MATLAB TOUR 2017

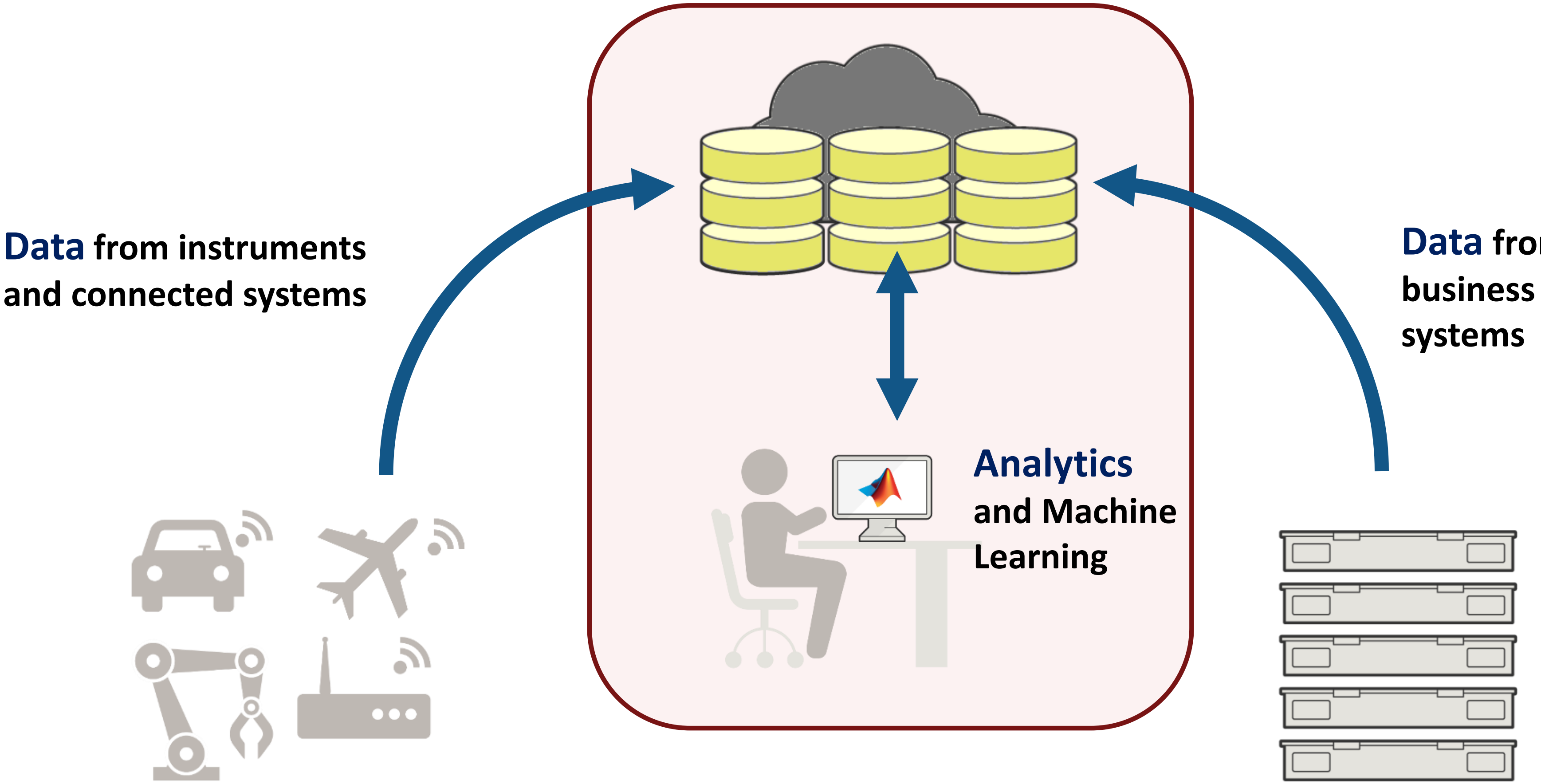
Big Data con MATLAB

Lucas García

Agenda

- Introduction
- Remote Arrays in MATLAB
- Tall Arrays for Big Data
- Scaling up
- Summary

Architecture of an analytics system



How big is big?

What does “Big Data” even mean?

“Any collection of data sets so large and complex that it becomes difficult to process using … traditional data processing applications.”

(Wikipedia)

“Any collection of data sets so large that it becomes difficult to process using traditional MATLAB functions, which assume all of the data is in memory.”

(MATLAB)

How big is big?

In 1085 William 1st commissioned a survey of England

- ~2 million words and figures collected over two years
- too big to handle in one piece
- collected and summarized in regional pieces
- used to generate revenue (tax), but most of the data then sat unused

The Large Hadron Collider reached peak performance on 29 June 2016

- 2076 bunches of 120 billion protons currently circulating in each direction
- ~1.6x10^{14} collisions per week, >30 petabytes of data per year
- too big to even store in one place
- used to explore interesting science, but taking researchers a long time to get through

