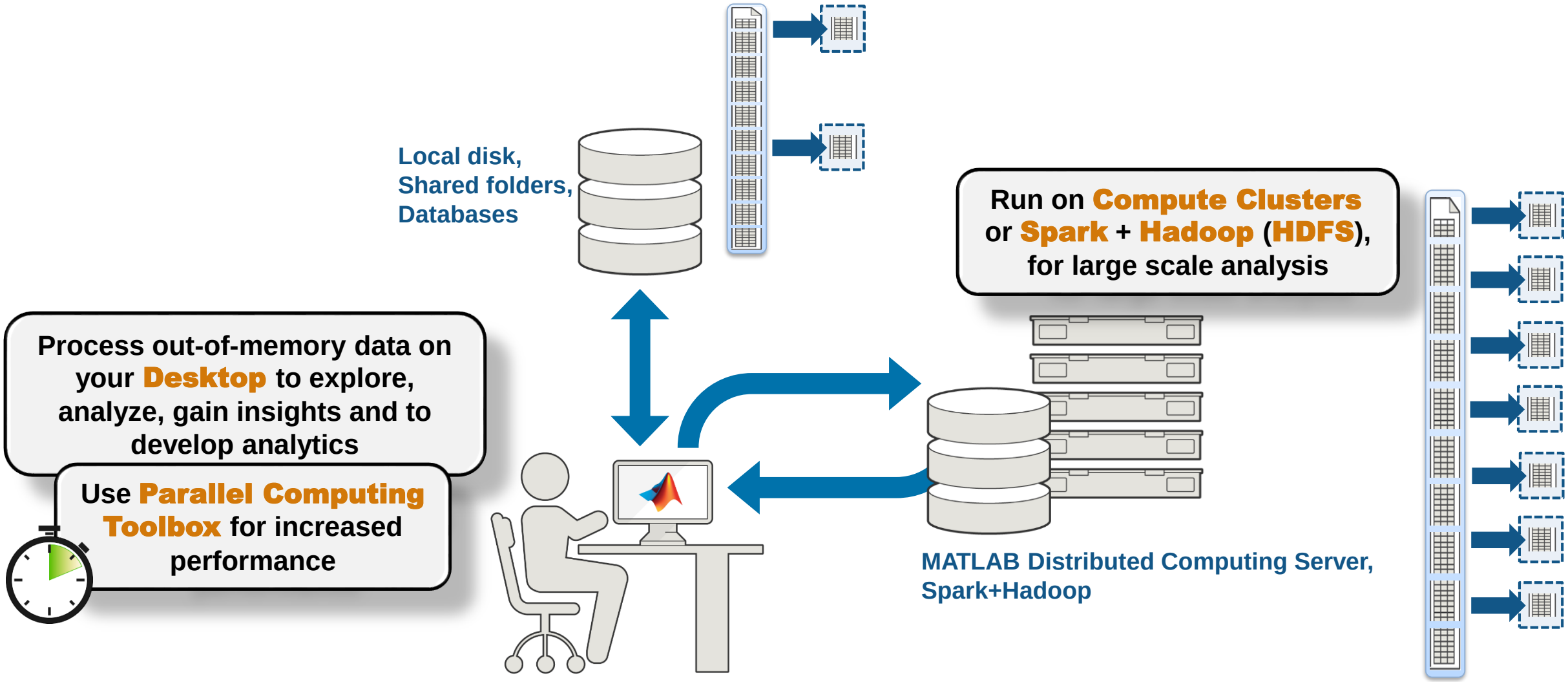
Math
Statistics
Missing Data
Visualization
Date/Time
String

Machine Learning

Linear Model
Logistic Regression
Discriminant Analysis
K-means
PCA
Random Data Sampling
Summary Statistics

