Tackling Big Data with MATLAB

Adam Filion
Application Engineer
MathWorks, Inc.
Challenges of Big Data

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

- Getting started
- Rapid data exploration
- Development of scalable algorithms
- Ease of deployment
MATLAB and Memory
Best Practices for Memory Usage

- Use 64-bit MATLAB whenever possible
- Use the appropriate data storage
  - Use only the precision you need
  - Sparse Matrices
  - Categorical Arrays
  - Be aware of overhead of cells and structures
- Minimize Data Copies
  - Lazy copy
  - Nested functions
  - In place operations
  - If using objects, consider handle classes

```matlab
define primaryFcn
  x = 1;
  y = x;
end

define nestedFcn
  x = x + 1;
end
de```

function primaryFcn
```matlab
x = 1;
y = x;
nestedFcn

function nestedFcn
  x = x + 1;
end
end```
Big Data Capabilities in MATLAB

Memory and Data Access
- 64-bit processors
- Memory Mapped Variables
- Disk Variables
- Databases
  - Datastores

Programming Constructs
- Streaming
- Block Processing
  - Parallel-for loops
- GPU Arrays
- SPMD and Distributed Arrays
- MapReduce

Platforms
- Desktop (Multicore, GPU)
- Clusters
- Cloud Computing (MDCS on EC2)
  - Hadoop
Considerations for Choosing an Approach

- Data characteristics
  - Size, type and location of your data

- Compute platform
  - Single desktop machine or cluster

- Analysis Characteristics
  - Embarrassingly Parallel
    - Analyze sub-segments of data and aggregate results
  - Operate on entire dataset
Techniques for Big Data in MATLAB

Load, Analyze, Discard
parfor, datastore

MapReduce

Distributed Memory
SPMD and distributed arrays

Complexity

Embarrassingly Parallel

Non-Partitionable

out-of-memory
in-memory
Techniques for Big Data in MATLAB

- Embarrassingly Parallel
- Load, Analyze, Discard
  - `parfor`, `datastore`
- MapReduce
- Distributed Memory
  - `SPMD` and distributed arrays
- Complexity
Parallel Computing with MATLAB
Speed up Using Simultaneous Workers

MATLAB Desktop (Client)

Time

Worker
Worker
Worker
Demo: Determining Land Use
Using Parallel for-loops (parfor)

- **Data**
  - Arial images of agriculture land
  - 24 TIF files

- **Analysis**
  - Find and measure irrigation fields
  - Determine which irrigation circles are in use (by color)
  - Calculate area under irrigation
When to Use `parfor`

- **Data Characteristics**
  - Can be of any format (i.e. text, images) as long as it can be broken into pieces
  - The data for each iteration must fit in memory

- **Compute Platform**
  - Desktop (Parallel Computing Toolbox)
  - Cluster (MATLAB Distributed Computing Server)

- **Analysis Characteristics**
  - Each iteration of your loop must be independent
Techniques for Big Data in MATLAB

- Load, Analyze, Discard
  - `parfor`, `datastore`

- MapReduce

- Distributed Memory
  - SPMD and distributed arrays

Complexity

- Embarrassingly Parallel
- Non-Partitionable
Access Big Data

datastore

- Easily specify data set
  - Single text file or collection of text files
  - Data stored on HDFS

- Preview data structure and format

- Select data to import using column names

- Incrementally read subsets of the data

```matlab
airdata = datastore('*.csv');
airdata.SelectedVariables = {'Distance', 'ArrDelay'};
data = read(airdata);
```
Demo: Vehicle Registry Analysis

Using a DataStore

- Data
  - Massachusetts Vehicle Registration Data from 2008-2011
  - 16M records, 45 fields

- Analysis
  - Examine hybrid adoptions
  - Calculate % of hybrids registered by quarter
  - Fit growth to predict further adoption
When to Use datastore

- **Data Characteristics**
  - Text data in files or stored in the Hadoop Distributed File System (HDFS)

- **Compute Platform**
  - Desktop

- **Analysis Characteristics**
  - Supports Load, Analyze, Discard workflows
  - Incrementally read chunks of data, process within a `while` loop
Reading in Part of a Dataset from Files

- Text file, ASCII file
  - datastore

- MAT file
  - Load and save part of a variable using the matfile

- Binary file
  - Read and write directly to/from file using memmapfile
  - Maps address space to file

- Databases
  - ODBC and JDBC-compliant (e.g. Oracle, MySQL, Microsoft SQL Server)
Techniques for Big Data in MATLAB

