



Different meaning for everybody



Why MATLAB ?



Language Of Technical Computing

**Object Oriented Programming
Unit Testing**

Software Platform Development

Data Mining

Business
Intelligence

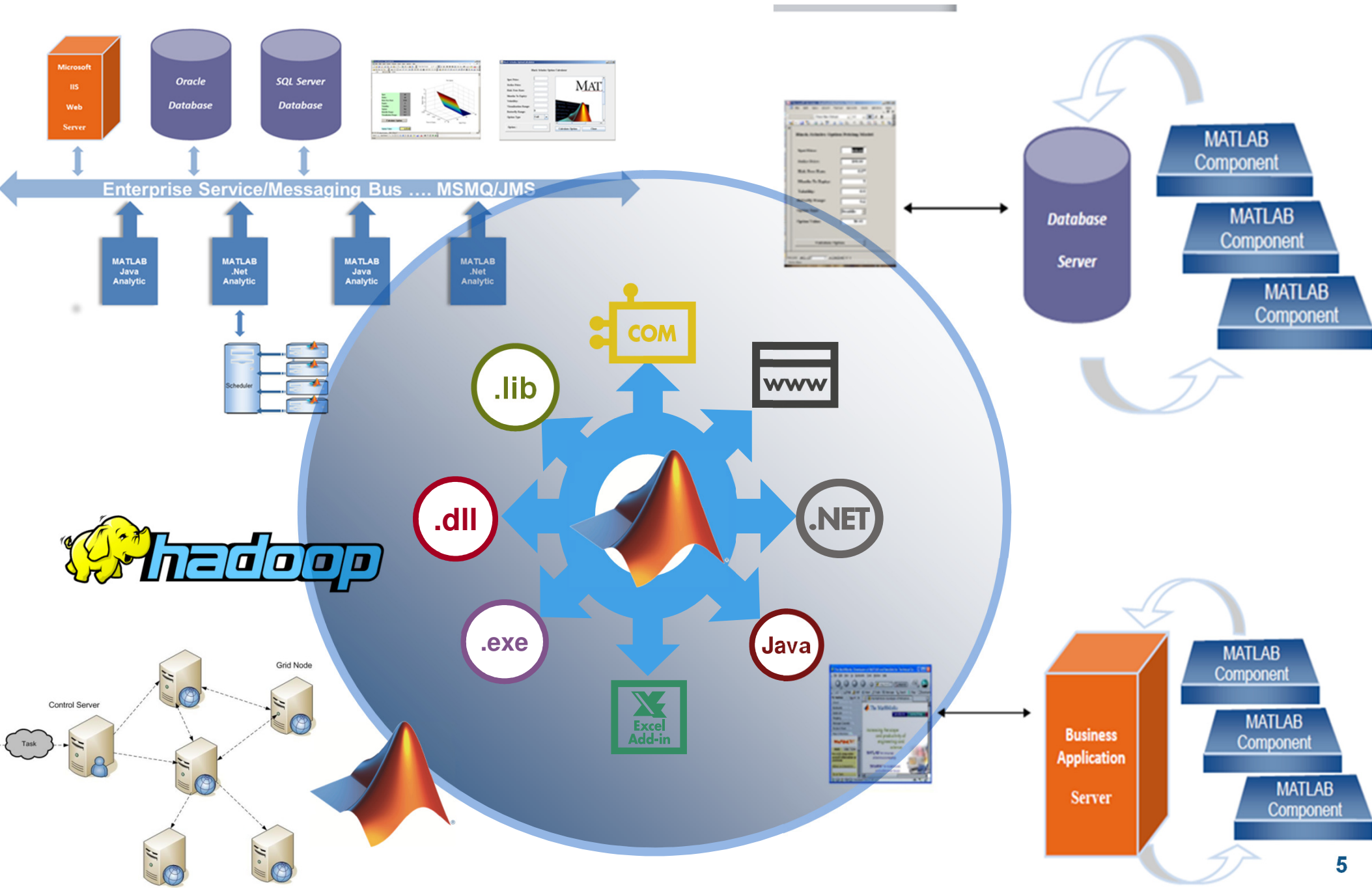