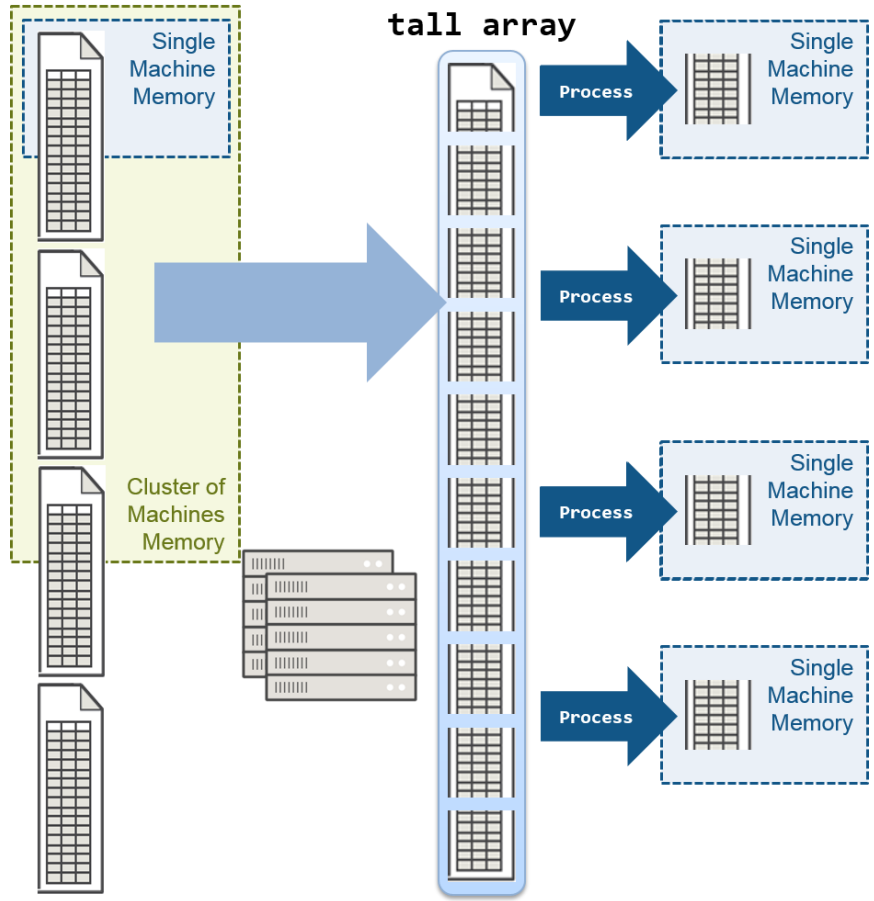


Summary for `tall` arrays



When to Use `tall` Arrays

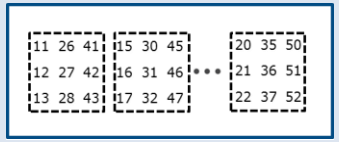
- Data Characteristics
  - Columnar data stored in files that are too big to fit into memory
- Compute Platform
  - Work with out-of-memory problems on the desktop
  - Parallelize on the desktop or a traditional cluster
  - Run on a Spark + Hadoop cluster
- Analysis Characteristics
  - Consists of functions supported by tall arrays (data preprocessing, visualizations, stats, machine learning, etc.)



