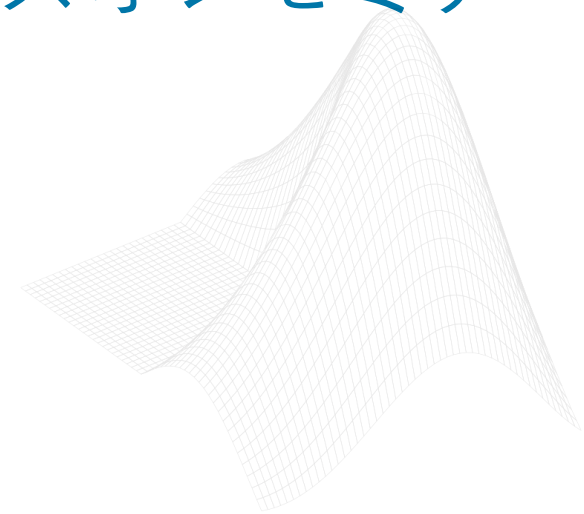
AOBA-B を利用した
カンタン MATLAB 並列処理ハンズオンセミナー



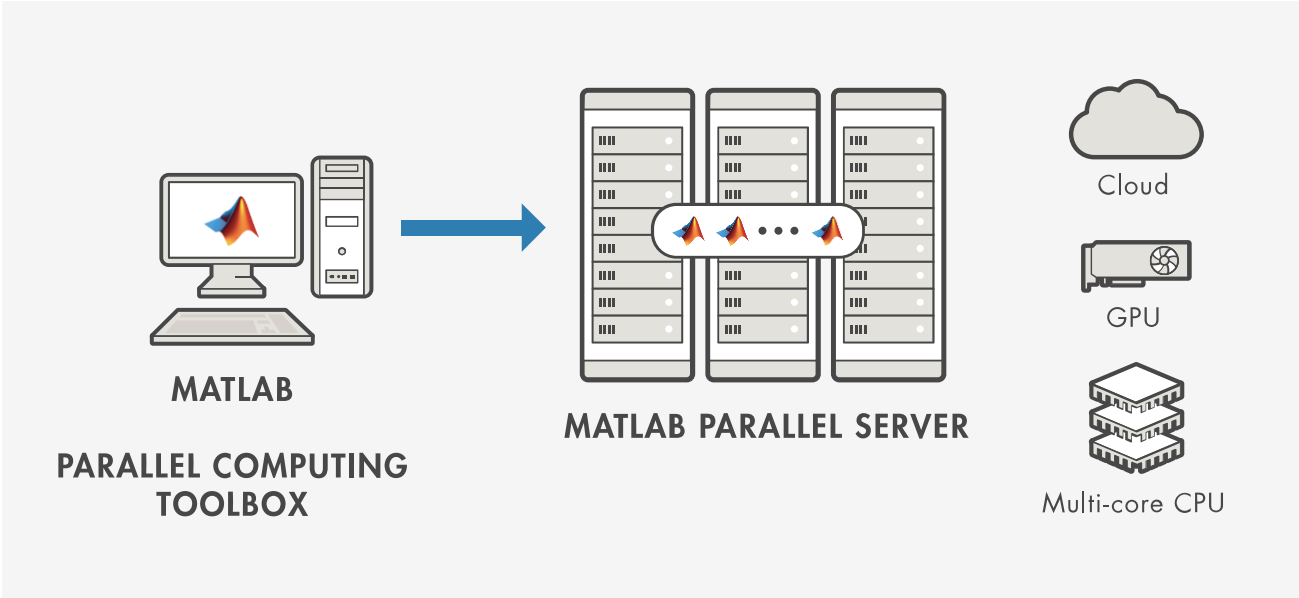
2022/11/16

MathWorks Japan
アプリケーションエンジニアリング部　データサイエンスチーム
シニアアプリケーションエンジニア
齊藤　甲次朗 (ksaito@mathworks.com)

