

Parallel Computing / Server Solutions

[E-MOBİLİTE]

[YAPISAL ANALİZ]

[MOBİLİTE]

[AR-GE DERGİSİ]
[EĞİTİM]

[SİMÜLASYON]

[SAVUNMA SANAYİ]

[OTOMOTİV]

[YAPAY ZEKA]

[AR-GE PROJELERİ]

Parallel Computing / Server Solutions

[NÜKLEER TEKNOLOJİ]

[İLERİ MÜHENDİSLİK]

[YEŞİL ENERJİ]

[SİMÜLASYON]

[YAZILIM]

[YAPISAL ANALİZ]

[MATLAB & SIMULINK]

Mustafa Onur Özkan
MATLAB Information Systems Specialist



Agenda

- 01 Introduction To The Presentation**
- 02 MATLAB Parallel Computing / Server**
- 03 Installation, Configuration and System Requirement**
- 04 Usage Scenarios / User Story**
- 05 Common Problems and Solutions**

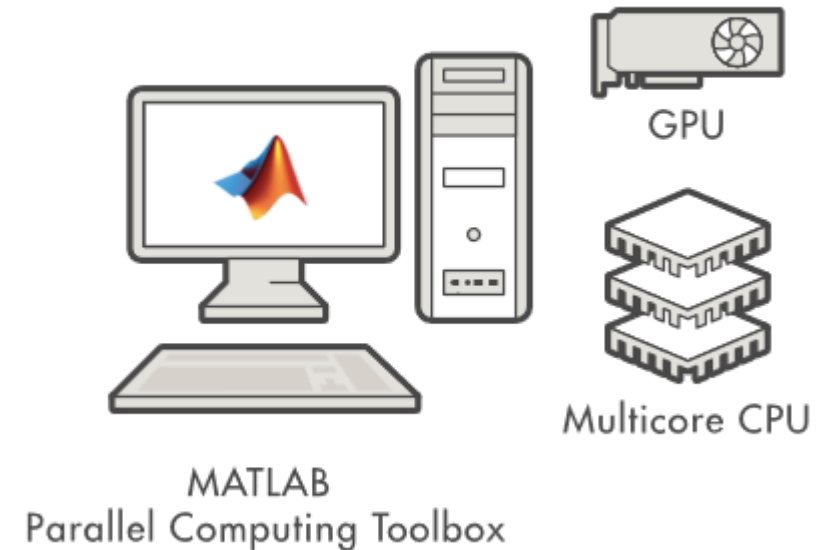
Purpose and Scope of the Presentation

- Purpose of the presentation: To examine the features, installation and use of MATLAB Parallel Server/ Computing Solution in detail.
- Scope: Parallel Computing Basic concepts, installation and configuration, usage scenarios, advanced topics, frequently encountered problems and solutions.



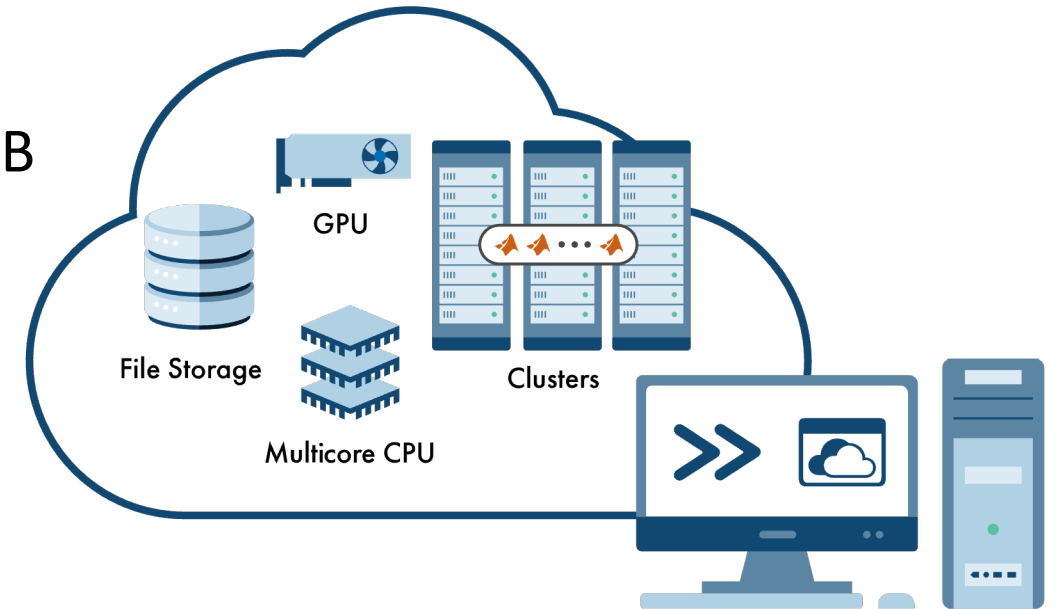
What is Parallel Computing Toolbox?

- Parallel Computing Toolbox offers MATLAB users the ability to perform parallel computing on their local machines.
- This Toolbox allows your local computer to speed up operations by using its processors and GPUs.



What is MATLAB Parallel Server ?

- MATLAB Parallel Server allows you to run MATLAB operations in parallel on a cluster of computers.
- It increases the speed of calculations by distributing the workload across multiple computers.



Why Do We Need Parallel Computing?

- Fast processing of large data sets.
- Accelerating simulation and modeling processes.
- Cost savings through effective use of resources.



Parallel - Enabled Toolboxes (MATLAB Product Family)