Agenda

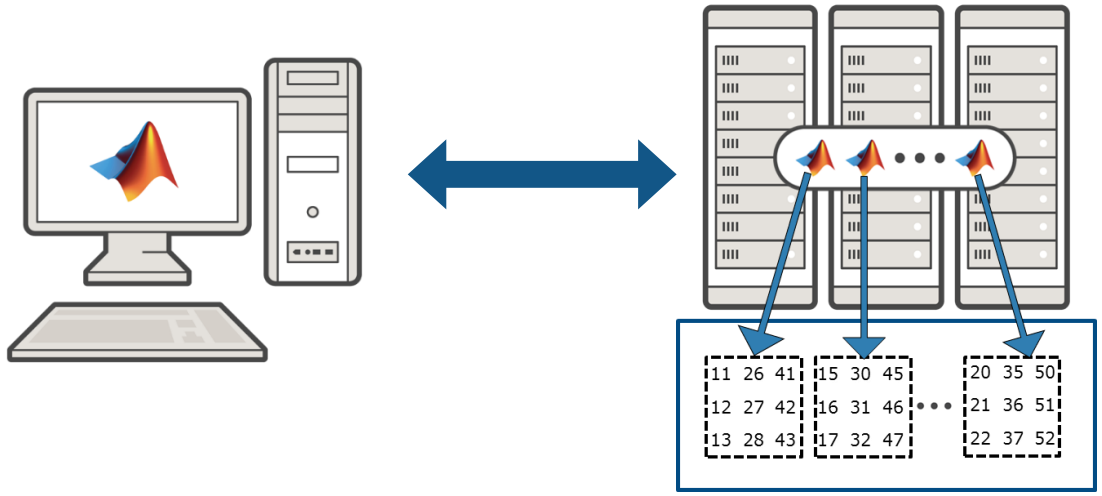
- New tall arrays for easily working with big data
- Distributed arrays for big matrix operations
- Other MATLAB tools for big data
- Summary

MATLAB Array Types for Working with Big Data

tall array (R2016b)	 **Process in segments on a desktop, cluster or Spark+Hadoop**	**Wide range of MATLAB data processing, statistics and machine learning algorithms, including regression, classification and clustering methods**
distributed array (R2006b)	 **Process in combined memory of a cluster**	**Rich subset of the core MATLAB language including linear algebra, convolutions and filtering, signal processing, and more**