アジェンダ

- MATLAB の並列処理
- AOBA-B での MATLAB の利用
- AOBA-B での MATLAB の演習
- 参考情報

MATLAB の並列処理



並列処理って

逐次実行

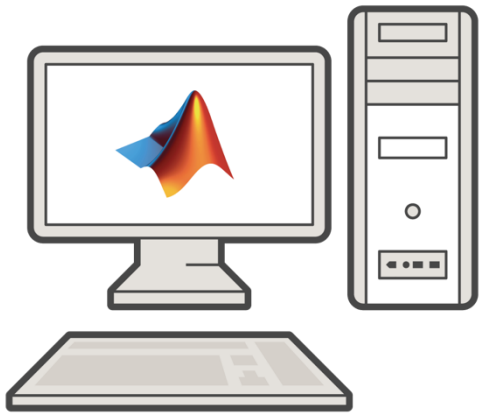
所要時間

並列実行

所要時間

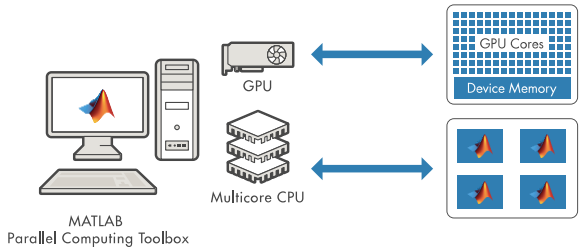
MATLAB の並列処理

MATLAB 本体のみ



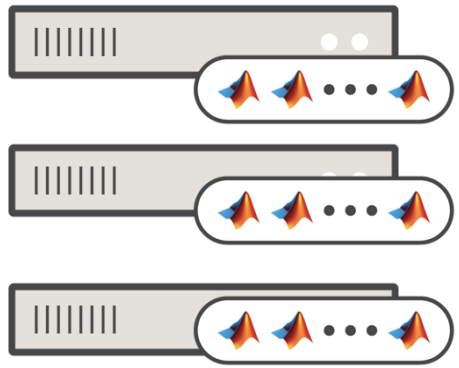
線形代数演算や要素毎の数値演算などの関数が自動的にマルチスレッドで処理されます

Parallel Computing Toolbox



並列処理に対応した MATLAB コードがマルチプロセスまたはマルチスレッド(R2020a から)、マルチGPU で処理されます

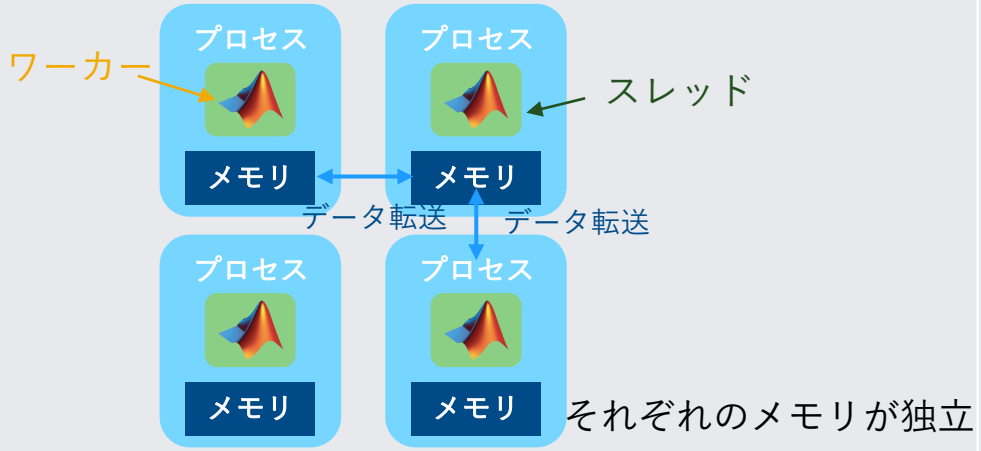
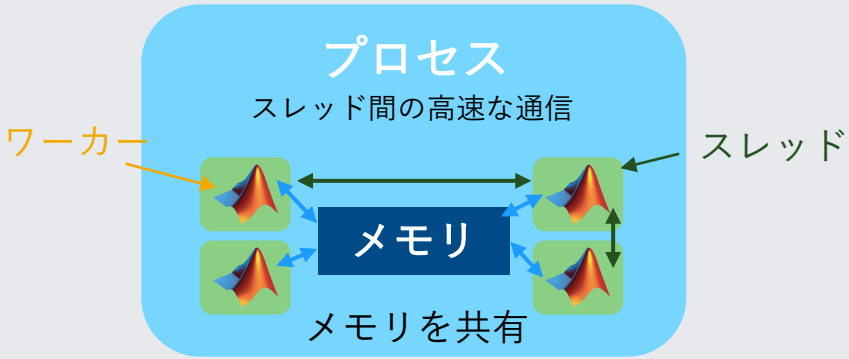
MATLAB Parallel Server



ジョブスケジューラと連携し、並列処理に対応したMATLAB コードがマルチプロセス、マルチGPU で処理されます

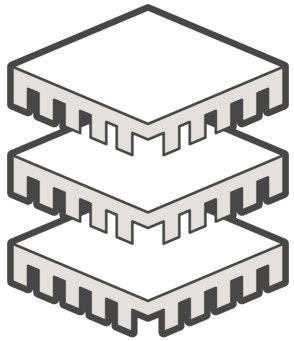
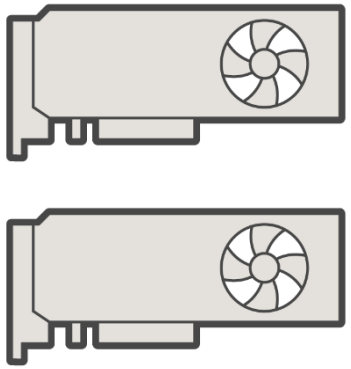
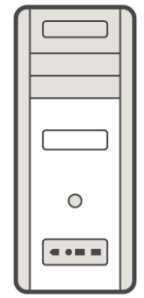
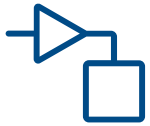
※Campus-Wide License では上記3 つの製品を全て使用できます

プロセスベースとスレッドベースの並列プール

プロセスベースの並列プール (従来どおり)	スレッドベースの並列プール (R2020a 以降)
parpool	parpool('threads')
	