Signal
Processing

Quantitative
Finance

Image
Video

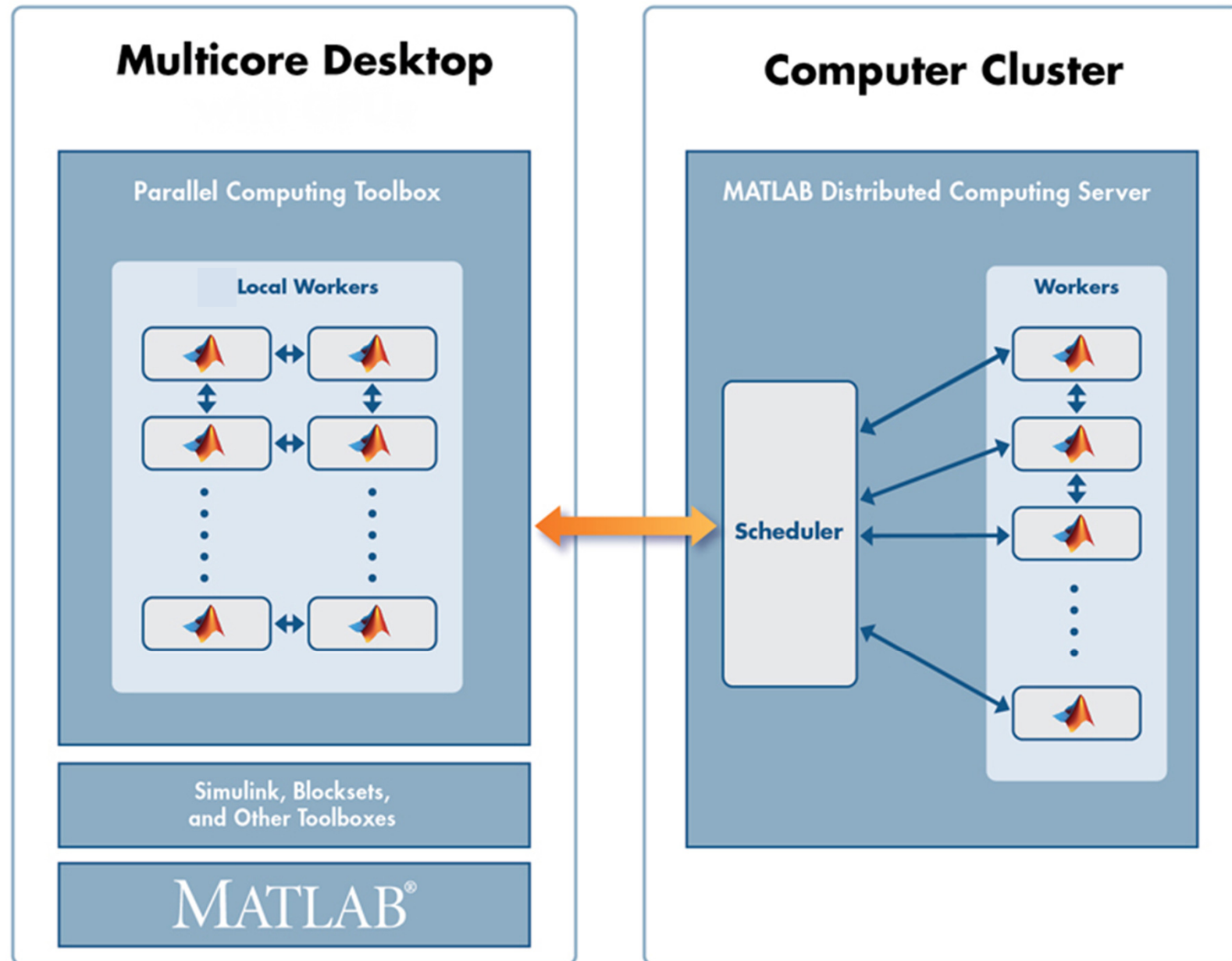
Data
Acquisition

Compiling MATLAB to go everywhere

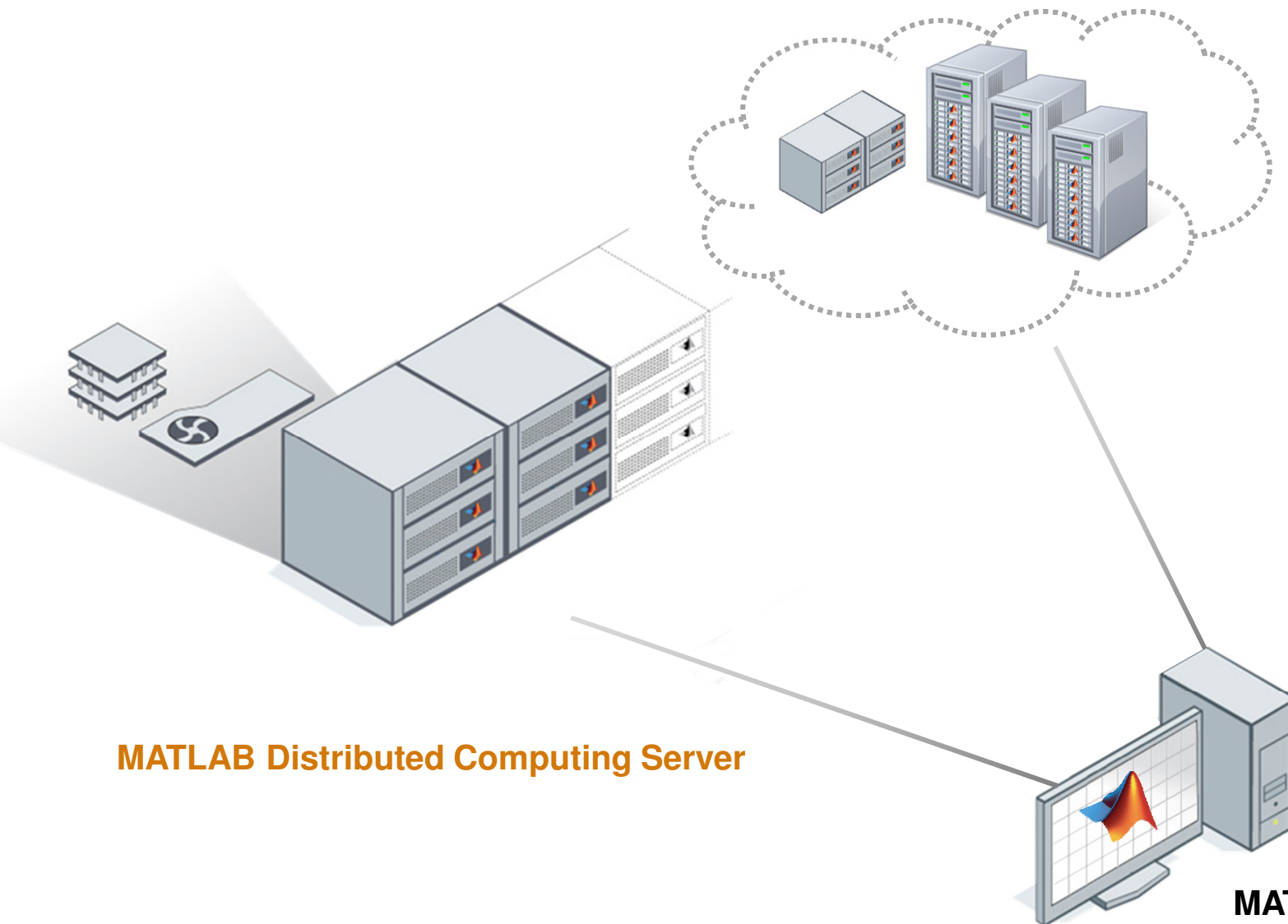


A scalable solution for big data computation !