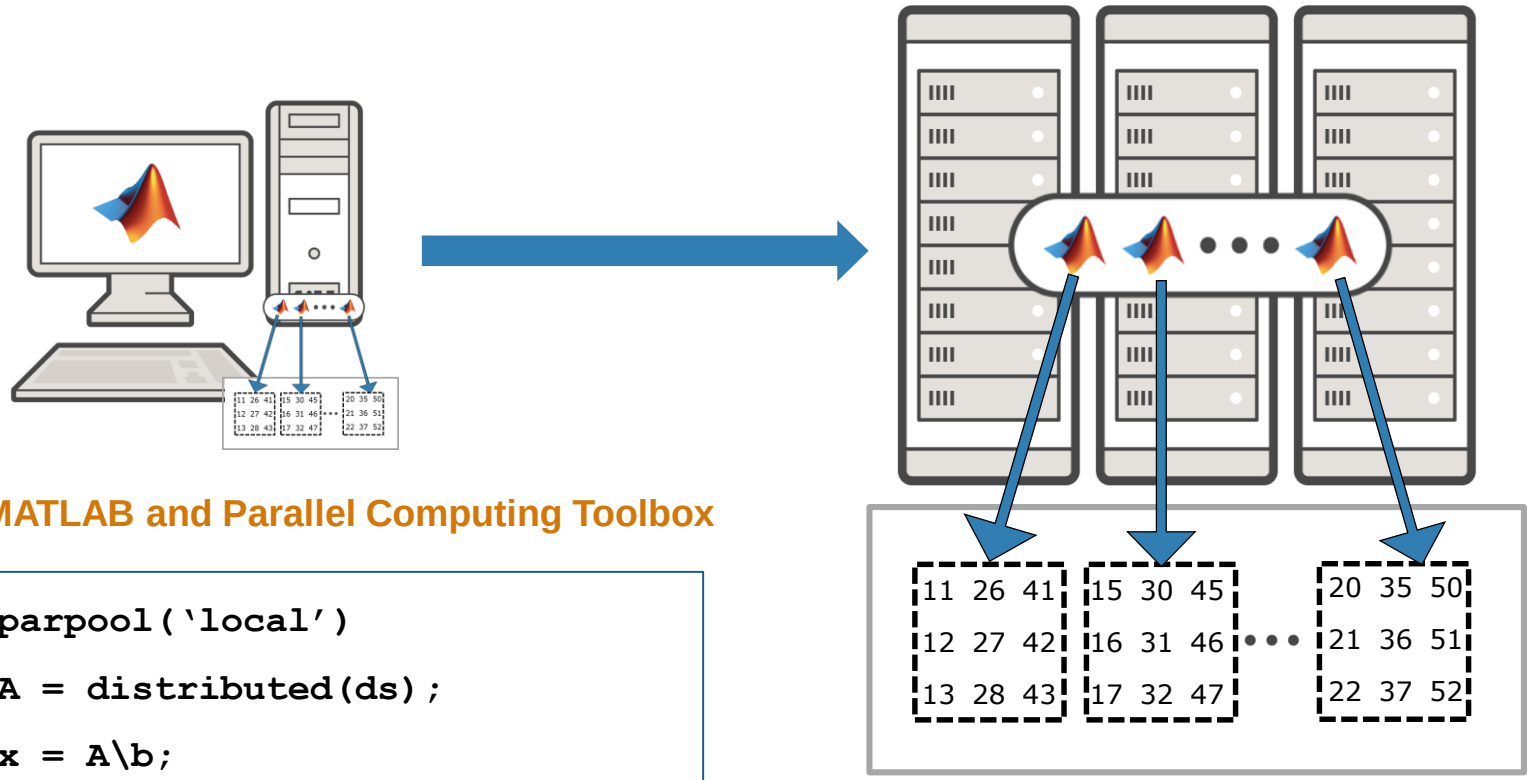
Distributed Arrays

- Distributed Arrays keep large datasets in-memory, split among workers running on a cluster
- Manipulate directly from client MATLAB (desktop)
- Several hundred MATLAB functions overloaded for distributed arrays
- Create from a datastore R2016b



Supported functions for Distributed Arrays

Distributed Arrays



MATLAB and Parallel Computing Toolbox

```
parpool('local')
A = distributed(ds);
x = A\b;
xg = gather(x);
% prototype with small A,b
% A,b are distributed arrays
```

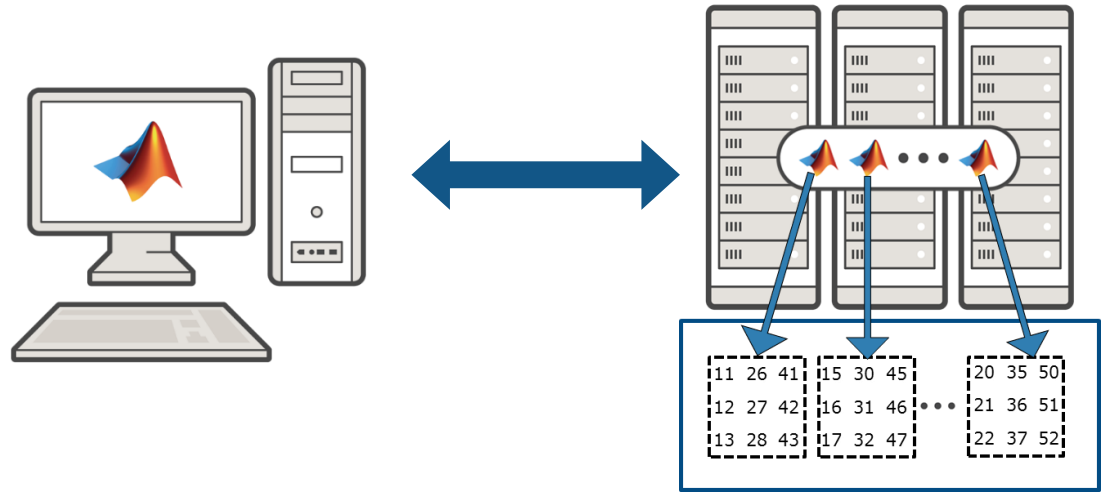
MATLAB Distributed Computing Server

```
parpool(<cluster>)
A = distributed(ds);
x = A\b;
xg = gather(x);
% For large A,b
% A,b are distributed arrays
```

Develop applications once, change run environment by changing the profile

When to Use Distributed Arrays

- Data Characteristics
  - Data must be fit in collective memory across multiple machines
- Compute Platform
  - Prototype with subset of data on desktop
  - Run on a cluster or cloud
- Analysis Characteristics
  - Consists of functions supported by distributed arrays (linear algebra, filtering, signal processing, etc.)