特徴 • 並列言語を完全にサポート • 以前のリリースとの下位互換性 • クラッシュ時のより高いロバスト性 • 外部ライブラリをスレッドセーフにする必要なし • 複数台にまたぐ並列処理にも拡張可能	**特徴** • 並列コードがスレッドベースの環境でサポートされている場合は有効 • データ転送量が多い(>100 MB) 場合は有効 • メモリ使用量の削減、スケジューリングの高速化、データ転送の低減 • 1台のみの並列処理をサポート

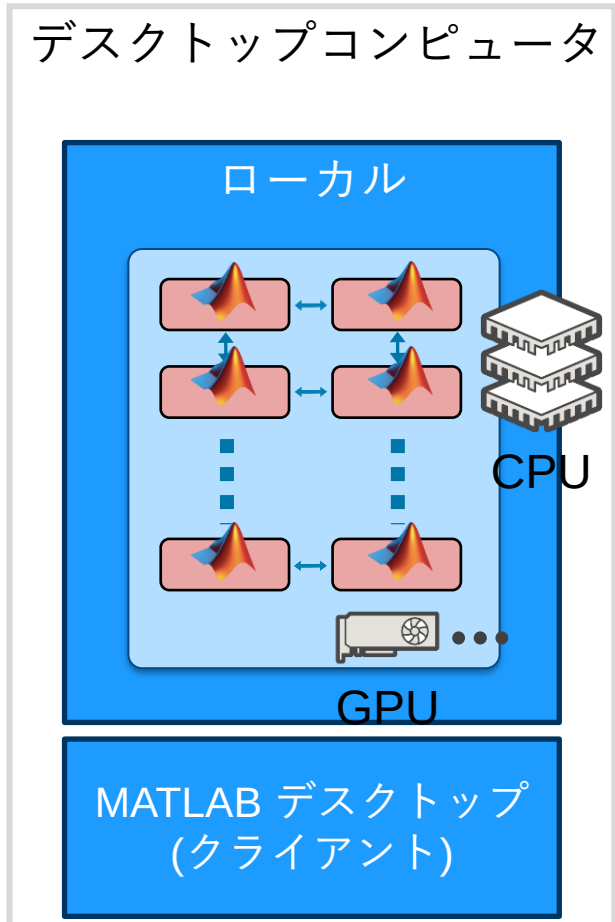
https://jp.mathworks.com/help/parallel-computing/choose-between-thread-based-and-process-based-environments.html

並列処理の種類

CPU	GPU	マシンの台数
		1 台 
主なアプリケーション  機械学習  最適化  シミュレーション	主なアプリケーション  画像処理  ディープラーニング	 複数台

Parallel Computing Toolbox
1台のマシンでのMATLAB関数及びSimulinkモデルの並列分散実行

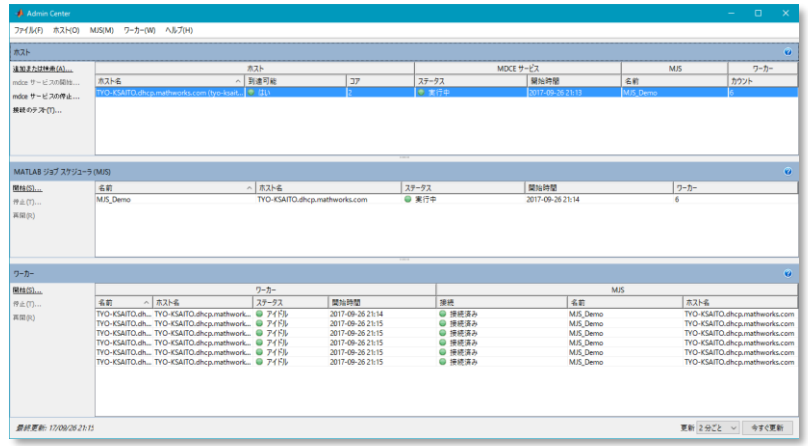
- ローカルでの MATLAB & Simulink プロダクトファミリと連携した並列処理による演算の高速化
  - 並列 for ループ
  - 並列アルゴリズムの使用
  - GPU 演算
  - バッチジョブの並列実行
- ジョブおよびタスクの制御
- ビッグデータ解析の分割処理
  - メモリに収まらないデータの扱い
  - スパース分散行列の作成
- 対話的な並列コマンドウィンドウ環境
  - pmode