Image Processing

Batch Image Processor, Block Processing, GPU-enabled functions



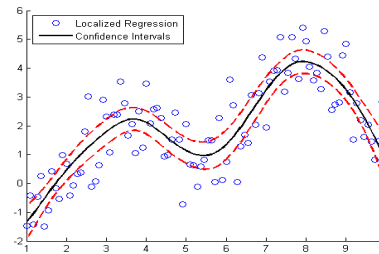
Original Image of Peppers



Recolored Image of Peppers

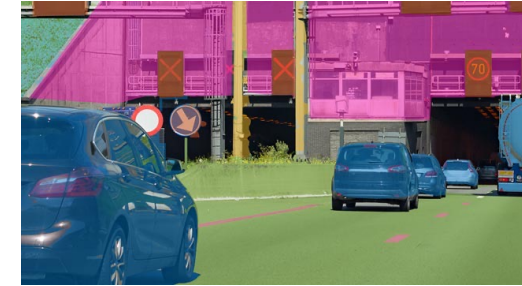
Statistics and Machine Learning

Resampling Methods, k-Means clustering, GPU-enabled functions



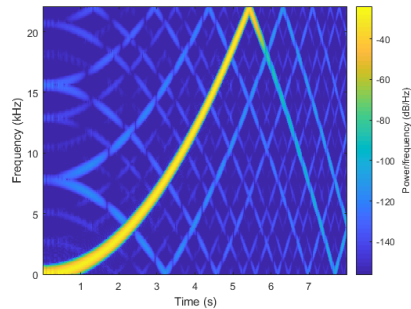
Deep Learning

Deep Learning, Neural Network training and simulation



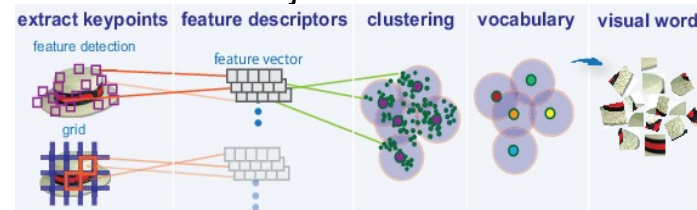
Signal Processing and Communications

GPU-enabled FFT filtering, cross correlation, BER simulations



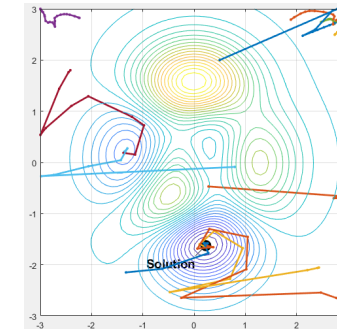
Computer Vision

Bag-of-words workflow, object detectors



Optimization & Global Optimization

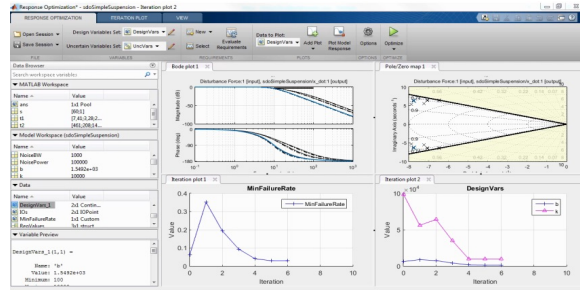
Estimation of gradients, parallel search



Parallel - Enabled toolboxes (Simulink Product Family)