Agenda

- New tall arrays for easily working with big data
- Distributed arrays for big matrix operations
- Other MATLAB tools for big data
- Summary

Big Data capabilities in MATLAB

1 ACCESS DATA

More data and collections of files than fit in memory

Datastores

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

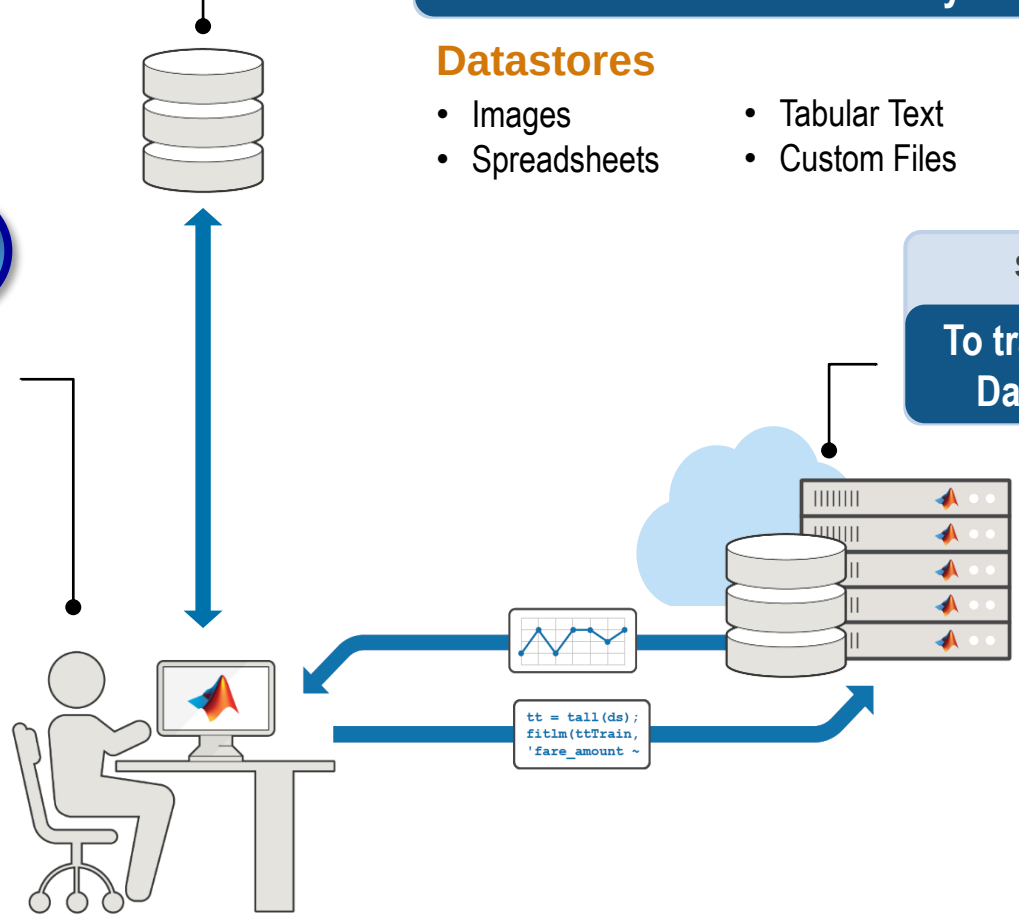
2 PROCESS ON THE DESKTOP

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

Tall Arrays

- Math
- Statistics
- Visualization
- Machine Learning

MapReduce



3 SCALE PROBLEM SIZE

To traditional clusters and Big Data systems like Hadoop

Tall Arrays

- Math, Stats, Machine Learning on Spark

Distributed Arrays

- Matrix Math on Compute Clusters

SPMD

MapReduce

MATLAB API for Spark

Access Big Data

- Datastore: *easily access large sets of data*
  - Object designed for accessing data
  - Preview data structure and format
  - Variety of types for different data sources:
    - TabularText Datastore
    - Spreadsheet Datastore
    - Database Datastore
    - KeyValue Datastore
    - File Datastore
    - Image Datastore
  - Incrementally read portions of the data
  - Use with Parallel Computing tools

```
ds = datastore('airlinesmall.csv', 'TreatAsMissing', 'NA');
ds.SelectedVariableNames = {'ActualElapsedTime', 'Distance', ...
                            'DepDelay', 'ArrDelay'};
```

```
preview(ds)
```

```
ans =

    ActualElapsedTime    Distance    DepDelay    ArrDelay
    _________________    ________    ________    ________

     53                  308         12           8
     63                  296          1           8
     83                  480         20          21
     59                  296         12          13
     77                  373         -1           4
     61                  308         63          59
     84                  447         -2           3
    155                  954         -1          11
```

```
sums = [];
counts = [];
while hasdata(ds)
    T = read(ds);

    sums(end+1) = sum(T.ArrDelay);
    counts(end+1) = length(T.ArrDelay);
end
```

When to Use `datastore`

- Data Characteristics
  - Data stored in files supported by datastore
- Compute Platform
  - Desktop or cluster
- Analysis Characteristics
  - Supports Load, Analyze, Discard workflows
  - Incrementally read chunks of data, process within a **`while`** loop

```
ds = datastore('airlinesmall.csv', 'TreatAsMissing', 'NA');
ds.SelectedVariableNames = {'ActualElapsedTime', 'Distance', ...
                            'DepDelay', 'ArrDelay'};