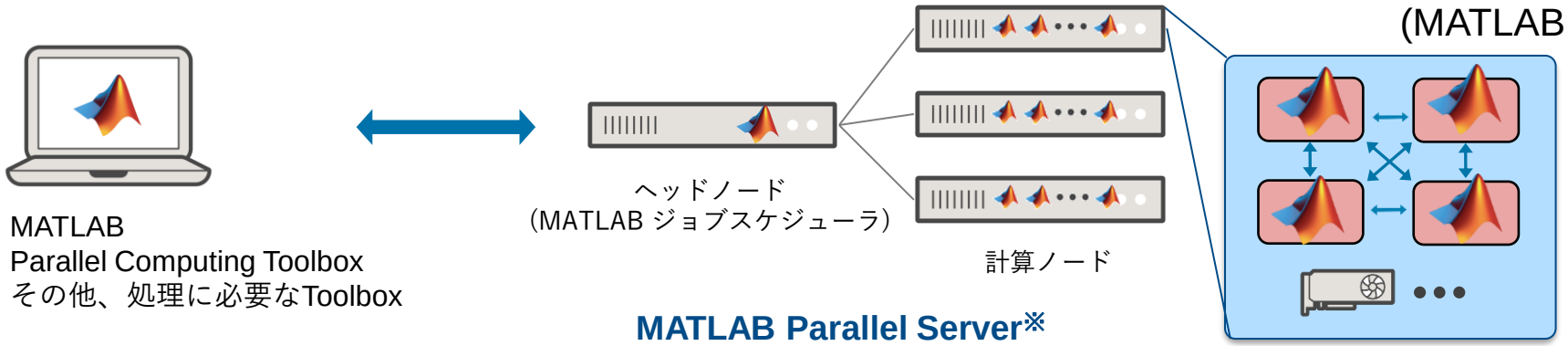
MATLAB Parallel Server

クラスター環境を利用したMATLAB 関数及びSimulink モデルの並列分散実行

- **クラスターマシン**のマルチCPU・GPU での並列処理
- 簡単に使えるスケジューラ(MATLAB ジョブスケジューラ)を内蔵
- サードパーティスケジューラとの連携
- Hadoop®・Spark®との連携によるビッグデータ処理
- クライアントの製品構成に依存しない計算環境を提供



GUI による簡単な操作
(MATLAB ジョブスケジューラ)



- ✓ MATLAB
- ✓ Parallel Computing Toolbox
- ✓ その他、処理に必要なToolbox

MATLAB Parallel Server※

Hadoop/Spark との連携

※R2018bまでの名称はMATLAB Distributed Computing Server (MDCS)

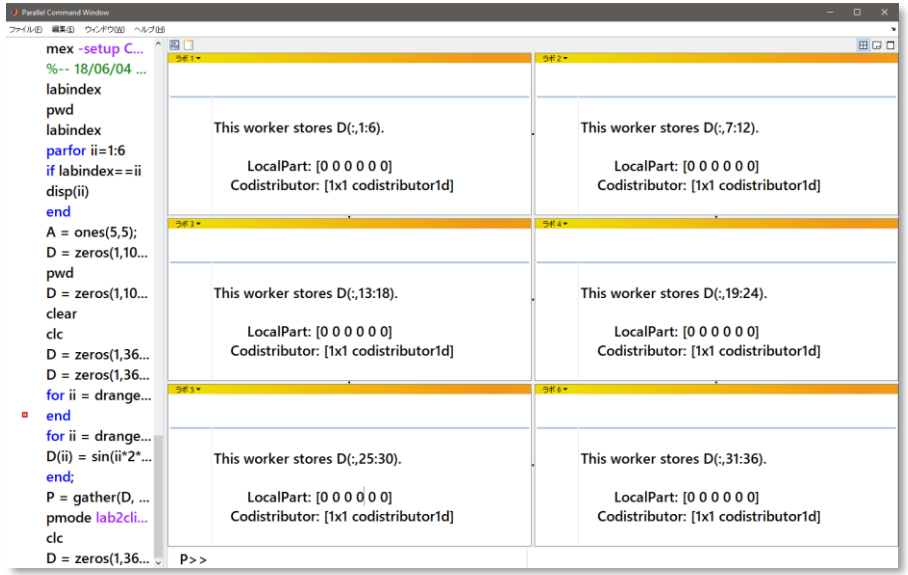
並列処理の関数

1. インタラクティブジョブ
   - parfor: 汎用的な並列処理
   - spmd: データを分割して同じ処理を実行
     (single process multiple data)
   - pmode: 対話的GUI を使った並列処理
   - parsim: 並列シミュレーション(R2017a以降)

parfor の例

```
n = 200;
A = 500;
a = zeros(n);
parfor i = 1:n
  a(i) = max(abs(eig(rand(A))));
end
```

pmode による対話的な並列処理



spmd の例 (円周率の計算)

```
spmd
  a = (labindex - 1)/numlabs;
  b = labindex/numlabs;
  fprintf('Subinterval: [%-4g, %-4g]¥n', a, b);
end

spmd
  myIntegral = integral(@pctdemo_aux_quadpi, a, b);
  fprintf('Subinterval: [%-4g, %-4g] Integral: %4g¥n', ...
  a, b, myIntegral);
end

% 結果の合計
spmd
  piApprox = gplus(myIntegral);
end
```

並列処理の関数

2. オフロードジョブ
   - バッチ:
     クラスターでMATLAB スクリプトを実行
   - 独立ジョブ:
     クラスターの1ワーカーで実行するジョブ
   - 通信ジョブ:
     クラスターの複数ワーカーで実行するジョブ

```
% クラスターの作成
c = parcluster();
% バッチ処理の実行
j = batch(c,@rand,1,{10,10}, 'CaptureDiary', ...
  true, 'CurrentFolder', '.');
% ジョブの終了待ち
wait(j)
% ジョブ結果の回収
out = fetchOutputs(j);
% ジョブの消去
delete(i)
```

バッチの例

```
% クラスターの作成
c = parcluster;
% ジョブの作成
j = createCommunicatingJob(c,'Type', 'pool');
% タスクの作成
createTask(j, @myFunction, 1, {100});
% ジョブの投入
submit(j);
% ジョブの終了待ち
wait(j)
% ジョブ結果の回収
out = fetchOutputs(j)
% ジョブの消去
delete(j)

%% カスタム関数
function result = myFunction(N)
  result = 0;
  parfor ii=1:N
    result = result + max(eig(rand(ii)));
  end
end
```