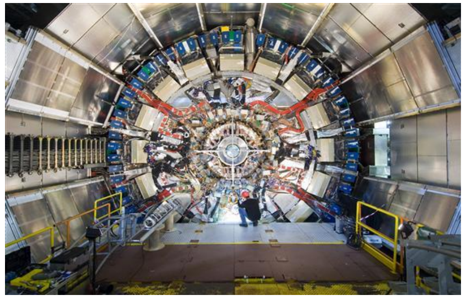
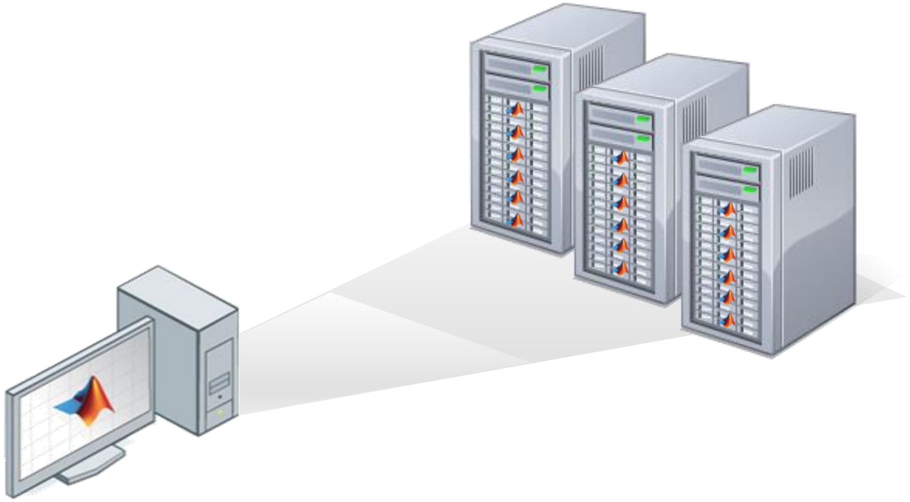


Image courtesy of CERN.
Copyright 2011 CERN.

How big is big?

Most of our data lies somewhere in between the extremes

- >10GB might be too much for one laptop / desktop (“inconveniently large”)



Big problems

So what’s the big problem?

- Standard tools 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

We want to let you:

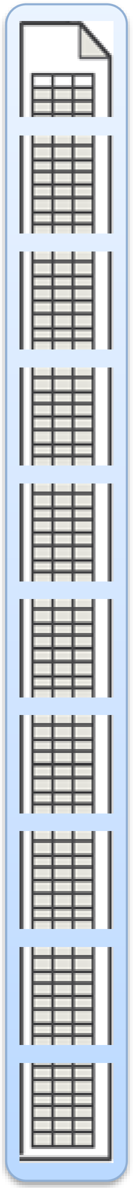
- Prototype algorithms quickly using small data
- Scale up to huge data-sets running on large clusters
- **Use the same MATLAB code for both**

New solution starting in R2016b: `tall` arrays

Quick overview (detail later!):

- Treat data in multiple files as one large table/array
- Write normal array / table code
- Behind the scenes operate on pieces

`tall array`

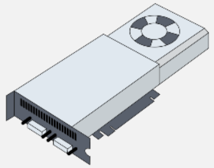
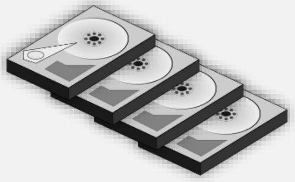


Agenda

- Introduction
- Remote Arrays in MATLAB
- Tall Arrays for Big Data
- Scaling up
- Summary

Remote arrays in MATLAB

MATLAB provides array types for data that is not in “normal” memory

distributed array (since R2006b)		Data lives in the combined memory of a cluster of computers
gpuArray (since R2010b)		Data lives in the memory of the GPU card
tall array (since R2016b)		Data lives on disk, maybe spread across many disks (distributed file-system)

Remote arrays in MATLAB

Rule: take the calculation to where the data is

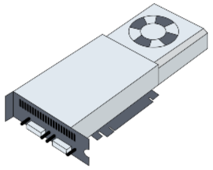
Normal array – calculation happens in main memory:

```
x = rand(...)

x_norm = (x - mean(x)) ./ std(x)
```

Remote arrays in MATLAB

Rule: take the calculation to where the data is

`gpuArray` – all calculation happens on the GPU:

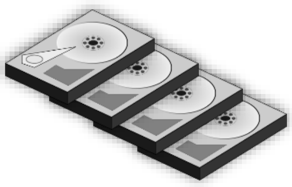


```
x = gpuArray(...)

x_norm = (x - mean(x)) ./ std(x)
```

`distributed` – calculation is spread across the cluster:

```
x = distributed(...)

x_norm = (x - mean(x)) ./ std(x)
```

`tall` – calculation is performed by stepping through files:



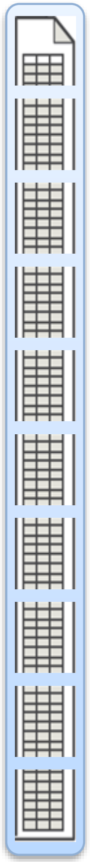
```
x = tall(...)

x_norm = (x - mean(x)) ./ std(x)
```

Agenda

- Introduction
- Remote Arrays in MATLAB
- Tall Arrays for Big Data
- Scaling up
- Summary

`tall` arrays (new R2016b)

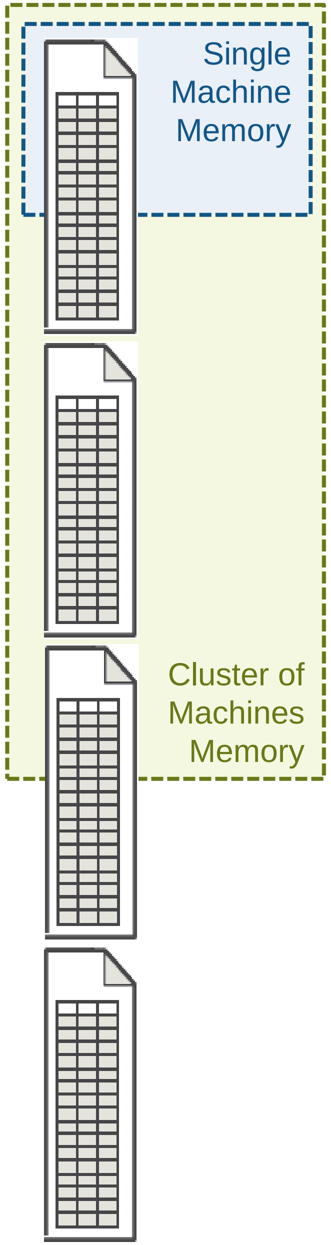
- MATLAB data-type for data that doesn’t fit into memory
- Ideal for lots of observations, few variables (hence “tall”)
- Looks like a normal MATLAB array
  - Supports numeric types, tables, datetimes, categoricals, strings, etc.
  - Basic maths, stats, indexing, etc.
  - **Statistics and Machine Learning Toolbox** support (clustering, classification, etc.), **Database Toolbox**.