MATLAB DISTRIBUTED COMPUTING SERVER



Parallel Computing with MATLAB



MATLAB Distributed Computing Server

MATLAB

Parallel Computing Toolbox

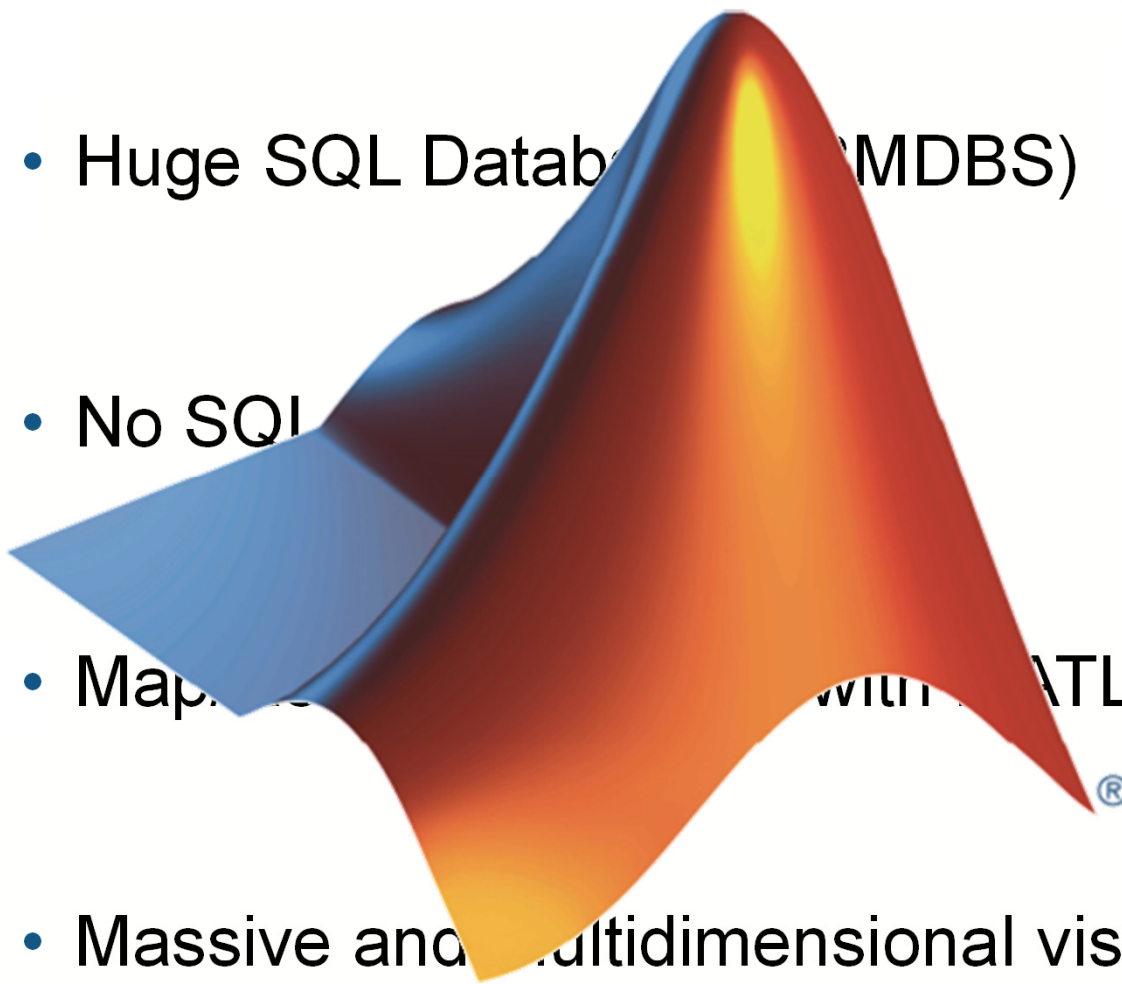
“Parpool” for multiple uses

- parfor – parallel loop for
Task parallel (no communication between workers)
- **spmd** – single program multiple data
Data parallel: communication between « workers » with message passing)
Execute blocks of code on workers
Each worker runs on separate core (ideally)
Separate workspace for each worker
Mix parallel and serial code in same program
- Built-in parallelism in toolboxes: Optimization, Global Optimization, Neural Network, Statistics, Image Processing, Signal Processing, etc...
- GPU support (Tesla, Kepler)
- Advanced programming constructs - interface for experts



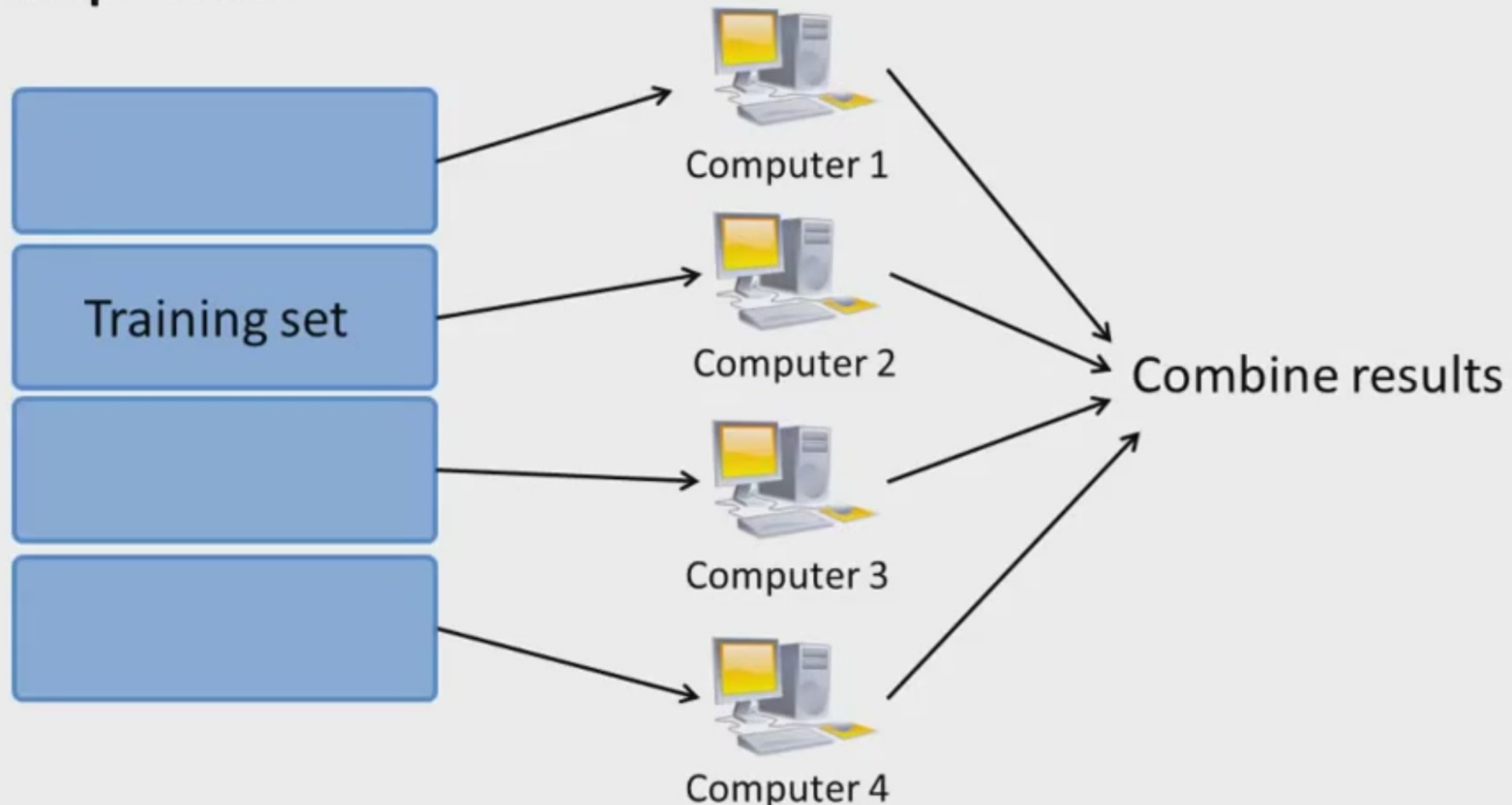
Big data within MATLAB

- Huge flat files
- Huge SQL Databases (MDBS)
- No SQL
- MapReduce with MATLAB MDCS[®]
- Massive and multidimensional visualization
- Machine learning