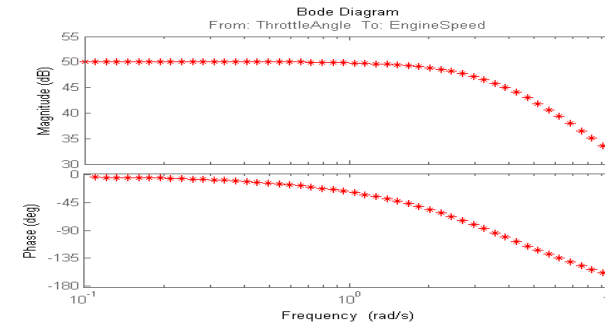
Simulink Design Optimization

Response optimization, sensitivity analysis, parameter estimation



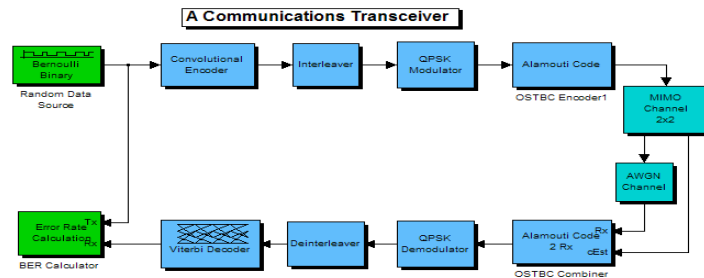
Simulink Control Design

Frequency response estimation



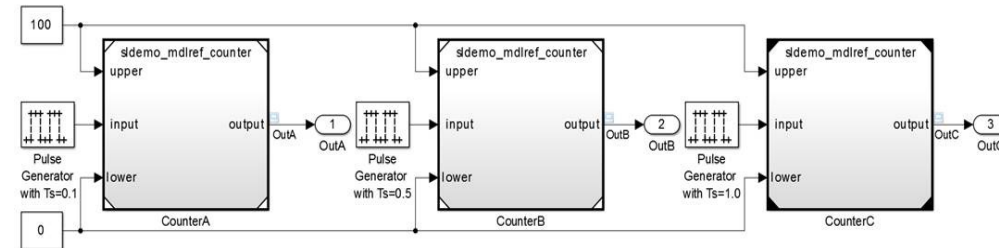
Communication Systems Toolbox

GPU-based System objects for Simulation Acceleration



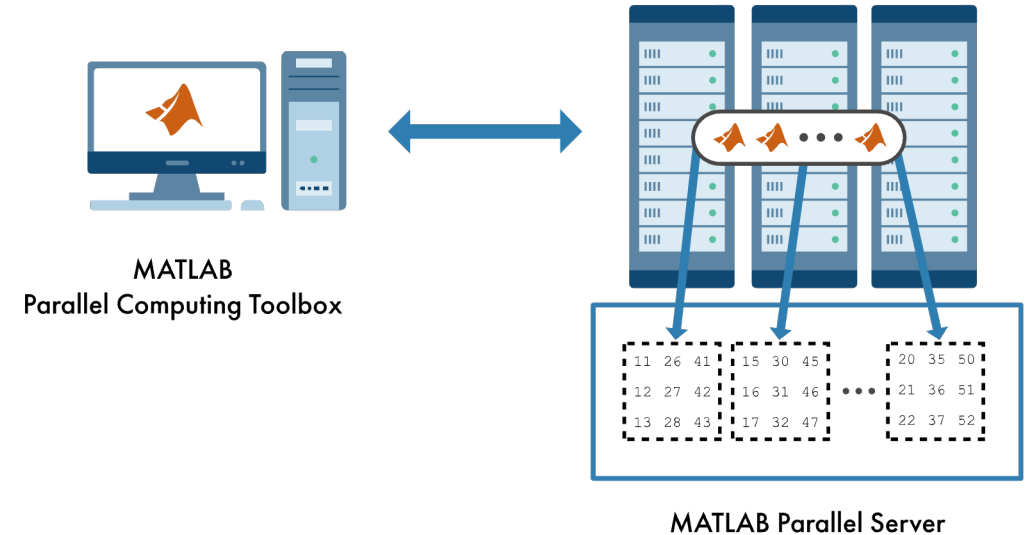
Simulink/Embedded Coder

Generating and building code



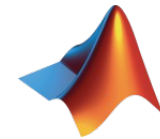
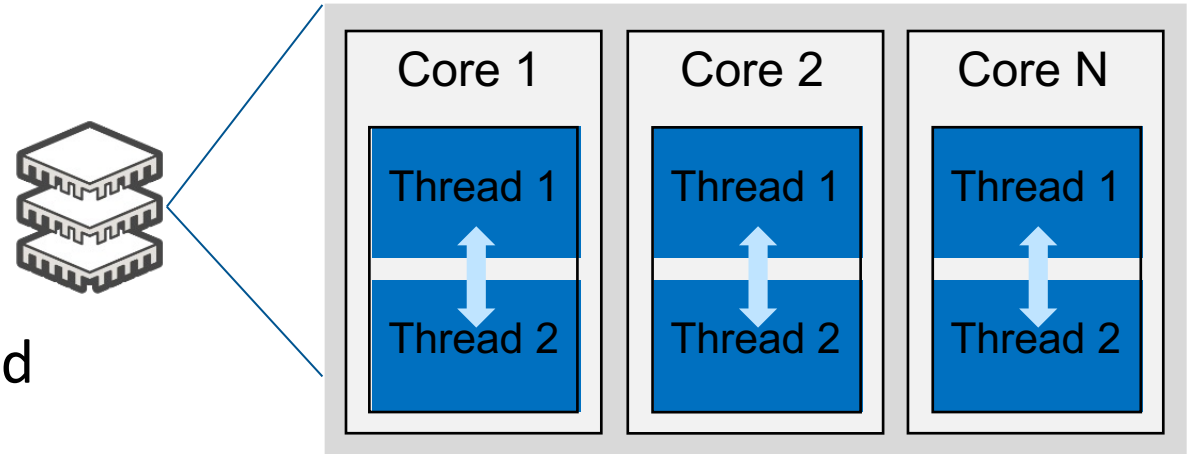
MATLAB Parallel Server Basic Concepts

- Cluster: The structure where computational resources come together.
- Worker: Independent processing units that perform parallel computation.
- Job: The set of operations or tasks to be performed.
- Task: Independent processing unit within a job.



MATLAB Parallel Server Architectural

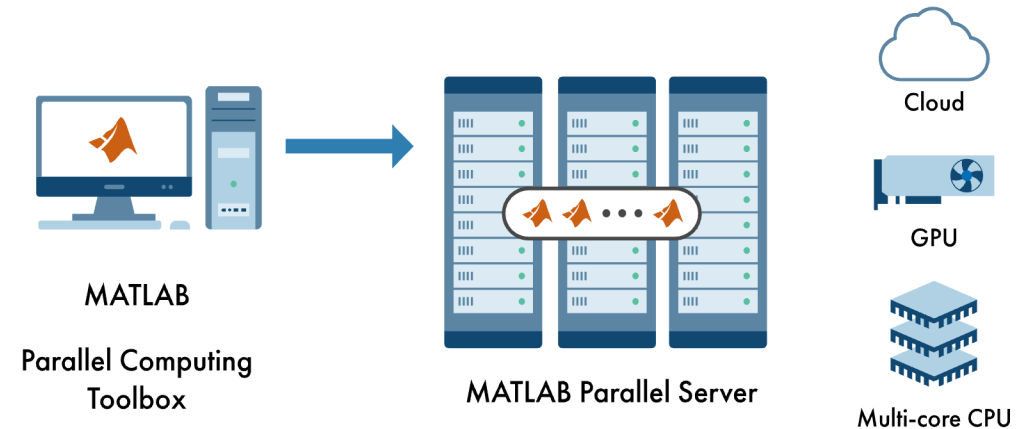
- Server-client structure.
- Scheduler and Resource Manager components.
- Communication between MATLAB client and worker.



MATLAB Parallel Server Working Principle

- Creation of jobs and tasks.
- Distribution and execution of tasks in parallel.
- Collection and processing of results.

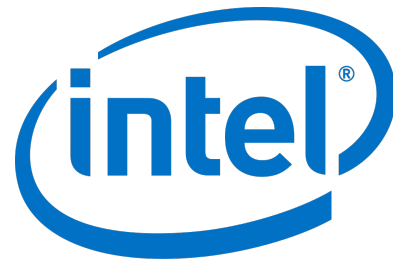
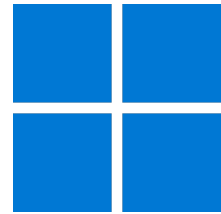
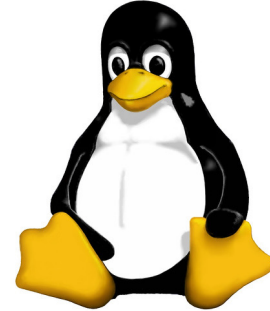
```
parpool("HPC",1000);  
  
parfor ii = 1:3000  
    c(ii,:) = eig(rand(1000));  
end
```



Installation and Configuration – System Requirement

System Requirement

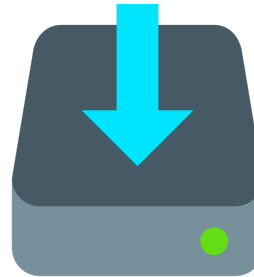
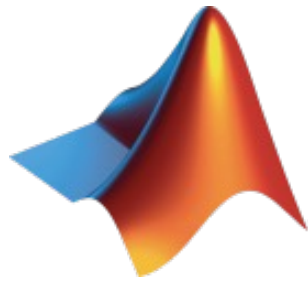
- Supported operating systems.
- Required hardware components (CPU, GPU, RAM, Disk, etc.)



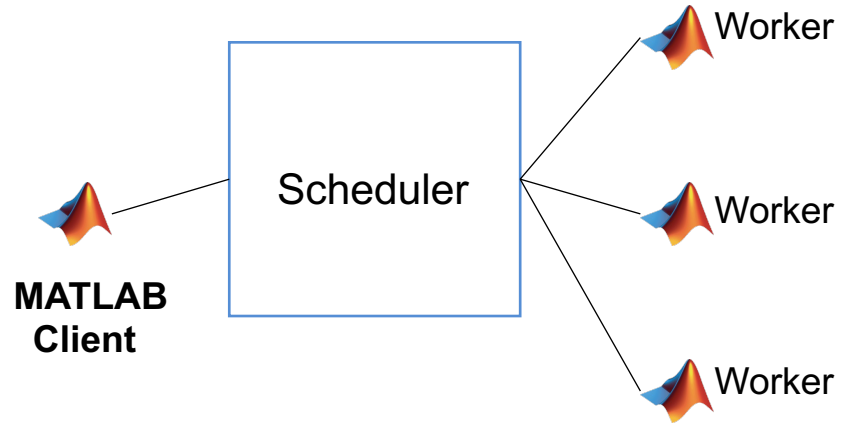
Red Hat



Supported Job Schedulers



MATLAB MJS



Admin Center

File Hosts Scheduler Workers Help

Hosts

Add or Find...		Host		MJS Service		MATLAB Job Sche...	Workers
Start MJS Service...	Hostname	Reachable	Cores	Status	Up Since	Name	Count
Stop MJS Service...	ComputeNodeHostname	yes	6	running	2019-01-03 15:55		4
Test Connectivity...	HeadNodeHostname	yes	6	running	2019-01-03 15:53	myJobManager	0