```

```
preview(ds)
```

```
ans =

    ActualElapsedTime    Distance    DepDelay    ArrDelay
    _________________    ________    ________    ________

     53                  308         12           8
     63                  296          1           8
     83                  480         20          21
     59                  296         12          13
     77                  373         -1           4
     61                  308         63          59
     84                  447         -2           3
    155                  954         -1          11
```

```
sums = [];
counts = [];
while hasdata(ds)
    T = read(ds);

    sums(end+1) = sum(T.ArrDelay);
    counts(end+1) = length(T.ArrDelay);
end
```

Low Level Big Data Programming

Programming frameworks for applying any MATLAB functionality to big data

SPMD: *An MPI-like framework for low-level in-memory parallel processing on traditional clusters*

MapReduce: *Write MapReduce programs in MATLAB and run them anywhere you run MATLAB, including on Hadoop*

MATLAB API for Spark: *Use the Spark RDD API from within MATLAB to create standalone Spark applications*

```
function maxDelayMap(data, info, intermKVStore)

    partMax = max(data.ArrDelay);
    add(intermKVStore,'PartialMaxDelay',partMax)
```

```
function maxDelayReduce(intermKey, intermVal, outKVstore)
    maxVal = -inf;

    while hasnext(intermVal)
        maxVal = max(getnext(intermVal),maxVal);
    end

    add(outKVStore,'MaxArrivalDelay',maxVal)
```

```
maxDelay = mapreduce(ds, @maxDelayMap, @maxDelayReduce);
```

When to Use Low Level Programming

- Data Characteristics
  - Data stored in files supported by datastore
- Compute Platform
  - Desktop or cluster depending on the technique
- Analysis Characteristics
  - Analysis not supported by tall or distributed arrays
  - Algorithms can be re-architected into low level API
  - Prior experience with low level programming recommended

Agenda

- New tall arrays for easily working with big data
- Distributed arrays for big matrix operations
- Other MATLAB tools for big data
- Summary

Summary

- MATLAB makes it easy, convenient, and scalable to work with big data
  - **Access** any kind of big data from any file system using **datastores**
  - Use **tall arrays** and **distributed arrays** to **process and analyze** that data on your desktop, clusters, or on Spark + Hadoop
  - If needed, low level programming is available for use beyond what tall and distributed arrays support

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