通信ジョブの例

```
% クラスターの作成
c = parcluster
% ジョブの作成
j = createJob(c);
% タスクの作成
for ii = 1:10
createTask(j,@rand,1,{10});
end
% ジョブの投入
submit(j);
% ジョブの終了待ち
wait(j);
% ジョブ結果の回収
out = fetchOutputs(j);
% ジョブの消去
delete(j)
```

独立ジョブの例

Toolbox の関数での並列処理

関数のオプションで指定

Optimization Toolbox

```
optimoptions(..., 'UseParallel', true)
```

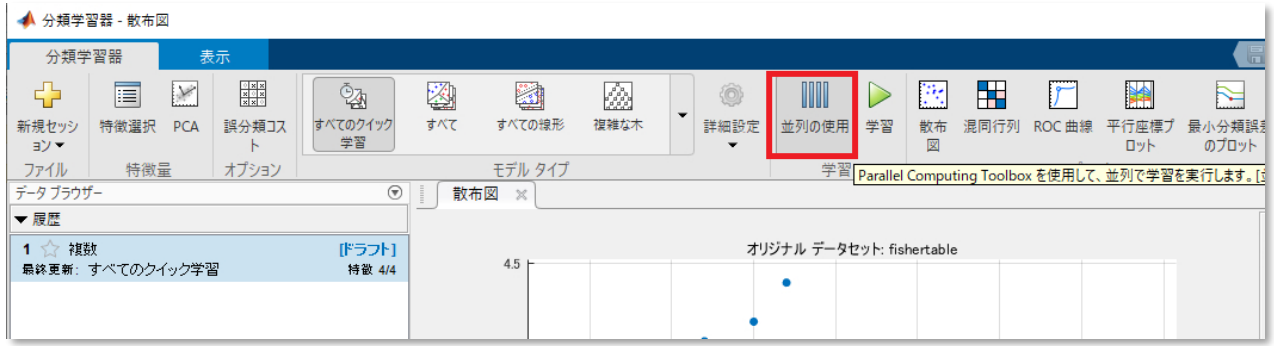
Deep Learning Toolbox

```
trainingOptions(..., 'ExecutionEnvironment', 'parallel')
```

既定のクラスタープロファイルの複数GPU 、またはCPU を使用

アプリのオプションで変更

Statistics and Machine Learning Toolbox



Simulink の並列シミュレーションコマンドで

```
model = 'vdp';
in = Simulink.SimulationInput(model);
out = parsim(in)
%out = batchsim(in)
```

GPU 処理

gpuArrayの使用

C言語 + CUDA
数百～数千行

```
#include "cufft.h"
#include "cuda_runtime.h"

void pack_r2c(cufftComplex  *output_float,
              double *input_re,
              int N
{
 /* Allocate array                    */
  cudaMalloc( (voi
&rhs_complex_d,size                    plex)*N*M);

  /* Copy input arr                   evice */
  cudaMemcpy( rhs_c                    nput_single,
sizeof(cufftCo                         pyHostToDevice);
```

MATLABでの記述

```
% GPU上で配列を作成
GX = gpuArray(X);

%  GPU上での演算
GY = fft2(GX);

%  ローカルワークスペースへ転送
Y = gather(GY);
```

関数のオプションでGPU 配列を作成

```
ones(10,1,'gpuArray');
```

GPU 配列をサポートするToolbox

ツールボックス名	gpuArray をサポートする関数のリスト
MATLAB	gpuArray をサポートする関数
Statistics and Machine Learning Toolbox™	gpuArray をサポートする関数 (Statistics and Machine Learning Toolbox)
Image Processing Toolbox™	gpuArray をサポートする関数 (Image Processing Toolbox)
Deep Learning Toolbox™	gpuArray をサポートする関数 (Deep Learning Toolbox) *(GPU を使用した深層学習も参照)
Computer Vision Toolbox™	gpuArray をサポートする関数 (Computer Vision Toolbox)
Communications Toolbox™	gpuArray をサポートする関数 (Communications Toolbox)
Signal Processing Toolbox™	gpuArray をサポートする関数 (Signal Processing Toolbox)
Audio Toolbox™	gpuArray をサポートする関数 (Audio Toolbox)
Wavelet Toolbox™	gpuArray をサポートする関数 (Wavelet Toolbox)
Curve Fitting Toolbox™	gpuArray をサポートする関数 (Curve Fitting Toolbox)

ディープラーニングは自動的にGPU で実行

- `trainNetwork` (Deep Learning Toolbox)
- `predict` (Deep Learning Toolbox)
- `predictAndUpdateState` (Deep Learning Toolbox)
- `classify` (Deep Learning Toolbox)
- `classifyAndUpdateState` (Deep Learning Toolbox)
- `activations` (Deep Learning Toolbox)