MATLAB Job Scheduler

Start...	Name	Hostname	Status	Up Since	Workers
Stop...	myJobManager	HeadNodeHostname	running	2019-01-03 16:03	4
Resume					

Workers

Add or Find...		Worker			MATLAB Job Scheduler		
Start...	Name	Hostname	Status	Up Since	Connection	Name	Hostname
Stop...	Worker1	ComputeNode...	idle	2019-01-03 16:03	connected	myJobManager	HeadNodeHos...
Resume	Worker2	ComputeNode...	idle	2019-01-03 16:04	connected	myJobManager	HeadNodeHos...
	Worker3	ComputeNode...	idle	2019-01-03 16:04	connected	myJobManager	HeadNodeHos...
	Worker4	ComputeNode...	idle	2019-01-03 16:04	connected	myJobManager	HeadNodeHos...

Last updated: 03/01/19 16:04

Update every 2 minutes Update Now



Parallel Loops / Parfor Loop

- Parfor loop is used to run independent iterations in parallel.
- Parfor runs the iteration independently and speeds up the process.
- In this example, the total for each row is calculated in parallel.

```
Editor - C:\Users\onur.ozkan\OneDrive - figes.com.tr\Masaüstü\Optimization Example\bigdata.m
myObjectiveFunction.m x mainNoneParallel.m x mainParallel.m x bigdata.m x +
1 % Matris boyutu
2 N = 1000;
3
4 % Rastgele bir matris oluşturun
5 A = rand(N, N);
6 B = zeros(N, 1);
7
8 % parfor döngüsü
9 parfor i = 1:N
10     B(i) = sum(A(i, :));
11 end
12
13 disp('Sonuç:');
14 disp(B);
15
```



Parfeval Loop

- Parfeval is used to perform asynchronous operations
- In this example we will calculate the average of a sequence of random numbers.
- We will perform the calculation in the background with parfeval, so the MATLAB session will be available before the calculation is completed.

```
Editor - C:\Users\onur.ozkan\OneDrive - figes.com.tr\Masaüstü\Parallel Server Example MATLAB\calculateMeanExam
calculateMeanExample.m x Perfor_For_Example.m x parsim_example.m x calculateMean.m x unt
1 % Rastgele bir sayı dizisinin ortalamasını hesaplamak için parfeval
2 % kullanımı Örneği
3
4 % Dizin uzunluğunu belirliyoruz
5 n = 1e7;
6
7 % Asenkron olarak 'calculateMean' fonksiyonunu çağırıyoruz
8 f = parfeval(@calculateMean, 1, n);
9
10 % Görevin tamamlanmasını bekleyin ve sonucu alın
11 wait(f);
12 meanValue = fetchOutputs(f);
13
14 % Sonucu ekrana yazdırıyoruz
15 disp("Hesaplanan Ortalama Değer: " + meanValue);
16
17 % Fonksiyonu Tanımlıyoruz.
18 function result = calculateMean(n)
19     data = rand(1, n); % n uzunluğunda rastgele bir dizi oluştur
20     result = mean(data); % Dizin ortalamasını hesapla
21 end
22
```



SPMD Blocks – Codistributed Function

- spmd (single program multiple data) enables parallel computation by sharing data between different workers
- A distributed matrix is created with the codistributed function.

```
Editor - C:\Users\onur.ozkan\OneDrive - figes.com.tr\Masaüstü\Optimization Example\bigdata.m
myObjectiveFunction.m x mainNoneParallel.m x mainParallel.m x bigdata.m x +
1 % Matris boyutu
2 N = 1000;
3
4 % spmd bloğu içinde paralel hesaplama
5 spmd
6     A = codistributed.rand(N, N); % Dağıtılmış matris oluşturma
7     B = sum(A, 2); % Satır toplamlarını hesaplama
8 end
9
10 disp('Sonuç:');
11 disp(B{1}); % İlk işçinin sonucunu gösterme
12
```



Batch usage (HPC)

- The batch command runs a specific function in the background.
- When the job is completed with `fetchOutputs`, the result is obtained.

```
Editor - C:\Users\onur.ozkan\OneDrive - figes.com.tr\Masaüstü\Optimization Example\bigdata.m
myObjectiveFunction.m x mainNoneParallel.m x mainParallel.m x bigdata.m x +
1      % Uzak bir iş oluştur ve çalıştır
2      j = batch(@( ) sum(rand(1e6,1)), 'Profile', 'local');
3
4      % İşin tamamlanmasını bekle
5      wait(j);
6
7      % Sonucu al ve göster
8      result = fetchOutputs(j);
9      disp('Sonuç:');
10     disp(result{1});
11
12     % İş i temizle
13     delete(j);
14
```



createJob Function

- The createJob function creates and runs jobs on the parallel pool.
- Tasks are added to the job with createTask and the job is started with submit.

```
Editor - C:\Users\onur.ozkan\OneDrive - figes.com.tr\Masaüstü\Optimization Example\bigdata.m
myObjectiveFunction.m x mainNoneParallel.m x mainParallel.m x bigdata.m x +
1 % Paralel havuz oluşturma
2 p = parpool('local');
3
4 % İş oluşturma
5 job = createJob(p);
6
7 % Görev ekleme
8 createTask(job, @sum, 1, {rand(1e6,1)});
9
10 % İşi çalıştırma
11 submit(job);
12
13 % İşin tamamlanmasını bekle
14 wait(job);
15
16 % Sonucu al ve göster
17 result = fetchOutputs(job);
18 disp('Sonuç:');
19 disp(result{1});
20
21 % İşi temizle
22 delete(job);
23
```



Usage Scenarios

Application Examples:

- Analysis of large data sets and processing.
- Tall Array Usage

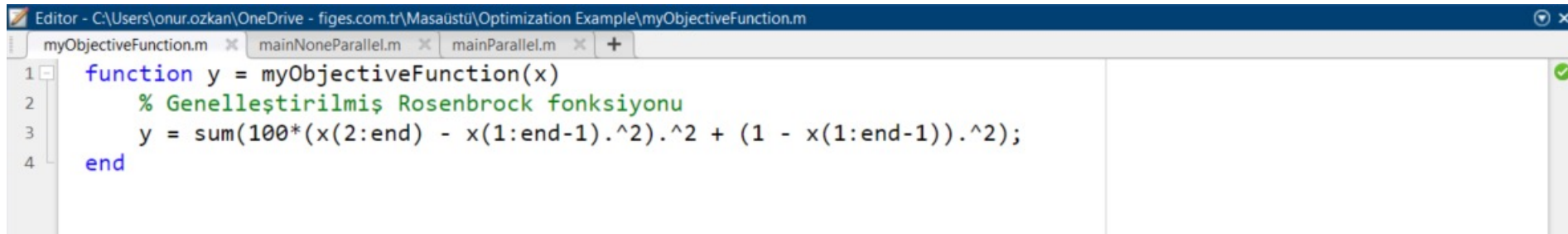
```
myObjectiveFunction.m x mainNoneParallel.m x mainParallel.m x bigdata.m x +
1 % Paralel pool oluşturma
2 parpool('local', 4); % 4 işçi ile paralel havuz oluşturur
3
4 % Büyük veri seti oluşturma
5 n = 1e7; % 10 milyon satır
6 d = 20; % 20 sütun
7 data = randn(n, d); % Rastgele sayılardan oluşan büyük veri seti
8
9 % Tall array oluşturma
10 tallData = tall(data);
11
12 % Zaman ölçümünü başlat
13 tic;
14
15 % Paralel hesaplama ile ortalama ve standart sapma hesaplama
16 opts = {'UseParallel', true};
17 meanValues_parallel = mean(tallData, opts{:});
18 stdValues_parallel = std(tallData, opts{:});
19
20 % Hesaplamaları yürütme
21 meanValues_parallel = gather(meanValues_parallel);
22 stdValues_parallel = gather(stdValues_parallel);
23
24 % Zaman ölçümünü durdur ve sonuçları görüntüle
25 time_parallel = toc;
26 disp(['Paralel hesaplama süresi: ', num2str(time_parallel), ' saniye']);
27 disp('Ortalama Değerler (Paralel):');
28 disp(meanValues_parallel);
29 disp('Standart Sapmalar (Paralel):');
30 disp(stdValues_parallel);
31 |
```



Usage Scenarios

Application Examples:

- Solution of complex optimization problems.
- Rosenbrock Function



```
Editor - C:\Users\onur.ozkan\OneDrive - figes.com.tr\Masaüstü\Optimization Example\myObjectiveFunction.m
myObjectiveFunction.m x mainNoneParallel.m x mainParallel.m x +
1 function y = myObjectiveFunction(x)
2     % Genelleştirilmiş Rosenbrock fonksiyonu
3     y = sum(100*(x(2:end) - x(1:end-1).^2).^2 + (1 - x(1:end-1)).^2);
4 end
```



Usage Scenerios

- Without using the Parallel Computing toolbox, the relevant function produced results in 1 minute and 46 seconds.

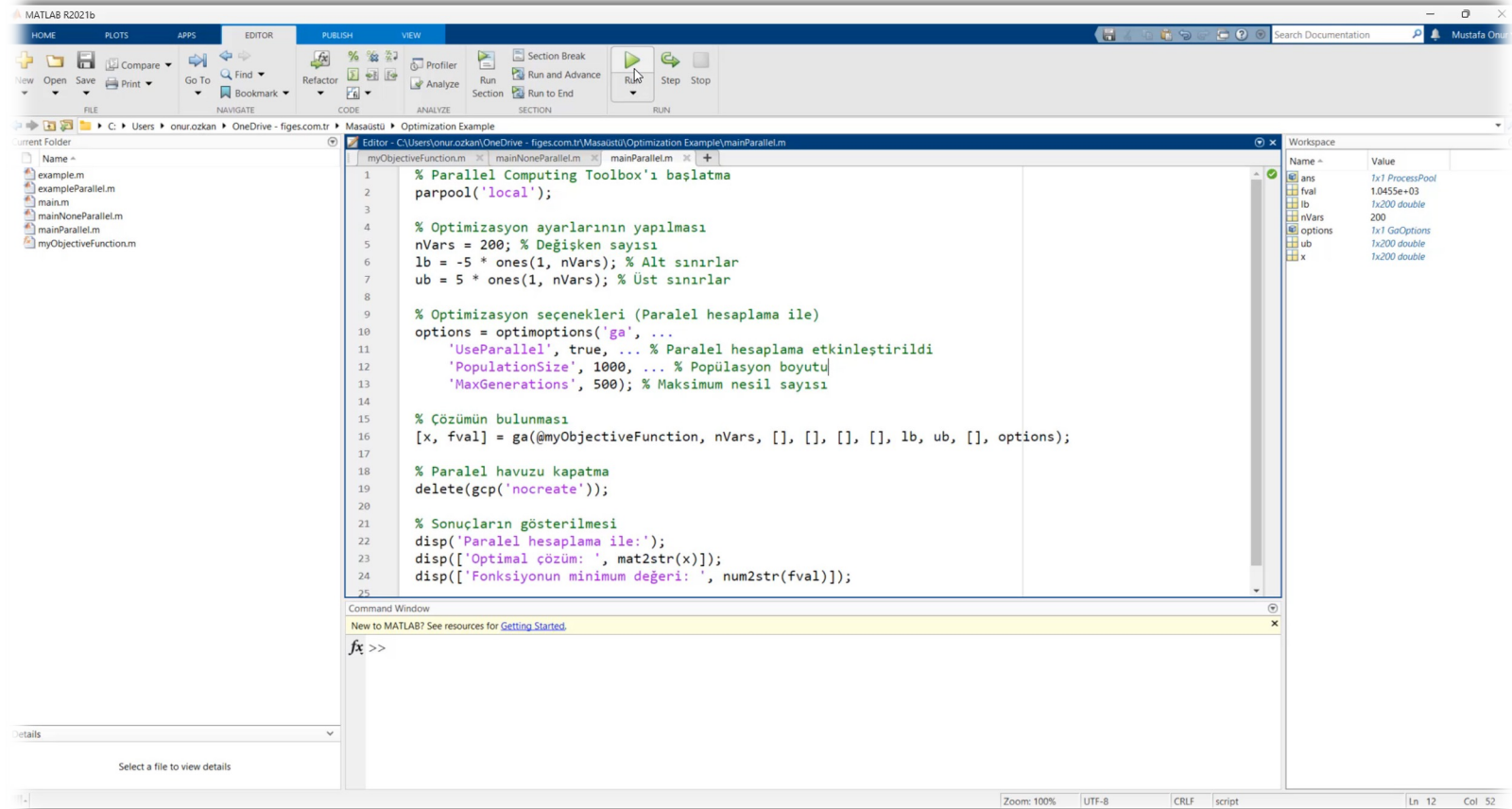
```
1 % Optimizasyon ayarlarının yapılması
2 nVars = 200; % Değişken sayısı
3 lb = -5 * ones(1, nVars); % Alt sınırlar
4 ub = 5 * ones(1, nVars); % Üst sınırlar
5
6 % Optimizasyon seçenekleri (Paralel hesaplama olmadan)
7 options = optimoptions('ga', ...
8     'UseParallel', false, ... % Paralel hesaplama devre dışı
9     'PopulationSize', 750, ... % Popülasyon boyutu
10    'MaxGenerations', 1000); % Maksimum nesil sayısı
11
12 % Çözümün bulunması
13 [x, fval] = ga(@myObjectiveFunction, nVars, [], [], [], [], lb, ub, [], options);
14
15 % Sonuçların gösterilmesi
16 disp('Paralel hesaplama olmadan:');
17 disp(['Optimal çözüm: ', mat2str(x)]);
18 disp(['Fonksiyonun minimum değeri: ', num2str(fval)]);
19
```