Distributed memory : MDCS & the MAP/REDUCE paradigm

Map-reduce



Big data & Machine learning

“It’s not who has the best algorithm that wins . It’s who has the most data”

Linear regression with gradient descent

$$h_{\theta}(x) = \sum_{j=0}^n \theta_j x_j$$

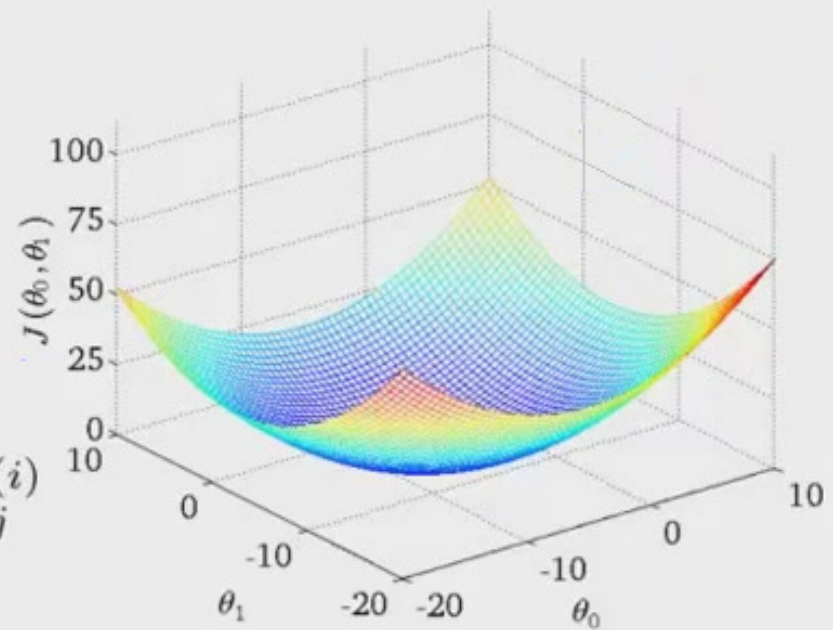
$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(for every $j = 0, \dots, n$)

}



Demo on the booth

Demo

Open a Pool of Local Workers

The previous step probably took a bit of time to import, maybe a full minute. Luckily, th

```
grid_size=12;
try
    parpool('MY_CLUSTER', grid_size);
catch
end
```

Load all different files on all workers

We load here all files distributed on our grid
Note here that the load is done in parallel with a speed up equivalent to the number of cores of your grid

```
tic
spmd
    % or same mounted drive on a linux : dataFolder = '/Volumes/
    disp(files(labindex).name);
    srcFile = fullfile(dataFolder, files(labindex).name);
    % Read the contents back into an array
    fid = fopen(srcFile);
    X = fread(fid, [file_size, nb_predictors+1], '*double');
    fclose(fid);
    % extracting output & input
    Y=X(:,end);
    X(:,end)=[];
    %     rhat = X\Y;
    %     disp(rhat);
end
sprintf('%g seconds to load all different files on each worker.\n', toc)
```

```
Lab 5:
    dataset_2.bin
Lab 8:
    dataset_5.bin
Lab 1:
    dataset_1.bin
Lab 2:
    dataset_10.bin
Lab 3:
    dataset_11.bin
Lab 4:
    dataset_12.bin
Lab 6:
    dataset_3.bin
Lab 7:
    dataset_4.bin
Lab 9:
    dataset_6.bin
Lab 10:
    dataset_7.bin
Lab 11:
    dataset_8.bin
Lab 12:
    dataset_9.bin
```

```
ans =
```

```
2.16159 seconds to load all different files on each worker.
```

One file on
each Lab

```
gpuDevice
```

Map/Reduce paradigm to compute the gradient over distributed test case files over a grid

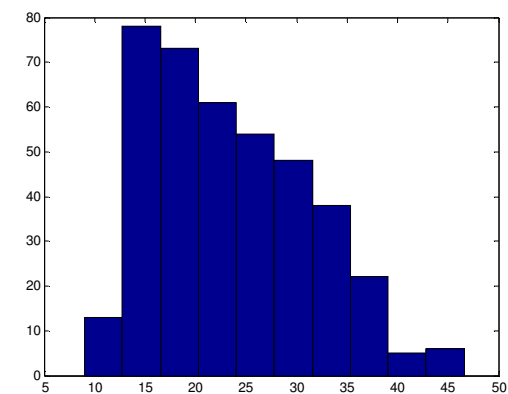
in machine learning most of the time, you can learn independently on different locally stored data sample repositories
what is true here with a linear regression would be true for deep learning neural net with backpropagation (which is also the very same computation of a gradient which is then summed up over all test cases
learning rate alpha

```
alpha=0.01;
% starting point
theta=[0;0;0;0;0;0];
nb_iteration=100;
for i=1:nb_iteration
    % distributed computation for each sample file on each machine of the grid
    % using a spmd bloc
    % each data comes from locally stored files (distributed files)
    % Note here that if our cluster had GPUs we could use inside the spmd
    % to significantly speed up the local computation
    spmd
        grid_local_gradient=(sum(bsxfun(@times, X, X*theta-Y))');
    end
    theta = theta-alpha./(file_size.*grid_size).*sum([grid_local_gradient{:}],2);
end
```

DeviceSelected: 1

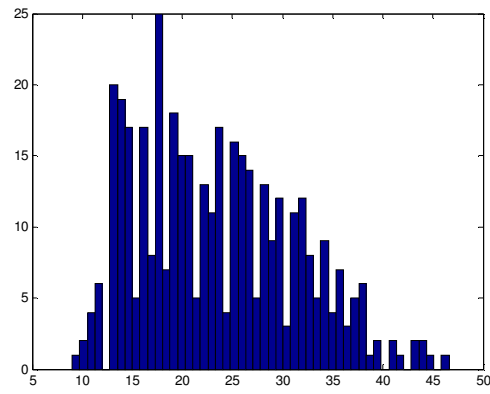
High volume multivariate data visualization

```
>> X = [MPG,Acceleration,Displacement,Weight,Horsepower];
```



