Building a Big Engineering Data Analytics System using MATLAB

Arvind Hosagrahara
Principal Technical Consultant
Arvind.Hosagrahara@mathworks.com
Tel: 310-819-3960
Trend: Data Economy

“Information is the oil of the 21st century, and analytics is the combustion engine”

Peter Sondergaard, Gartner Research

Access and Explore Data
Preprocess Data
Develop Predictive Models
Integrate Analytics with Systems
Problem Statement

- Can MATLAB® scale up and meet increasingly demanding fleet test data analytics requirements?

- Is it possible to build data analytics algorithms in a flexible, scalable manner and yet satisfy production requirements?
The Fleet Data Analysis System

http://goo.gl/EeE5JE
A quick recap of our connected cars

[Diagram with components: OBD2, Bluetooth, 4G LTE, HTTP, Amazon EC2, LAMR Stack (RAILS, ubuntu), Deployed MATLAB, Hadoop Ecosystem (MapReduce, hadoop, HDFS), MATLAB Desktop]

Apache, the Apache feather logo, Hadoop are registered trademarks or trademarks of the Apache Software Foundation.
Fleet Data Analysis System (Test Bed Summary)

- 8-25 Mb per day
- Non-telemetric data files
- Up to 25 operators
- 660Mb of data and counting
Insights (Engine Fuel Consumption and Efficiency)
Insights (8 mile traffic)

- Traffic Patterns (the case for roundabouts)

  - 0.0351 Gal/car at the intersection
  - 12 cars a minute on the average
  - A saving of 121.3 gallons of gasoline per day if the traffic lights were replaced with a round-about.
  - A rough saving of 4.5 million pounds of CO2 per year.
What has changed?

What has improved?

What is new?
Infrastructure Changes

- Enrichment of incoming streams

- Non-telemetric data
- Drag/drop addition and management of data
- MATLAB with visualization tools
- Direct Connectivity with MATLAB
- Caching of data for performance
- Upsized compute resources
- Load balancers
- Simulation sources
  - MATLAB
  - Simulink
Location and Methodology changes

- Web enabled dashboards
- Privacy and Security
- Enabling relational query
  - Cloudera Impala

- **Reference architectures** offer solutions with an emphasis on selecting the right tool for the task resulting in more efficient workflows

![Database Toolbox](image)

![TIBCO Spotfire](image)

![BigQuery](image)
Insights from the field
“By the way, this is pretty neat data to look at in MATLAB. Even without the GPS, I can pinpoint (1) the moment I nearly clobbered a deer this morning and (2) the merging onto and off of route 9…”
Performance

% Fetch the data

% Read the data into MATLAB
trackData = xlsread(dataFile);

% Isolate area of interest
idx = timeVec > 1403714100426 & timeVec < 1403717061373;

% Gather statistics
topSpeed = max(trackData(idx,7)); % in kmph.

% topSpeed =
% 167.4000

% Compute fuel usage
ccMin = trackData(:,17);
fuelUsedInCC = trapz(ccMin(idx))/(60);
fuelUsedInUSGal = 0.000264172052*fuelUsedInCC;
fuelCostInUSD = fuelUsedInUSGal*3.91;

% fuelCostInUSD =
% 6.1048
Discovering problems in veracity of collected data

- Software modifications to give higher peak ft-lbs of torque
- Increased boost pressure and optimized ignition timing
- Aftermarket software was conditioning the data to ensure proper operation
Operationalization of Analytics
Product Changes (1) - Big Data Capabilities in MATLAB

Memory and Data Access
- 64-bit processors
- Memory Mapped Variables
- Disk Variables
- Databases
- **Datastores** \( \text{R2014b} \)

Programming Constructs
- Streaming
- Block Processing
- Parallel-for loops
- GPU Arrays
- SPMD and Distributed Arrays
- **MapReduce** \( \text{R2014b} \)