Name	Value
ans	1x1 ProcessPool
fval	1.2621e+03
lb	1x200 double
nVars	200
options	1x1 GoOptions
ub	1x200 double
x	1x200 double



Usage Scenerios

- Using the Parallel Computing toolbox, the relevant function gave results in 1 minute.



The image shows the MATLAB R2021b interface. The main editor window displays a script for parallel optimization. The script includes comments in Turkish and MATLAB code. The code sets up a parallel pool, defines optimization variables and constraints, sets optimization options for parallel execution, and runs the optimization using the 'ga' function. The results are displayed in the command window.

```
1 % Parallel Computing Toolbox'ı başlatma
2 parpool('local');
3
4 % Optimizasyon ayarlarının yapılması
5 nVars = 200; % Değişken sayısı
6 lb = -5 * ones(1, nVars); % Alt sınırlar
7 ub = 5 * ones(1, nVars); % Üst sınırlar
8
9 % Optimizasyon seçenekleri (Paralel hesaplama ile)
10 options = optimoptions('ga', ...
11     'UseParallel', true, ... % Paralel hesaplama etkinleştirildi
12     'PopulationSize', 1000, ... % Popülasyon boyutu
13     'MaxGenerations', 500); % Maksimum nesil sayısı
14
15 % Çözümün bulunması
16 [x, fval] = ga(@myObjectiveFunction, nVars, [], [], [], [], lb, ub, [], options);
17
18 % Paralel havuzu kapatma
19 delete(gcp('nocreate'));
20
21 % Sonuçların gösterilmesi
22 disp('Paralel hesaplama ile:');
23 disp(['Optimal çözüm: ', mat2str(x)]);
24 disp(['Fonksiyonun minimum değeri: ', num2str(fval)]);
25
```

The Command Window shows the following output:

```
f >>
```

Name	Value
ans	1x1 ProcessPool
fval	1.0455e+03
lb	1x200 double
nVars	200
options	1x1 GoOptions
ub	1x200 double
x	1x200 double



Usage Scenerios

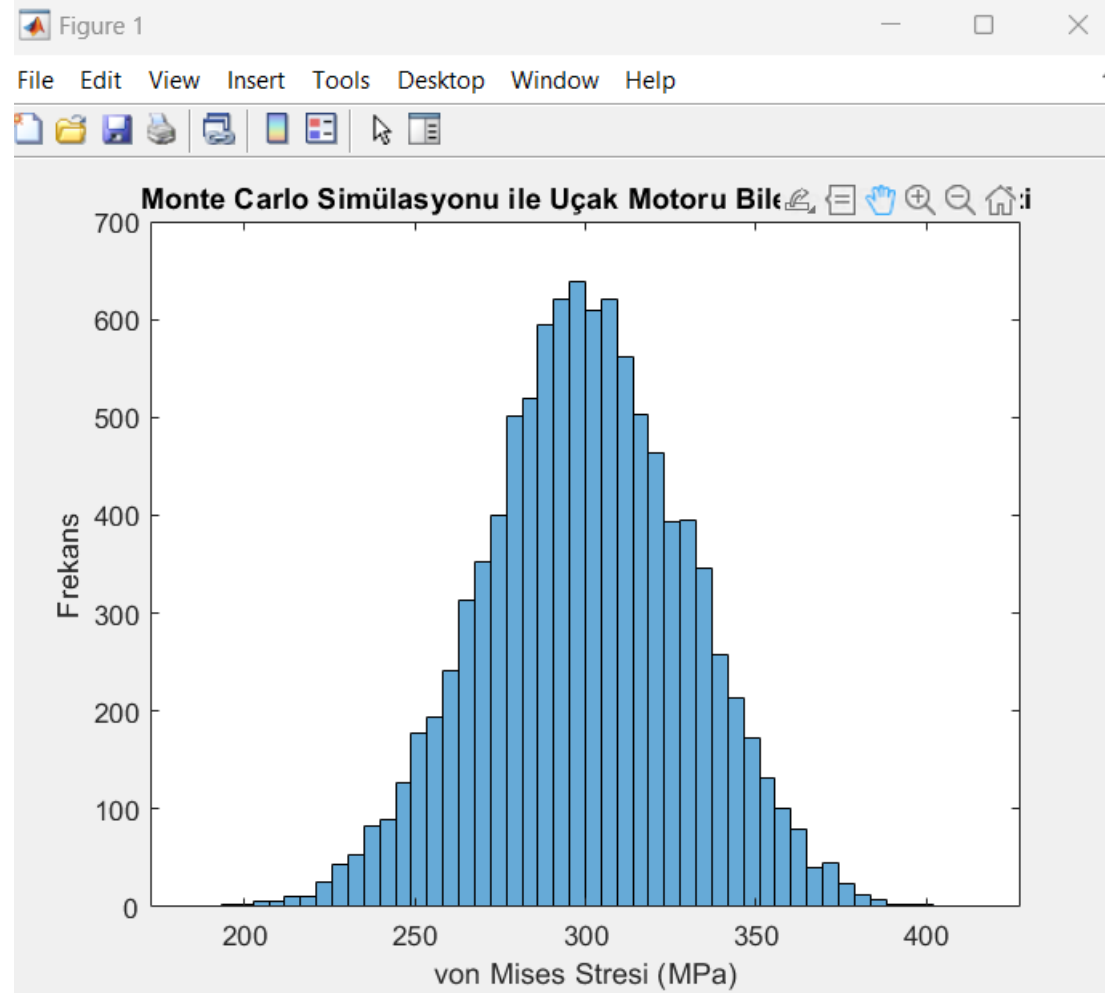
Application Examples:

- Simulation and modeling processes.
- Monte Carlo Simulation
- Aircraft Engine Component Stress Analysis

```
Editor - C:\Users\onur.ozkan\OneDrive - figes.com.tr\Masaüstü\Optimization Example\bigdata.m
myObjectiveFunction.m  mainNoneParallel.m  mainParallel.m  bigdata.m  +
1      % Uçak Motoru Bileşeni Stres Analizi - Monte Carlo Simülasyonu
2
3      % Paralel havuzu başlatın (mevcut local core'ları kullanarak)
4      if isempty(gcf('nocreate'))
5          parpool('local');
6      end
7
8      % Parametreler
9      numSimulations = 10000; % Simülasyon sayısı
10     meanTemperature = 800; % Ortalama sıcaklık (Celsius)
11     stdDevTemperature = 50; % Sıcaklık standart sapması (Celsius)
12     meanStress = 300; % Ortalama mekanik stres (MPa)
13     stdDevStress = 30; % Mekanik stres standart sapması (MPa)
14
15     % Simülasyon sonuçlarını saklamak için bir matris
16     stressResults = zeros(numSimulations, 1);
17
18     % Monte Carlo simülasyonu - paralel hesaplama
19     parfor i = 1:numSimulations
20         % Her simülasyon için sıcaklık ve stres değerlerini hesapla
21         temperature = meanTemperature + stdDevTemperature * randn();
22         stress = meanStress + stdDevStress * randn();
23         % Stres analizi hesaplaması (örneğin, von Mises stresi)
24         vonMisesStress = sqrt(stress^2 + 3 * (temperature * 1e-3)^2);
25         % Sonucu kaydet
26         stressResults(i) = vonMisesStress;
27     end
28
29     % Sonuçların analiz edilmesi
30     meanStressResult = mean(stressResults);
31     stdStressResult = std(stressResults);
```



Usage Scenarios



Usage Scenarios

Application Examples:

- Simulation and modeling processes.
- Monte Carlo Simulation
- Aircraft Engine Component Stress Analysis

```
30 meanStressResult = mean(stressResults);
31 stdStressResult = std(stressResults);
32
33 % Sonuçların görselleştirilmesi
34 histogram(stressResults, 50);
35 title('Monte Carlo Simülasyonu ile Uçak Motoru Bileşeni Stres Analizi');
36 xlabel('von Mises Stresi (MPa)');
37 ylabel('Frekans');
38
39 % Sonuçların ekrana yazdırılması
40 fprintf('Ortalama von Mises Stresi: %.2f MPa\n', meanStressResult);
41 fprintf('von Mises Stresi Standart Sapması: %.2f MPa\n', stdStressResult);
42
43 % Paralel havuzu kapatın
44 delete(gcf('nocreate'));
45
```



Usage Scenarios

Application Examples:

- Battery Management Systems (BMS)
- By simulating the temperatures and voltages in each cell in the electric vehicle battery pack, we will find the cell with the highest temperature through parallel calculation.

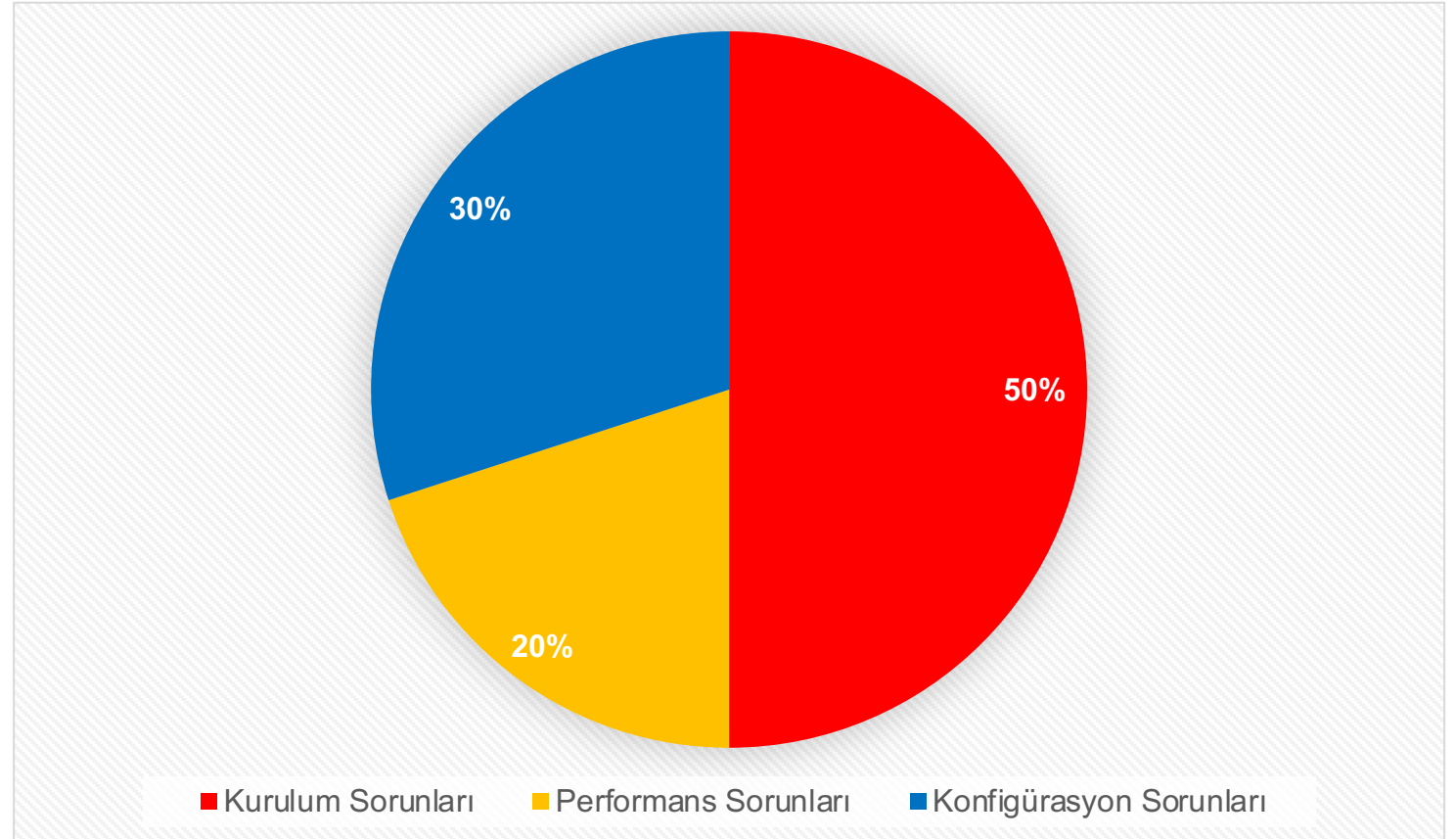
```
Editor - C:\Users\onur.ozkan\OneDrive - figes.com.tr\Masaüstü\Paralel Server Example MATLAB\BMS_Parallel_Computing.m
BMS_Parallel_Computing.m x +
1 % Script: BMS_Parallel_Computing.m
2 % Elektrikli araç batarya yönetim sistemi için paralel hesaplama örneği
3
4 % Hücre sayısını belirleyin
5 numCells = 5000;
6
7 % Hücrelerin sıcaklık ve voltaj değerlerini rastgele belirleyin
8 temperatureData = 30 + 20 * rand(1, numCells); % Sıcaklıklar 30-50°C arası
9 voltageData = 3.0 + 0.5 * rand(1, numCells); % Voltajlar 3.0-3.5V arası
10
11 % Paralel işleme havuzunu başlatın (aktif değilse)
12 if isempty(gcp('nocreate'))
13     parpool;
14 end
15
16 % En yüksek sıcaklık ve hücreyi bulmak için paralel hesaplama
17 maxTemperature = -inf;
18 maxCellIndex = -1;
19
20 % `parfor` döngüsü ile sıcaklıkları paralel olarak kontrol et
21 parfor i = 1:numCells
22     if temperatureData(i) > maxTemperature
23         maxTemperature = temperatureData(i);
24         maxCellIndex = i;
25     end
26 end
27
28 % Sonuçları görüntüleyin
29 fprintf('En yüksek sıcaklık: %.2f °C, Hücre ID: %d\n', maxTemperature, maxCellIndex);
30
31 % Paralel havuzu kapatın
32 delete(gcp);
```