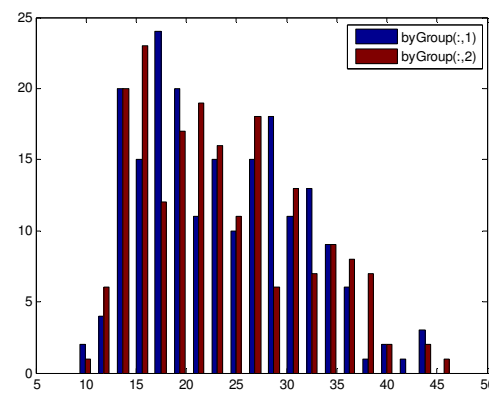
Basic Histogram

```
>> hist(x(:,1))
```



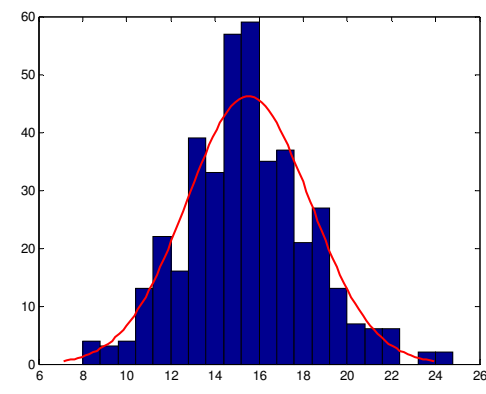
Custom Number of Bins

```
>> hist(x(:,1),50)
```



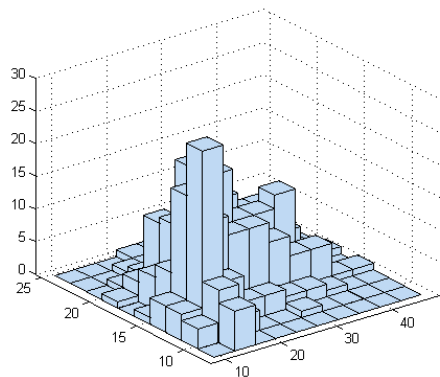
By Group

```
>> hist(byGroup,20)
```



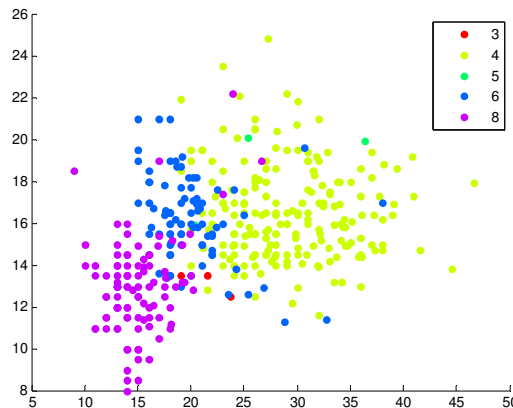
Gaussian fit

```
>> histfit(x(:,2))
```



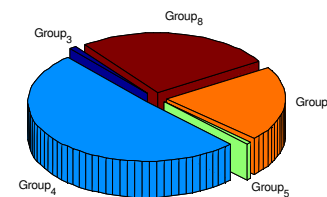
3D Histogram

```
>> hist3(x(:,1:2))
```



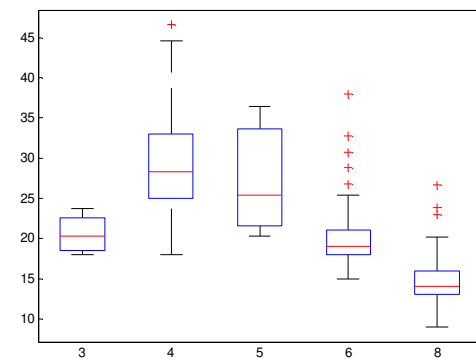
Scatter Plot

```
>> gscatter(x(:,1),x(:,2),groups)
```



Pie Chart

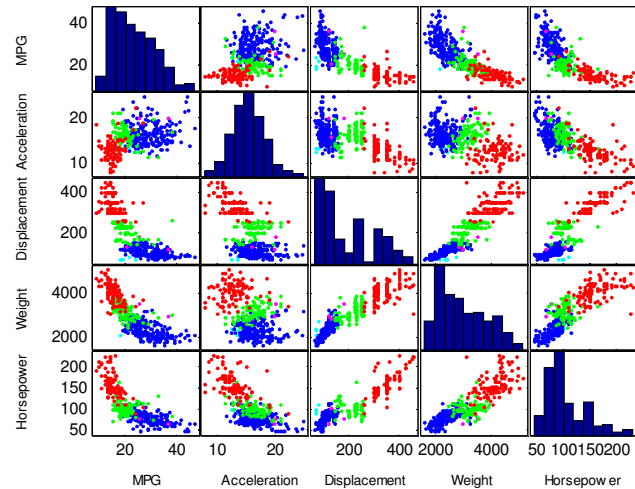
```
>> pie3(proportions,groups)
```



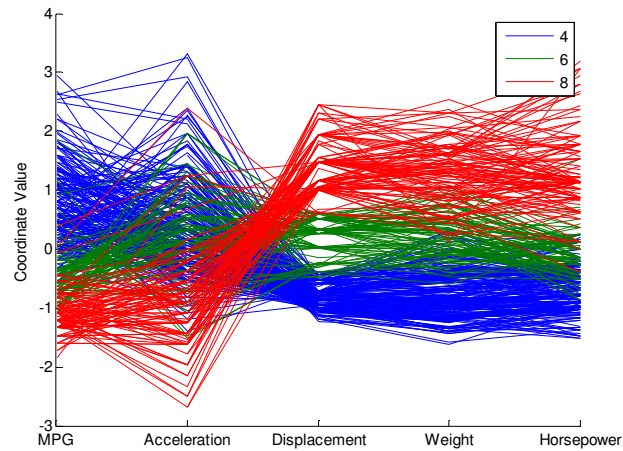
Box Plot

```
>> boxplot(x(:,1),groups)
```

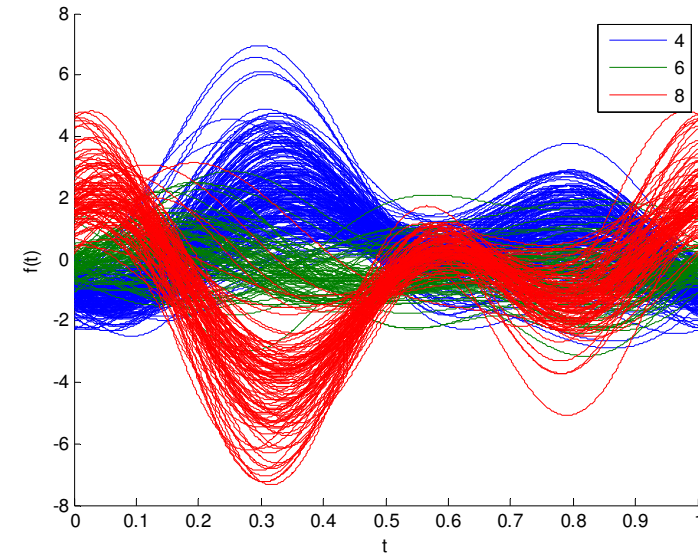
High volume multivariate data visualization