`tall` arrays (new R2016b)

- Data is in one or more files
- Typically tabular data
- Files stacked vertically
- Data doesn't fit into memory

  (even cluster memory)



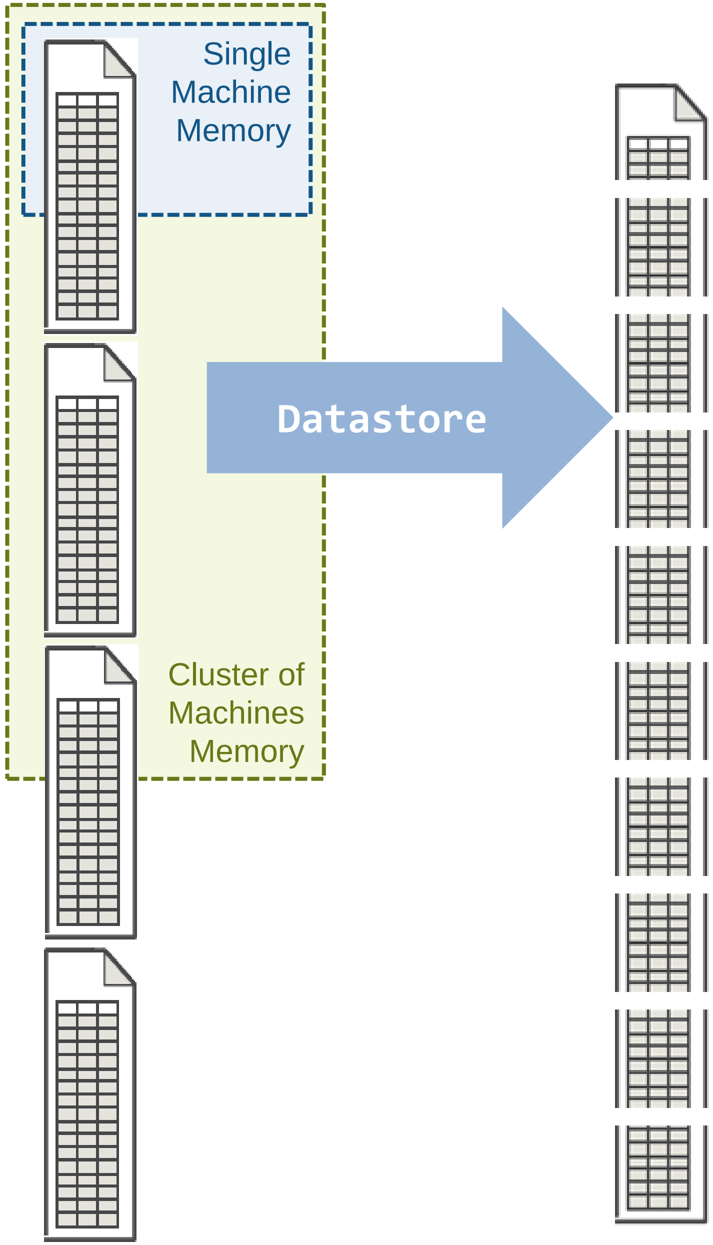
tall arrays (new R2016b)

- Use datastore to define file-list

```
ds = datastore('*.csv')
```

- Allows access to small pieces of data that fit in memory.

```
while hasdata(ds)
   piece = read(ds);
   % Process piece
end
```



tall arrays (new R2016b)

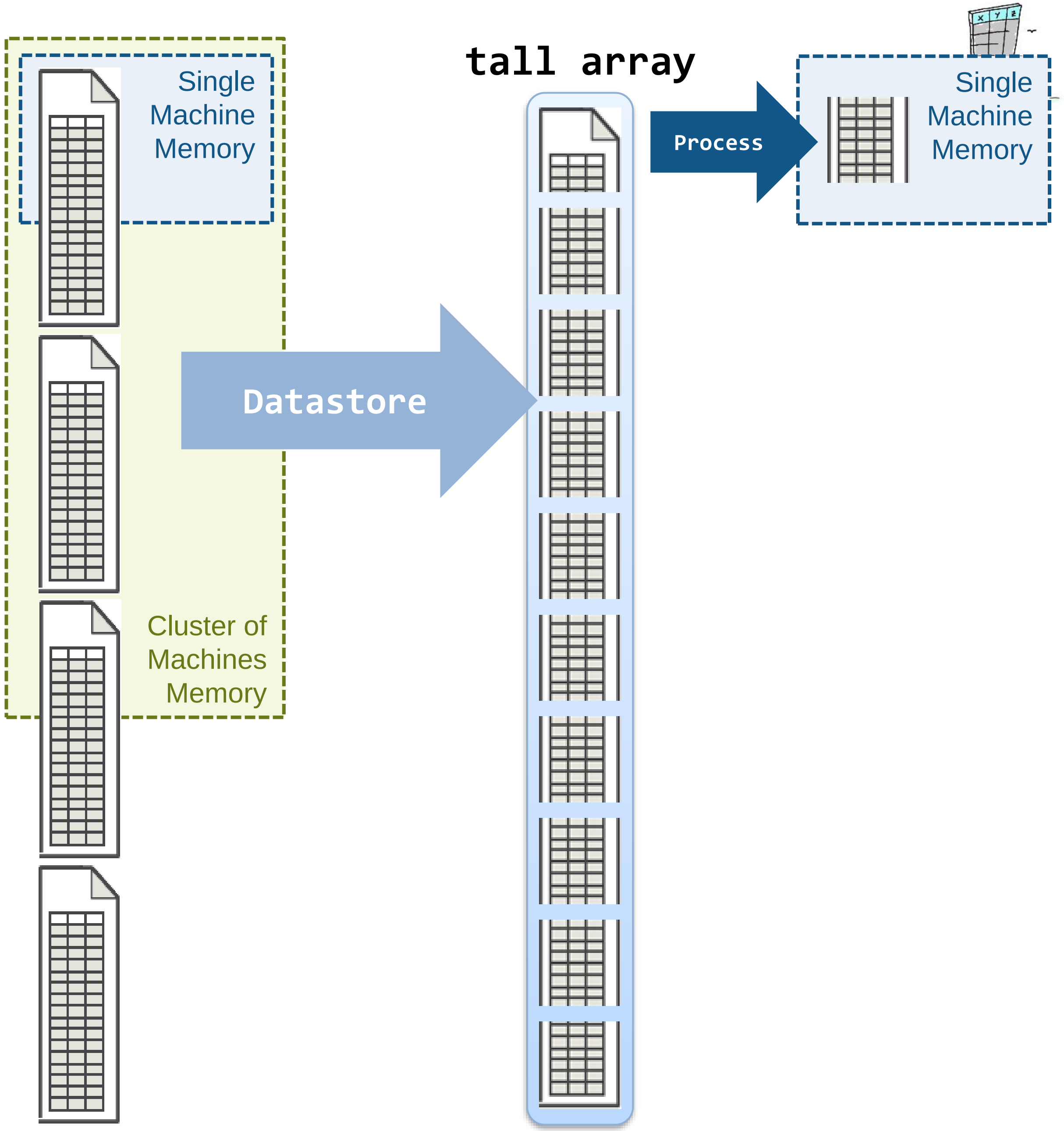
- Create tall table from datastore

```
ds = datastore('*.csv')
tt = tall(ds)
```