関数のオプションでも指定可能

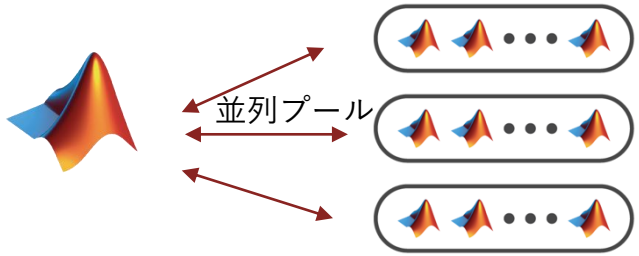
```
trainingOptions(..., 'ExecutionEnvironment', 'gpu') 1台で複数GPUを使用
trainingOptions(..., 'ExecutionEnvironment', 'multi-gpu')
```

https://jp.mathworks.com/help/parallel-computing/run-matlab-functions-on-a-gpu.html

クラスターサーバーへのジョブの投入の方法

インタラクティブジョブ

```
n = 200;
A = 500;
a = zeros(1,n);
parfor i = 1:n
  a(i) = max(abs(eig(rand(A))));
end
```



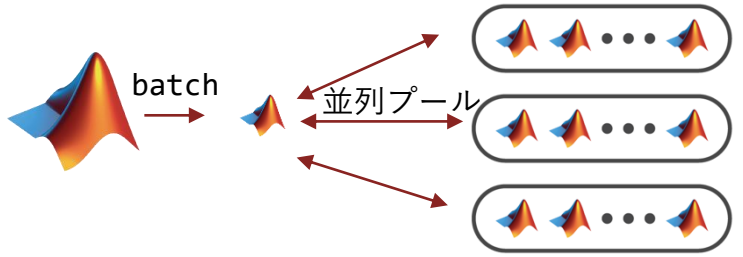
並列処理のワーカー

オフロードジョブ

myparfor.m

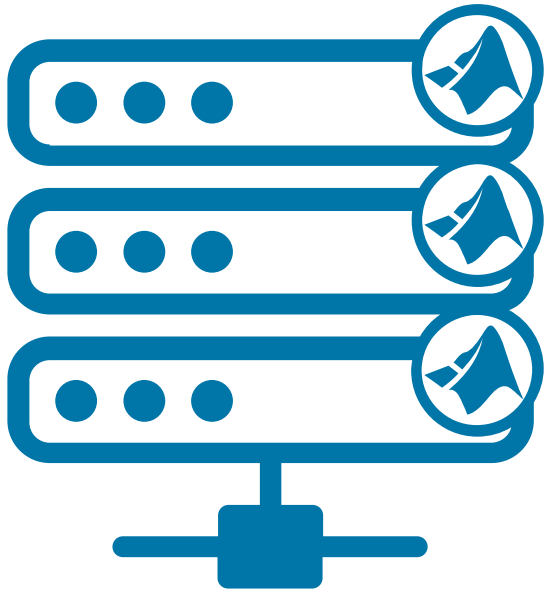
```
n = 200;
A = 500;
a = zeros(1,n);
parfor i = 1:n
  a(i) = max(abs(eig(rand(A))));
end
```

```
job = batch('myparfor', 'Pool', 3);
wait(job)
out = fetchOutputs(job);
out = out{1};
```



並列処理のワーカー

AOBA-B でのMATLAB の利用



AOBA-B でのMATLAB の利用

東北大学サイバーサイエンスセンター
大規模科学計算システム (AOBA)

サブシステム AOBA-A
(SX-Aurora TSUBASA)

クラウドサービス AOBA-C
(SX-Aurora TSUBASA)

サブシステム AOBA-B

LX 406Rz-2
(AMD EPYC プロセッサ)が
68 ノード

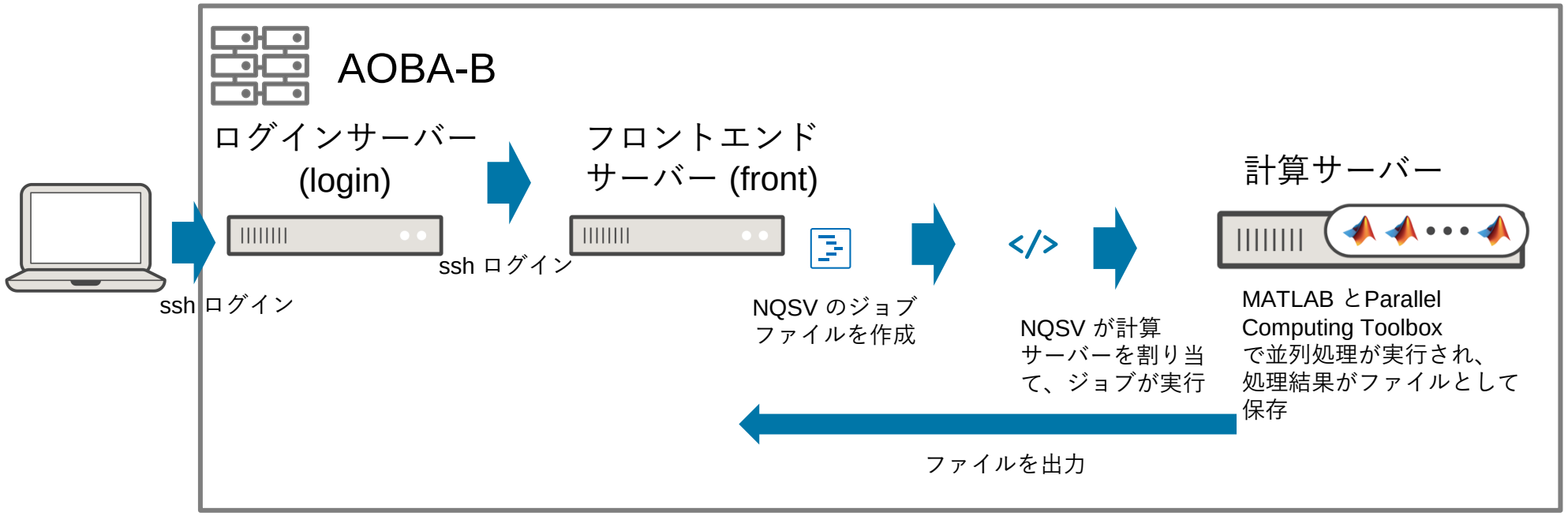
東北大学サイバーサイエンスセンターのHPより転載
https://www.ss.cc.tohoku.ac.jp/lx406rz-2/