Platforms
- Desktop (Multicore, GPU)
- Clusters
- Cloud Computing (MDCS on EC2)
- **Hadoop** \( \text{R2014b} \)
**Product Changes (2) – Datastores and MapReduce**

**Datastore**
- Incremental data processing feature for reading collections of data containing *tabulartext* or *keyvalue* pairs.
- Enables access to relational databases using Database Toolbox™

**MATLAB® MapReduce**
- Allows analysis of out-of-memory data.
- Deployment of mapreduce algorithms to:
  - Serial Mapreduce using local workers
  - Parallel Computing Toolbox™
  - MATLAB® Distributed Computing Server™
  - Hadoop® using the MATLAB Compiler
Out of memory processing using Datastores

Visualize Torque Speed Data from Morning Commute

This script will visualize the torque speed scatter using the datastore.

```matlab
% Auth/Revision: Arvind Busagahara
% Copyright 2014 The MathWorks Consulting Group
% $Id$

% Load our logged data
fleetDS = datastore('tracking.csv');

% Select columns of interest and chunk the data
fleetDS.SelectedVariableNames = {'Torque_Rpm','EngineRPM_rpm','FuelFlowRate_minute_gal_min'};
fleetDS.RowsPerRead = 1000;

% Preview the data to inspect for correctness
preview(fleetDS);

% Read the data and prep for plotting
plotData = read(fleetDS);
plotArray = table2array(plotData);
plotArray = table2array(plotData(:,[1,2]));

% Visualize the data
scatter(plotArray(:,1),plotArray(:,2),cellfun(@(str2num,plotArray(:,1)),',','');
xlabel('Speed (rpm)');
ylabel('Torque (Nm)');
title('Torque-Speed Distribution');
grid on;
```
Studying ride quality using MapReduce

function SimpleMapReduceDemo(varargin)

% Auth/Revision: Arvind Honagharana
% Copyright 2014 The MathWorks Consulting Group
% $Id$

% Load our logged data
fleetDS = datastore('tracklog-2014-Apr-11-10-17-32.csv', 'FileReadOptions', 'UseAddressReadMethods', true);
fleetDS = readtable(fleetDS, 'ReadLimit', 1000);

% Select columns of interest and chunk the data
fleetDS.SelectedVariableNames = {'0_x', '0_y', '0_z'};

% Run the MapReduce on serial MATLAB
rideDS = mapreduce(fleetDS, @RideMapFun, @RideReduceFun, 'OutputVariable', 'rideTable', 'ReducerOutputVariable', 'rideLenValue', 'OutputAddressReadMethod', 'SaveAddressFormat', 'struct');
end

Trial>> mapreduce(parpool)
Starting parallel pool (parpool) using the 'local' profile:
Connected to the parallel pool (number of workers: 8).
Trial>> SimpleMapReduceDemo
Parallel mapreduce execution on the local cluster:
Map Stage 0% complete
Reduce Stage 0% complete
Reduce Stage 100% complete

rideTable = struct('ride LenValue', 12345);

% Analyzing Fleet Test Data Using MATLAB

4:00-5:00 PM
Seth DeLand, MathWorks

Mapper Function

function RideMapFun(data, _, internKVStore)
% Assemble the ride data and compute ride metric based on 8 sensors
rideData = [data.0_x_data, data.0_y_data, data.0_z_data];
rideLenValue = [sum(rideData, 2), size(rideData, 1)];

% Store the key value for the reducer
internKVStore('rides', rideLenValue);
end

Reducer Function

function [outKVStore, outKVLen] = RideReduceFun(rideLenIter, outKVStore)

% Return the key value for the reducer

outKVStore = struct('Key', 'rideLenIter', 'Value', rideLenIter);
end

![](image1)

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Ride G Sensor Mean'</td>
<td>[5.7030]</td>
</tr>
</tbody>
</table>

![](image2)

![](image3)

MathWorks
Conclusion

- Single stack, open and extensible analytics toolset that plays well with other technologies

- New features support clean workflows for use in production systems and with Big Data

- A capacity for complexity