- Operate on whole tall table just like ordinary table

```
summary(tt)

max(tt.EndTime – tt.StartTime)
```

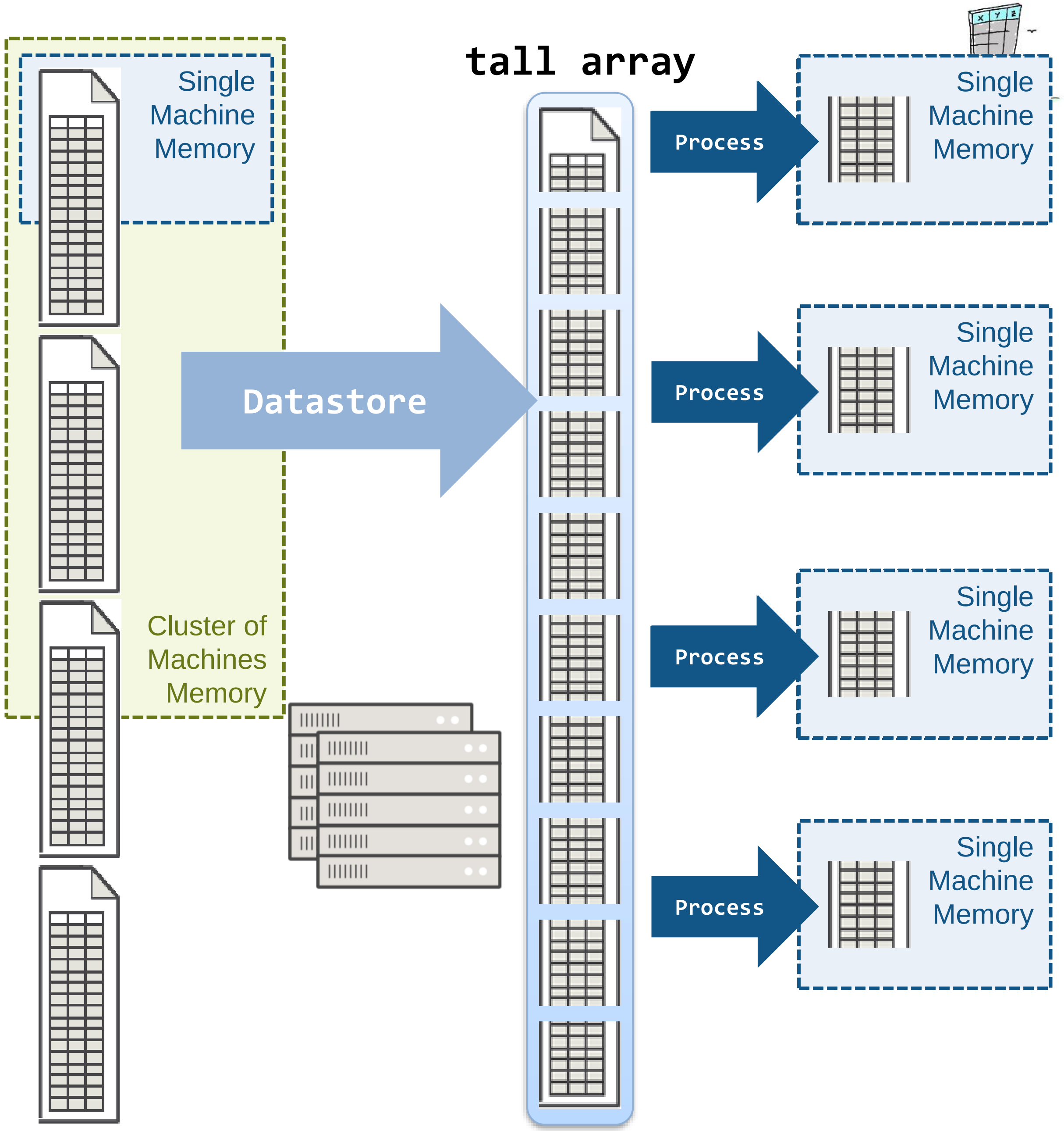
- “Chunk” processing is handled automatically