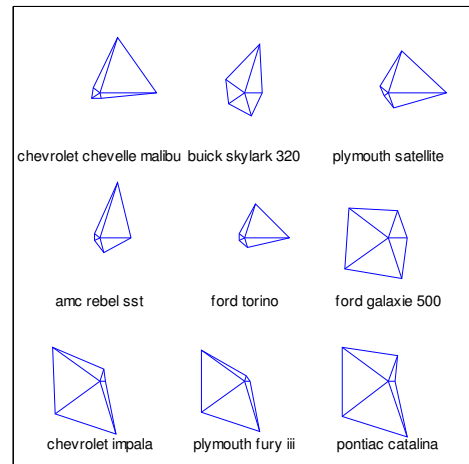
Plot Matrix by Group
`>> gplotmatrix(x,x,groups)`



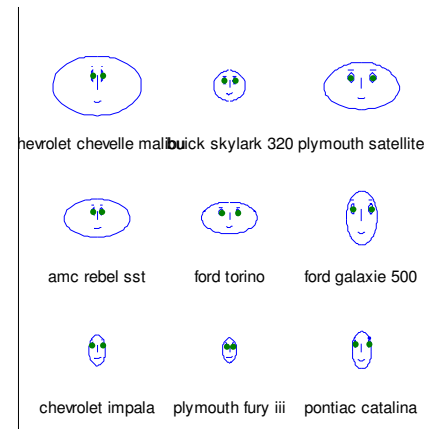
Parallel Coordinates Plot
`>> parallelcoords(x,'Group',groups)`