Common Problems and Solutions

Graphic

- Installation Problems
- Performance Problems
- Configuration Problems



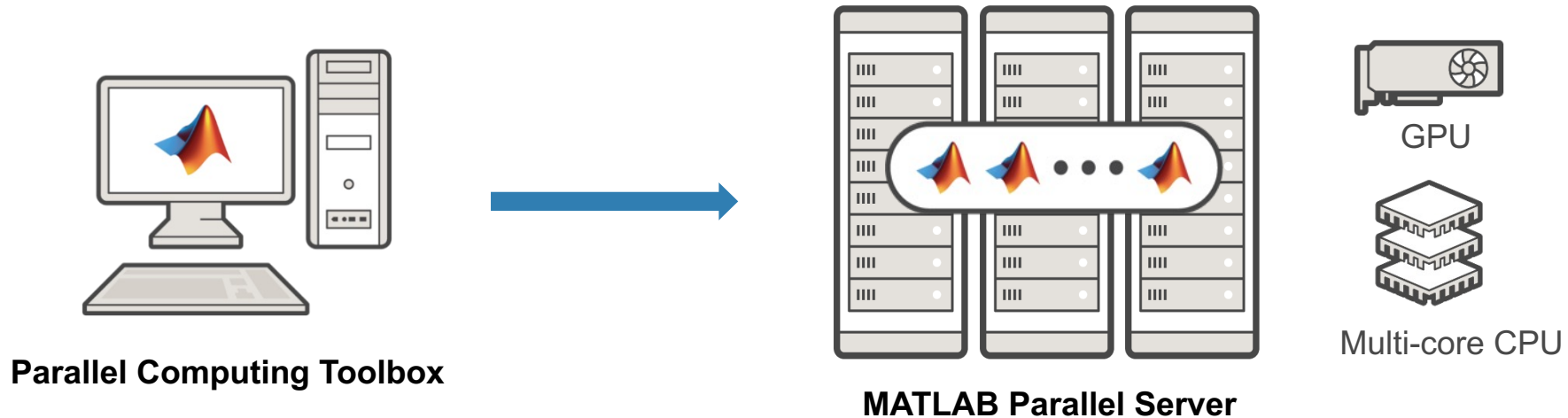
ALTAY COMPUTING SERVER SYSTEM TECHNICAL SPECIFICATIONS

	System Name	ALTAI		
	Processor type	AMD EPYC™ 7742	Intel XEON 8362	AMD EPYC 7543
	Number of compute nodes	88	30	10
	Node core count	128	64	
	Node memory amount	256 GB	512 GB	1024 GB
	Special nodes	-	<ul style="list-style-type: none"> • NVIDIA A100 80 GB PCIe • 8.8 TB scratch disk 	<ul style="list-style-type: none"> • 4 x NVIDIA A100 80GB NVLink • 12 TB scratch disk
	Performance network connection	HDR InfiniBand (200 Gbps)		HDR InfiniBand (200 Gbps)
	File System	BeeGFS, 1.6 PB		
	Operating System	Red Hat Enterprise Linux 8.5		

MATLAB Parallel Computing at Uhem



- When you are ready to run your MATLAB application beyond your desktop, **UHeM** has powerful HPC resources that allow you to run MATLAB to scale using MATLAB Parallel Server



- To get started:
 - <https://www.uhem.itu.edu.tr/MATLAB-par/matlab.html>
 - hesap@uhem.itu.edu.tr

[E-MOBİLİTE]

[YAPISAL ANALİZ]

[MOBİLİTE]

[AR-GE DERGİSİ]
EĞİTİM

[YAPAY ZEKA]

[SİMÜLASYON]

[SAVUNMA SANAYİ]

[OTOMOTİV]

[AR-GE PROJELERİ]

[TEKNOLOJİ]

*BUILDUP ACADEMY

[NÜKLEER TEKNOLOJİ]

[YEŞİL ENERJİ]

[İLERİ MÜHENDİSLİK]



YAZILIM
Thank You !

[YAPISAL ANALİZ]

[MATLAB & SIMULINK]

Contact: onur.ozkan@figes.com.tr

[YEŞİL ENERJİ]

[SİMÜLASYON]