`tall` arrays (new 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**

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

    VendorID     tpep_pickup_datetime     tpep_dropoff_datetime     passenger_count    trip_distance    pickup_longitude    pic
    ________    _____________________    _____________________    _______________    _____________    ________________    __

    2           2015-01-10 02:24:04      2015-01-10 02:36:10      2                  2.19             -73.999             40.
    1           2015-01-18 21:29:35      2015-01-18 21:34:15      1                     1             -74.017             40.
    2           2015-01-23 18:23:02      2015-01-23 18:39:32      3                  2.22             -73.973             40.
    1           2015-01-01 05:29:50      2015-01-01 05:48:55      1                   3.6             -73.943             40.
    1           2015-01-18 00:06:42      2015-01-18 00:11:43      1                   0.8             -73.983             40.
    2           2015-01-29 23:56:41      2015-01-30 00:02:49      5                  0.87             -73.982             40.
    2           2015-01-05 16:58:24      2015-01-05 17:03:33      5                  0.78             -73.992             40.
    1           2015-01-23 23:49:53      2015-01-23 23:55:42      2                   1.4             -73.956              40.
```

Example: Prototyping

Create a Tall Array

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

Input data is tabular – result is a tall table

Only the first few rows are displayed

```
>> tt = tall(ds)
tt =

  M×19 tall table

    VendorID    tpep_pickup_datetime    tpep_dropoff_datetime    passenger_count    trip_distance    pickup_longitude    pic
    ________    ____________________    _____________________    _______________    _____________    ________________    ___

    2           2015-01-10 02:24:04     2015-01-10 02:36:10      2                  2.19             -73.999             40.
    1           2015-01-18 21:29:35     2015-01-18 21:34:15      1                     1             -74.017             40.
    2           2015-01-23 18:23:02     2015-01-23 18:39:32      3                  2.22             -73.973             40.
    1           2015-01-01 05:29:50     2015-01-01 05:48:55      1                   3.6             -73.943             40.
    1           2015-01-18 00:06:42     2015-01-18 00:11:43      1                   0.8             -73.983             40.
    2           2015-01-29 23          30 00:02:49              5                  0.87             -73.982             40.
    2           2015-01-05 16          05 17:03:33              5                  0.78             -73.992             40.
    1           2015-01-23 23          23 23:55:42              2                   1.4             -73.956             40.
    :           :                       :                        :                  :                :                   :
    :           :                       :                        :                  :                :                   :
```

Example: Prototyping

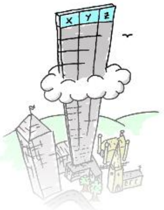
Calling Functions with a Tall Array

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 4 sec

mn =

   13.2763
```

- 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

MathWorks



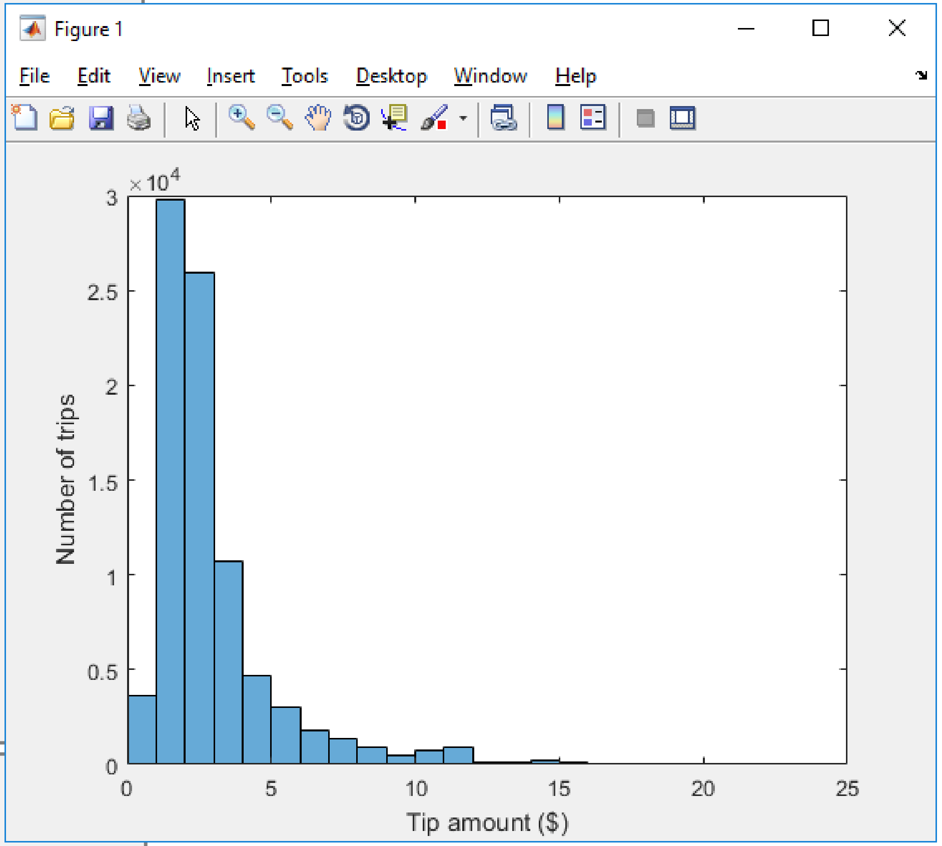
Example: Prototyping

Calling Functions with a Tall Array

```
% 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 time
    tt.trip_minutes >= 60 * 12 | ...    % unfeasibly long time
    tt.trip_distance <= 1 | ...         % really short distance
    tt.trip_distance >= 12 * 55 | ...   % unfeasibly far
    tt.speed_mph > 55 | ...             % unfeasibly fast
    tt.fare_amount < 0 | ...            % negative fares?!
    tt.fare_amount > 10000;             % unfeasibly large fares
tt(ignore, :) = [];

% Credit card payments have the most accurate tip data
keep = tt.payment_type == {'Credit card'};
tt = tt(keep,:);

% Show tip distributi
histogram( tt.tip_amou
```