Andrews' Plot
`>> andrewsplot(x,'Group',groups)`

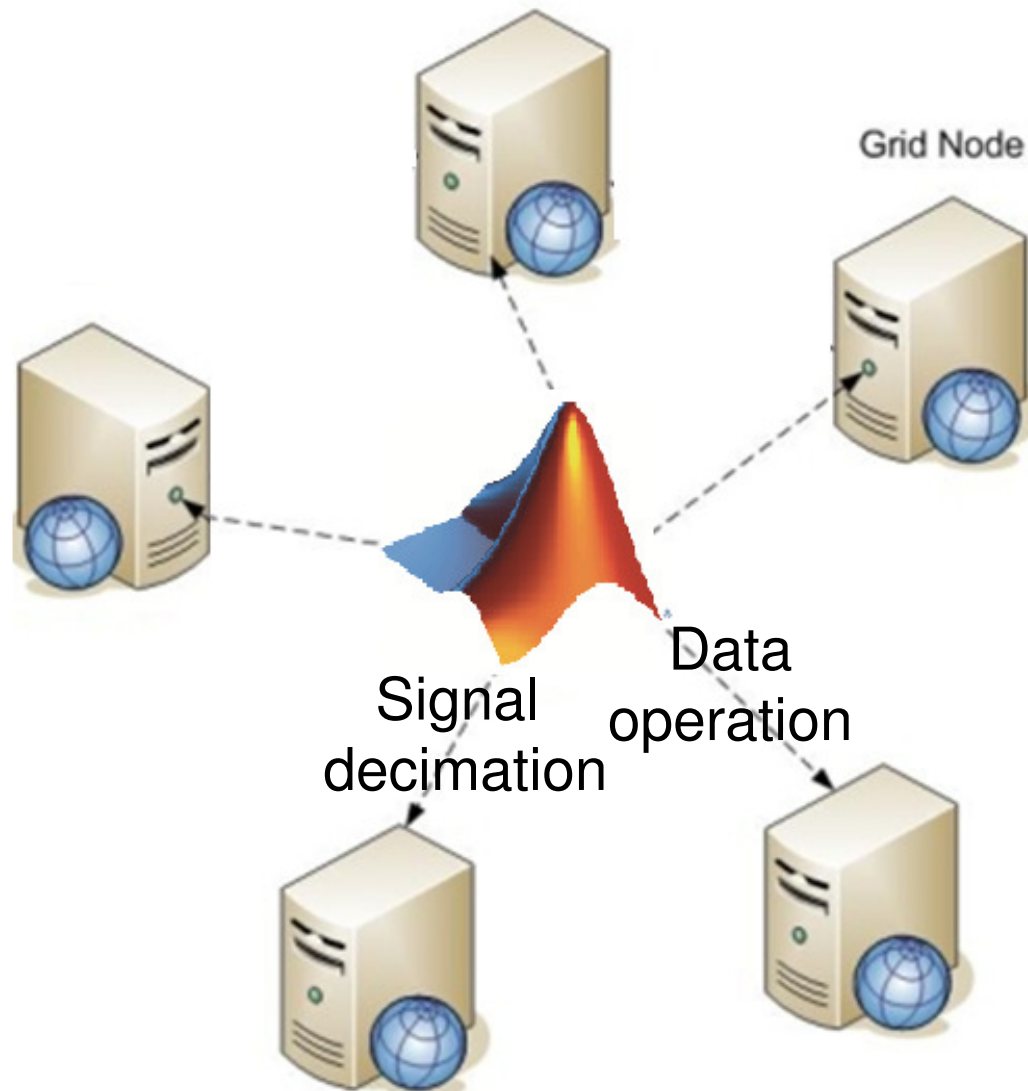


Glyph Plot
`>> glyphplot(x)`

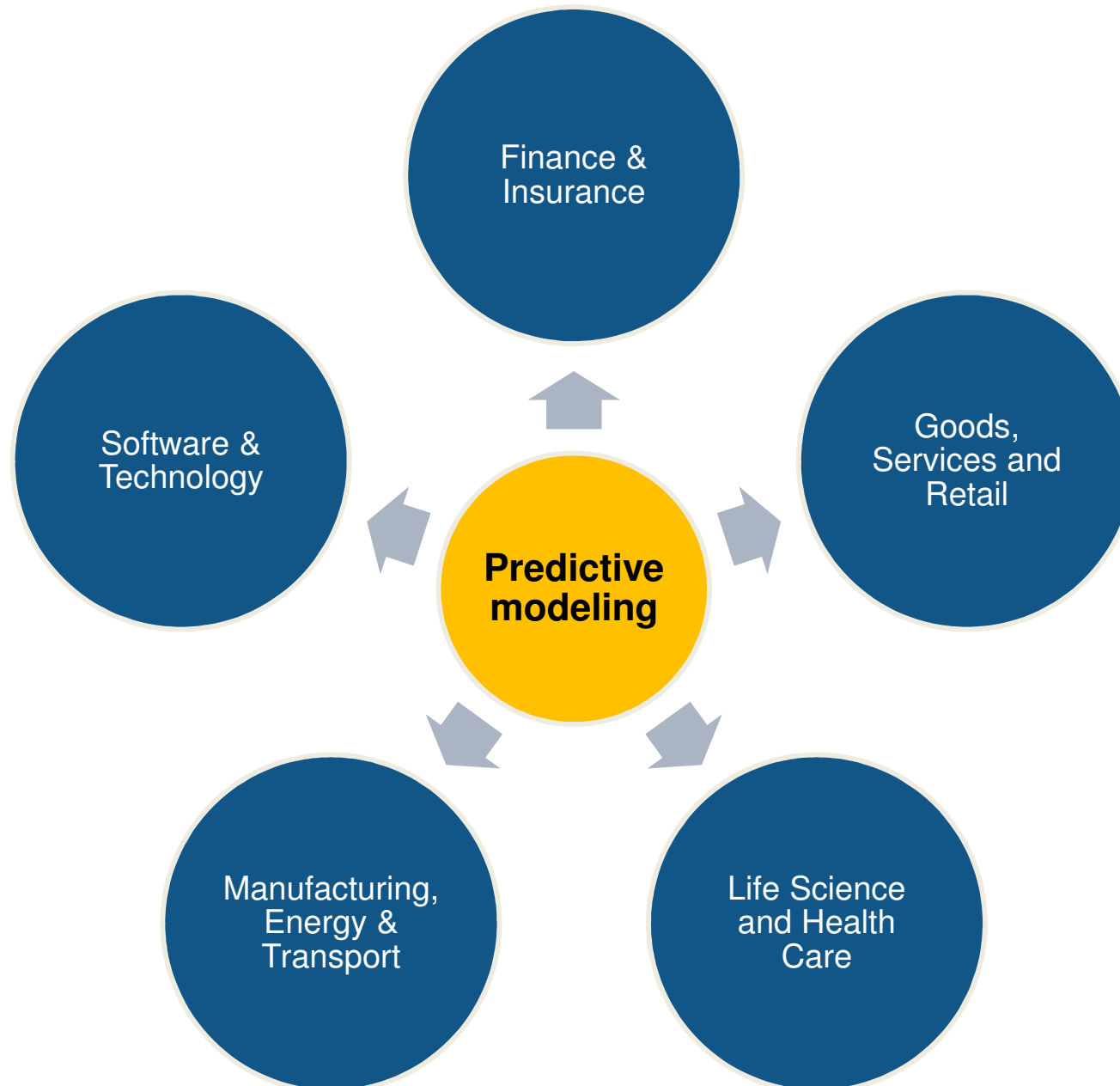


Chernoff Faces
`>> glyphplot(x,'Glyph','face')`

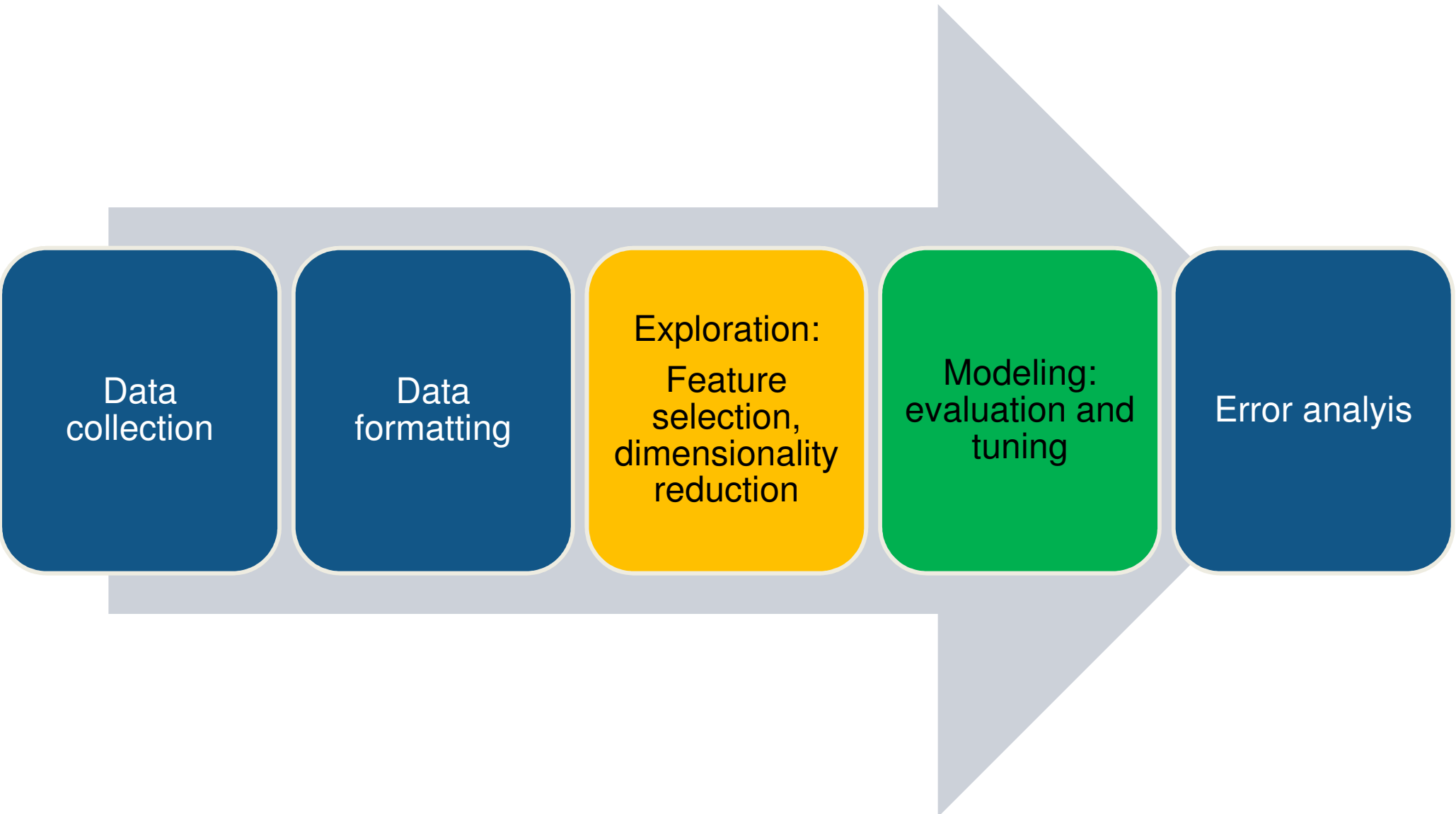
Speeding up flat files reading (or DB requests) & visualizing big data sets thanks to MDCS