AOBA-B のマシンで**MATLAB** 及び **MATLAB Parallel Server** が使えます

ジョブスケジューラはNEC Network Queuing System V (NQSV)

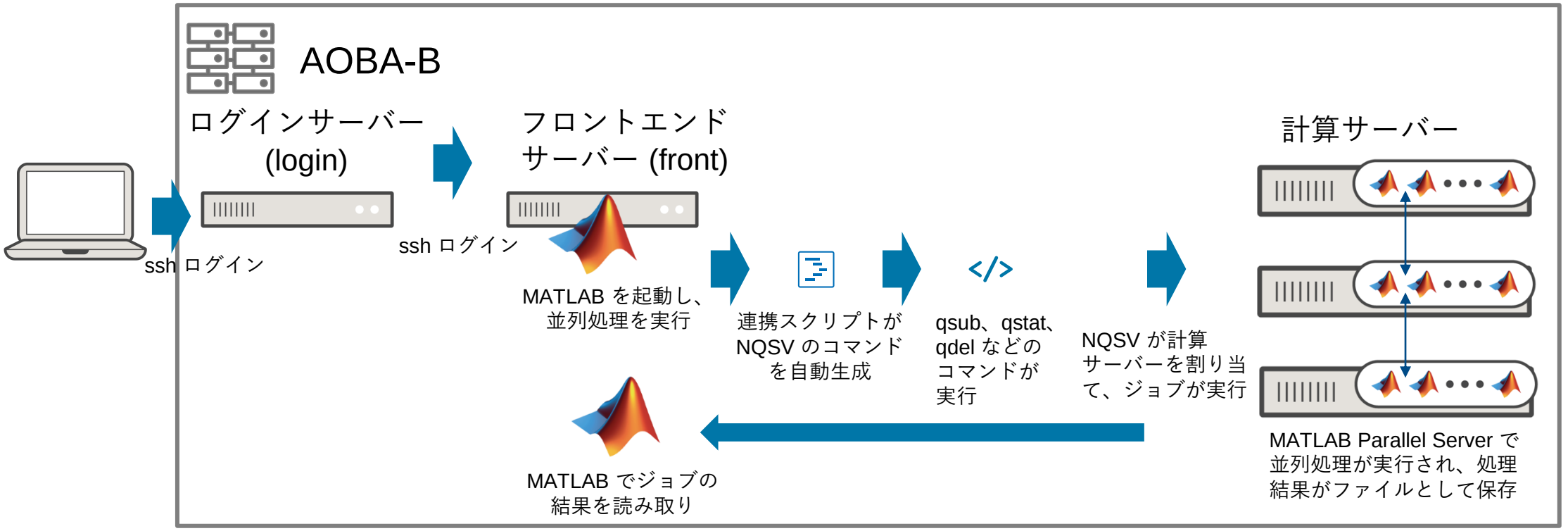
MATLAB とMATLAB Parallel Server とも R2020a、R2020b、R2021a、R2022a が利用可能 (2022年11月現在)

AOBA-B でのMATLAB の利用
1 台ずつの処理 (1 ノード内実行)



演習1 で体験します

AOBA-B でのMATLAB の利用
複数台にまたがる処理



演習2 で体験します

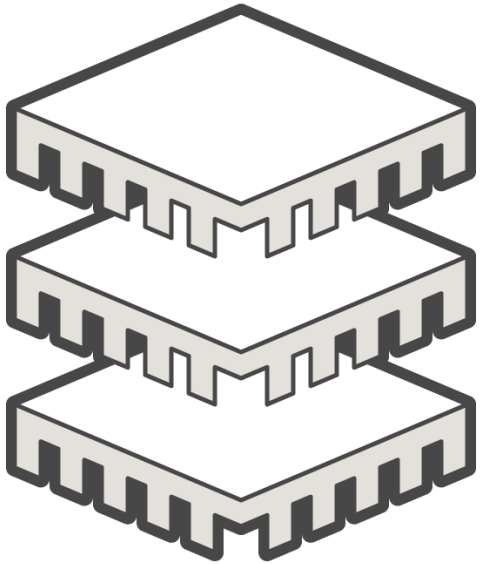
コマンドライン (CUI) でのMATLAB の起動

- MATLAB はコマンドライン(CUI) と画面で操作するGUI のどちらでも利用できます
- GUI で使用するにはX11 やVNC での画面転送が必要になります*1)