Data only read once, despite 21 operations

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



Example: Prototyping

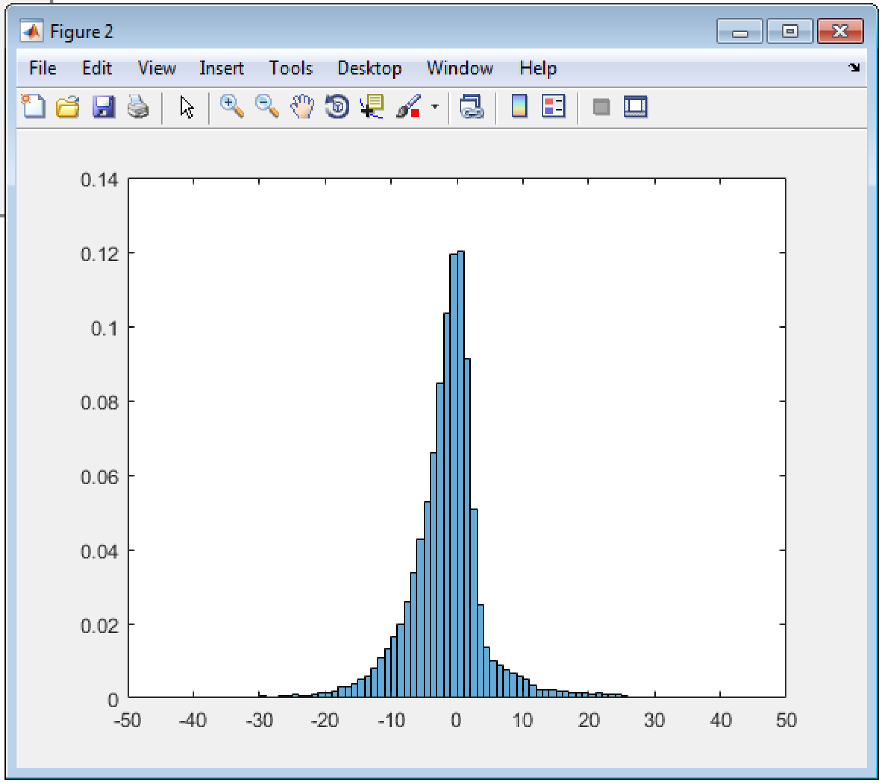
Fit predictive model

```
% 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 5 sec
Evaluation completed in 5 sec

model =

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

Estimated Coefficients:
                                  Estimate        SE          tStat       pValue 
                                 __________    __________    ________    _______

    (Intercept)                      2.3432      0.040181      58.318          0
    trip_distance                    2.5841     0.0063898      404.41          0
    hr_of_day                    -0.0012969     0.0018789    -0.69024    0.49005
    trip_minutes                    0.22098     0.0020412      108.26          0
    trip_distance:trip_minutes    -0.007857    0.00017539     -44.798          0

Number of observations: 42373, Error degrees of freedom: 42368
Root Mean Squared Error: 2.58
R-squared: 0.938,  Adjusted R-Squared 0.938
F-statistic vs. constant model: 1.59e+05, p-value = 0
```

Example: Prototyping

Predict and validate model

```
% Predict and validate
yPred = predict(model,ttValidation);
residuals = yPred - ttValidation.fare_amount;
figure
histogram(residuals,'Normalization','pdf','BinLimits',[-50 50])

Evaluating tall expression using the Local MATLAB Session:
- Pass 1 of 2: Completed in 5 sec
- Pass 2 of 2: Completed in 4 sec
Evaluation completed in 10 sec
```



Agenda

- Introduction
- Remote Arrays in MATLAB
- Tall Arrays for Big Data
- Scaling up
- Summary

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);
```

“small data” processing

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

% 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 time
    tt.trip_minutes >= 60 * 12 | ...    % unfeasibly long time
    tt.trip_distance <= 1 | ...         % really short distance
    tt.trip_distance >= 12 * 55 | ...   % unfeasibly far
    tt.speed_mph > 55 | ...             % unfeasibly fast
    tt.fare_amount < 0 | ...            % negative fares?!
    tt.fare_amount > 10000;             % unfeasibly large fares
tt(ignore, :) = [];
```