- Load, Analyze, Discard
  - `parfor`, `datastore`
- MapReduce
- Distributed Memory
  - `SPMD` and `distributed arrays`

- Embarrassingly Parallel
- Complexity
- Non-Partitionable

- out-of-memory
- in-memory
MapReduce Programming Model

- mapreduce
- datastore
- mapreduce
Demo: Vehicle Registry Analysis
*Using MapReduce*

- **Data**
  - Massachusetts Vehicle Registration Data from 2008-2011
  - 16M records, 45 fields

- **Analysis**
  - Examine hybrid adoptions
  - Calculate % of hybrids registered by quarter
# MapReduce

## Data Store

<table>
<thead>
<tr>
<th>Veh_typ</th>
<th>Q3_08</th>
<th>Q4_08</th>
<th>Q1_09</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SUV</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Car</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Car</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Car</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Car</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Car</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SUV</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Car</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SUV</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Car</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Car</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

## Map

- **Key: Q3_08**
  - Hybrid: 0 1 1 0
- **Key: Q4_08**
  - Hybrid: 0 1 1 1
- **Key: Q1_09**
  - Hybrid: 0 1 1 0

## Reduce

<table>
<thead>
<tr>
<th>Key</th>
<th>% Hybrid (Value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q3_08</td>
<td>40%</td>
</tr>
<tr>
<td>Q4_08</td>
<td>67%</td>
</tr>
<tr>
<td>Q1_09</td>
<td>71%</td>
</tr>
</tbody>
</table>
The Big Data Platform

Datastore

HDFS

Node

Data

Map

Reduce

Node

Data

Map

Reduce

Node

Data

Map

Reduce

Node

Data

Map

Reduce
Explore and Analyze Data on a Cluster

MATLAB MapReduce Code

Datastore

MATLAB Distributed Computing Server

Node Data Map Reduce

Node Data Map Reduce

Node Data Map Reduce
Deployed `mapreduce`
When to Use `mapreduce`

- **Data Characteristics**
  - Text data in files or stored in the Hadoop Distributed File System (HDFS)
  - Dataset will not fit into memory

- **Compute Platform**
  - Desktop
  - Traditional HPC cluster
  - Hadoop cluster (within Hadoop MapReduce framework)

- **Analysis Characteristics**
  - Must be able to be partitioned into two phases
    1. Map: filter or process sub-segments of data
    2. Reduce: aggregate interim results and calculate final answer
Techniques for Big Data in MATLAB

- Load, Analyze, Discard: parfor, datastore
- MapReduce
- Distributed Memory: SPMD and distributed arrays

Complexity

- Embarrassingly Parallel
- Non-Partitionable

out-of-memory
in-memory
Parallel Computing – Distributed Memory
spmd blocks

spmd
  % single program across workers
end

- Mix parallel and serial code in the same function
- Run on a pool of MATLAB resources
- Single Program runs simultaneously across workers
- Multiple Data spread across multiple workers
Example: Airline Delay Analysis

- **Data**
  - BTS/RITA Airline On-Time Statistics
  - 123.5M records, 29 fields

- **Analysis**
  - Calculate delay patterns
  - Visualize summaries
  - Estimate & evaluate predictive models
When to Use Distributed Memory

- **Data Characteristics**
  - Data must be fit in collective memory across machines

- **Compute Platform**
  - Prototype (subset of data) on desktop
  - Run on a cluster or cloud

- **Analysis Characteristics**
  - Consists of:
    - Parts that can be run on data in memory (*spmd*)
    - Supported functions for distributed arrays
Big Data on the Desktop

- Expand workspace
  - 64 bit processor support – increased in-memory data set handling

- Access portions of data too big to fit into memory
  - Memory mapped variables – huge binary file
  - Datastore – huge text file or collections of text files
  - Database – query portion of a big database table

- Variety of programming constructs
  - System Objects – analyze streaming data
  - MapReduce – process text files that won’t fit into memory

- Increase analysis speed
  - Parallel for-loops with multicore/multi-process machines
  - GPU Arrays
Further Scaling Big Data Capacity

- MATLAB has a range of programming constructs for clusters
- General compute clusters
  - Parallel for-loops: embarrassingly parallel algorithms
  - SPMD and distributed arrays: distributed memory
  - MapReduce: process big text files
- Hadoop clusters
  - MapReduce: analyze data stored in the HDFS

Use these constructs on the desktop to develop your algorithms

Migrate to a cluster without algorithm changes
Learn More

- MATLAB Documentation
  - Strategies for Efficient Use of Memory
  - Resolving "Out of Memory" Errors

- Big Data with MATLAB

- MATLAB MapReduce and Hadoop