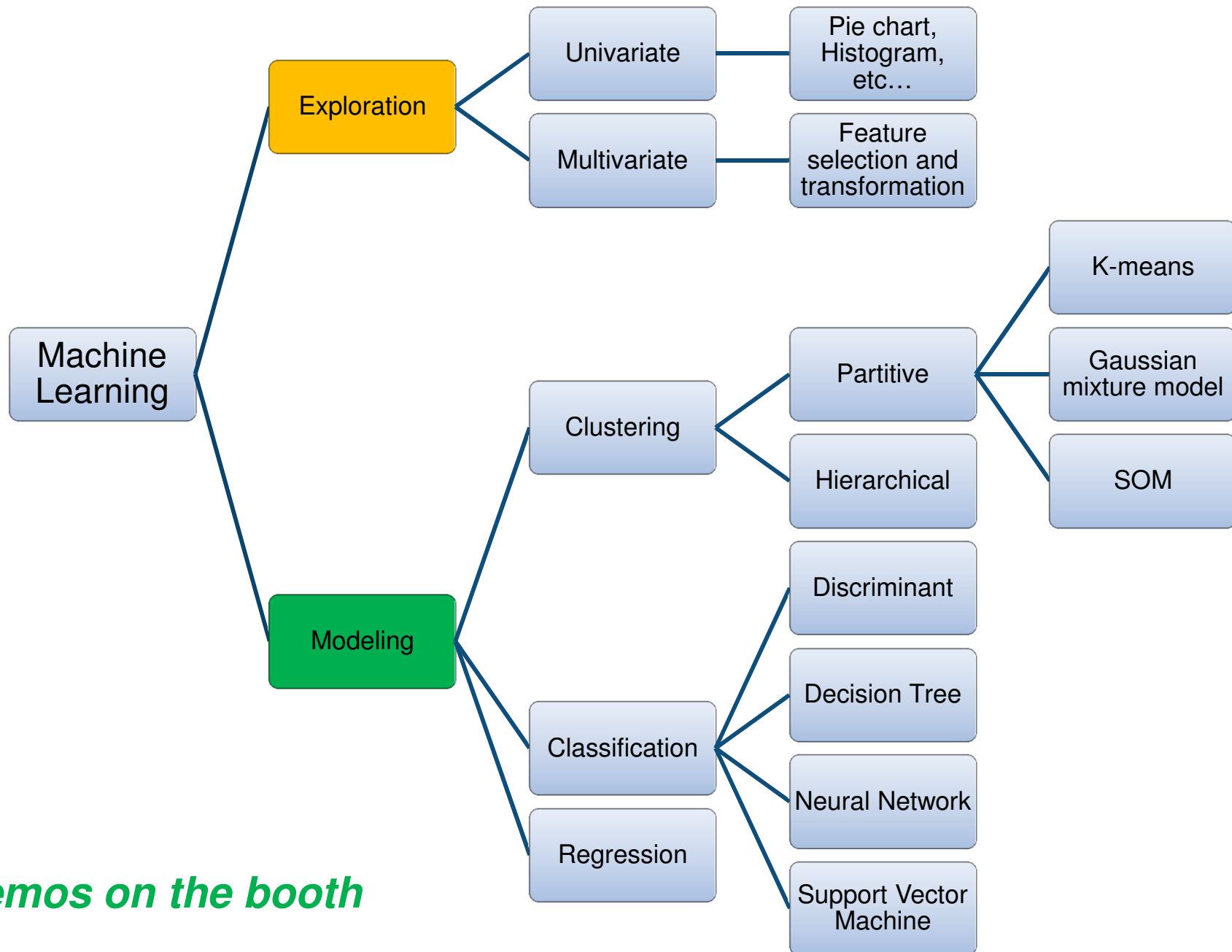
Machine Learning for predictive modeling



Common workflow



Machine learning algorithm proficiency



Demos on the booth

Key take-aways

- Today MATLAB architecture supports Big Data
- MATLAB facilitates Data Analytics for Big Data
- We can support your team projects through Training and Consulting Services

MATLAB EXPO 2014

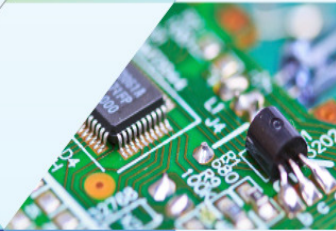
FRANCE

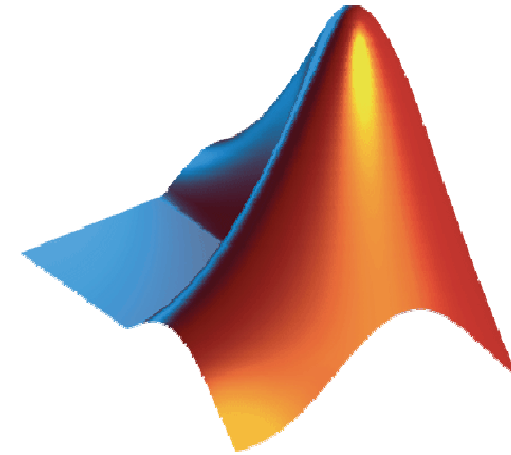
Découvrir, échanger et innover avec MATLAB et Simulink

2 Octobre – Paris

- Big Data: Snecma, Niland and CNP testimonials
- Modeling and Simulation: Airbus, Bosch
- Partners: HP & NVIDIA

Inscription :
matlabexpo.fr





<http://www.mathworks.fr/discovery/big-data-matlab.html>

Programming Parallel Applications (CPU)



Ease of Use

- Parallel-enabled functions in toolboxes
- Simple programming constructs:
`parfor`, `batch`, `distributed`
- Advanced programming constructs:
`createJob`, `labSend`, `spmd`



Greater Control

Programming Parallel Applications (GPU)

Ease of Use

- GPU-enabled functions in toolboxes
- Simple programming constructs:
`gpuArray`, `gather`
- Advanced programming constructs:
`arrayfun`, `spmd`
- Interface for experts:

CUDAKernel, **MEX support**

www.mathworks.com/help/distcomp/run-cuda-or-ptx-code-on-gpu

www.mathworks.com/help/distcomp/run-mex-functions-containing-cuda-code

Greater Control