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

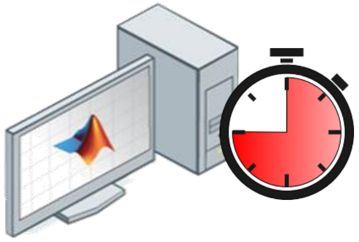
Big Data processing

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

% 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 time
    tt.trip_minutes >= 60 * 12 | ...    % unfeasibly long time
    tt.trip_distance <= 1 | ...         % really short distance
    tt.trip_distance >= 12 * 55 | ...   % unfeasibly far
    tt.speed_mph > 55 | ...             % unfeasibly fast
    tt.fare_amount < 0 | ...            % negative fares?!
    tt.fare_amount > 10000;             % unfeasibly large fares
tt(ignore, :) = [];
```

Scaling up

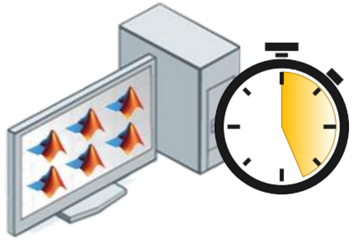
If you just have **MATLAB**:

- Run through each ‘chunk’ of data one by one



If you also have **Parallel Computing Toolbox**:

- Use all local cores to process several ‘chunks’ at once



If you also have a cluster with **MATLAB Distributed Computing Server (MDCS)**:

- Use the whole cluster to process many ‘chunks’ at once

Scaling up

Working with clusters from MATLAB desktop:

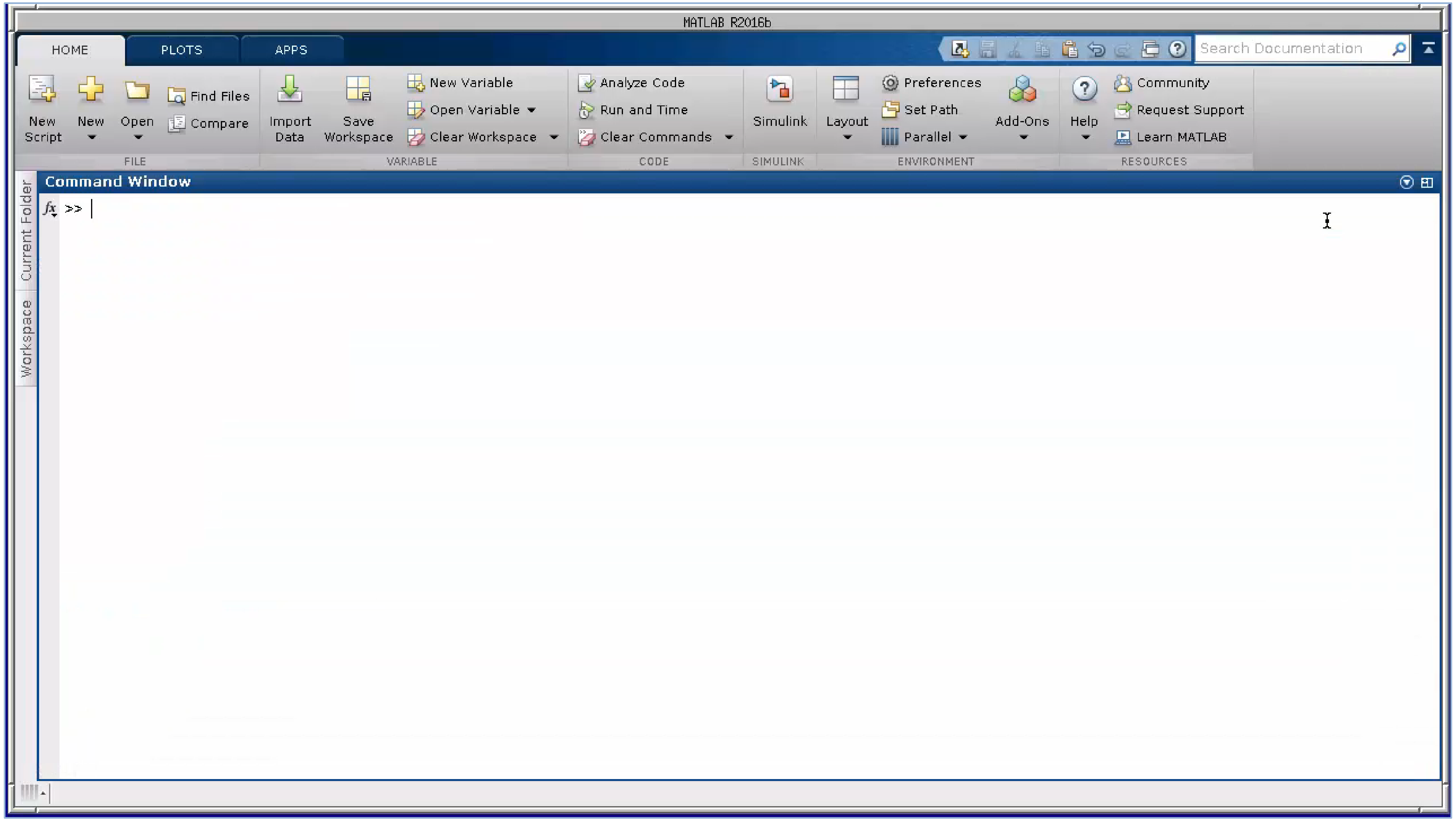
- General purpose MATLAB cluster
  - Can co-exist with other MATLAB workloads (`parfor`, `parfeval`, `spmd`, jobs and tasks, `distributed` arrays, …)
  - Uses local memory and file caches on workers for efficiency

- Spark-enabled Hadoop clusters
  - Data in HDFS
  - Calculation is scheduled to be near data
  - Uses Spark's built-in memory and disk caching

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



Agenda

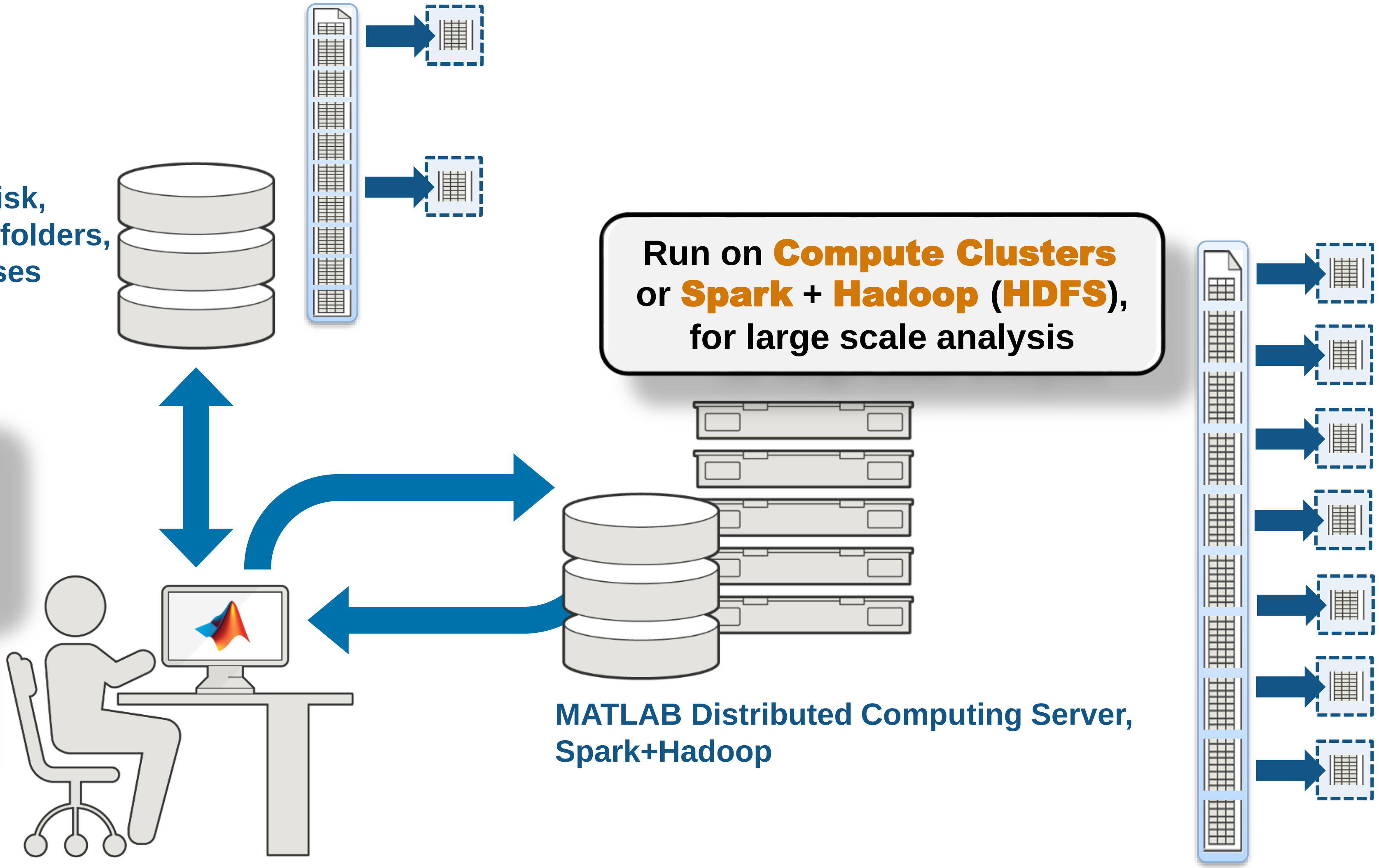
- Introduction
- Remote Arrays in MATLAB
- Tall Arrays for Big Data
- Scaling up
- Summary

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

Local disk,
Shared folders,
Databases



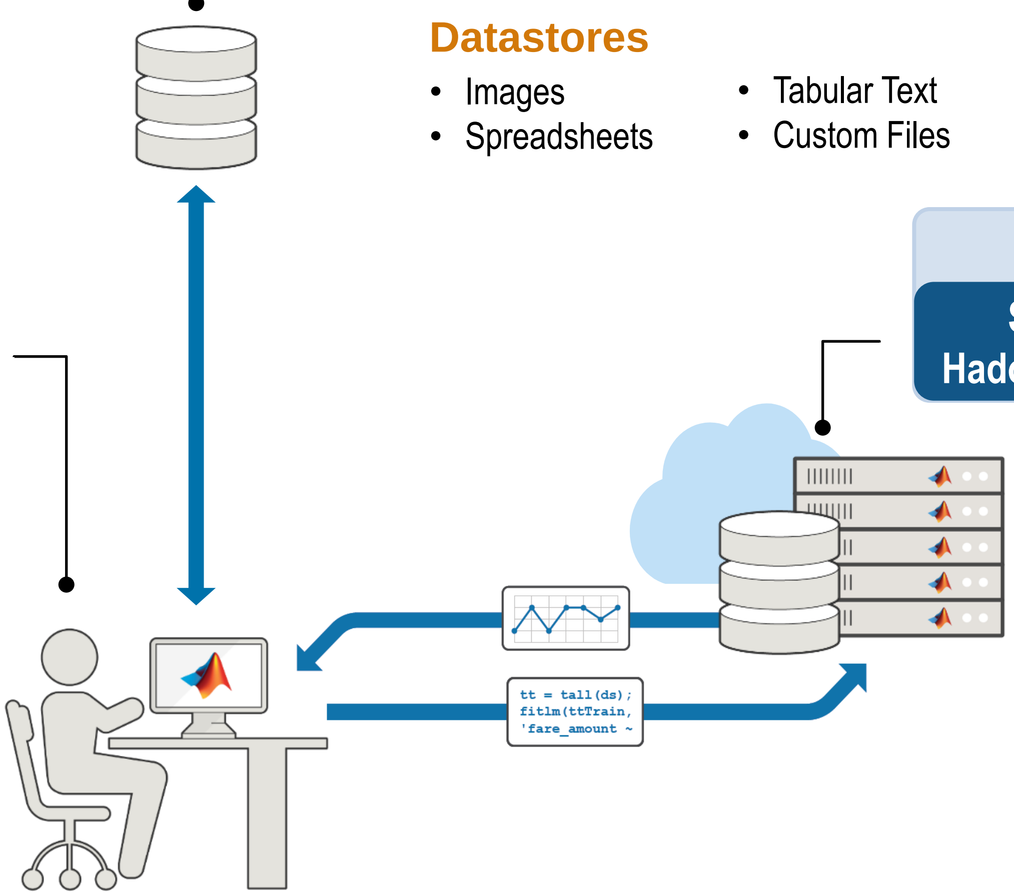
Big Data capabilities in MATLAB

ACCESS

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

Datastores

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



PROCESS AND ANALYZE

Purpose-built capabilities for domain experts to work with big data locally

Tall Arrays

- Math
- Statistics
- Visualization
- Machine Learning

GPU Arrays

- Matrix Math
- Image Processing

Deep Learning

- Image Classification

SCALE

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

Tall Arrays

- Math, Stats, Machine Learning on Spark

Distributed Arrays

- Matrix Math on Compute Clusters

MDCS for EC2

- Cloud-based Compute Cluster

MapReduce

MATLAB API for Spark

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.

Questions