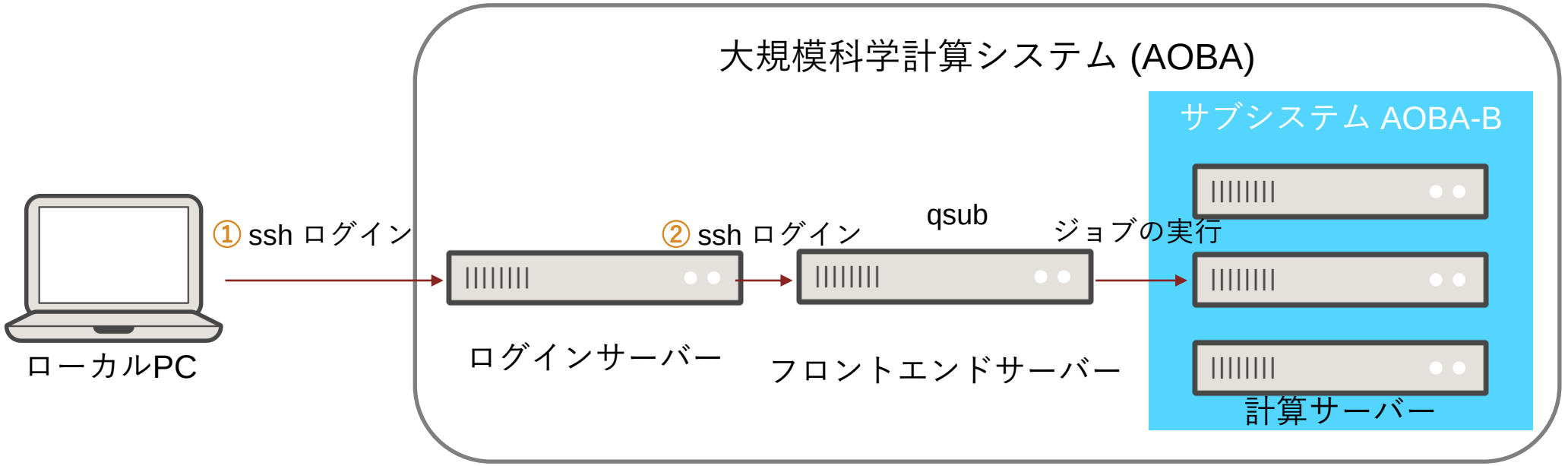
- この講習会ではコマンドラインでの演習とします
  - リモートだと画面転送のタイムラグがあるため

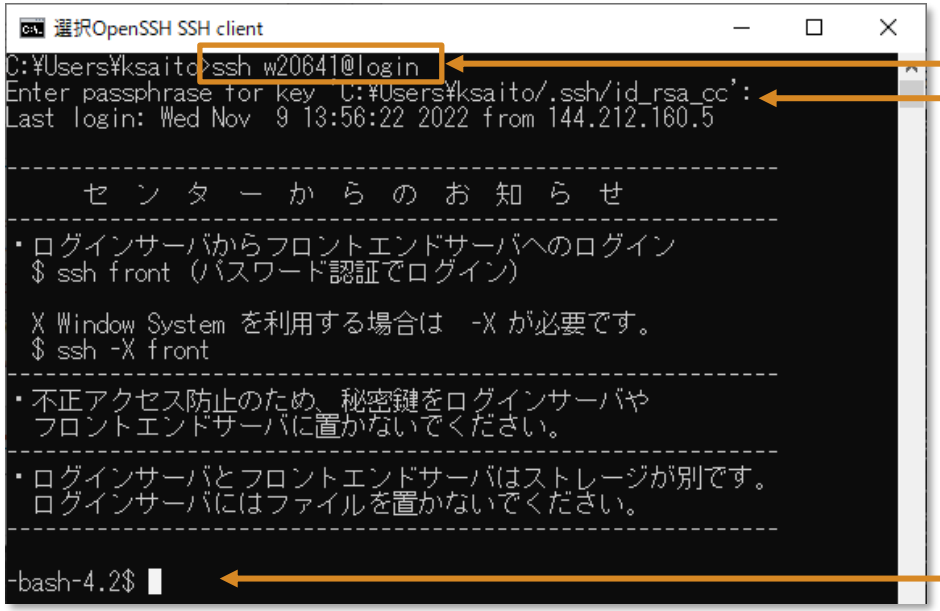
*1) 画面転送についてはアプリケーションサービスの紹介 の「2.4が画面転送する場合」参考

AOBA-B でのMATLAB の演習



AOBA-B へのログイン





① ssh ユーザー名@login

ssh 公開鍵のパスフレーズを入力

② ログインサーバーにログインした状態で
ssh front を実行し、アカウントのパスワードを入力

フロントエンドサーバーへのログイン成功

ログインサーバーへのログイン成功

事前準備
ファイルの転送

- AOBA-B のサーバーにscp でファイルを転送します

ローカルPC のターミナルで実行

```
scp ./MATLAB_HandsOn.zip front:~/
```

```
$ scp ./MATLAB_HandsOn.zip front:~/
Enter passphrase for key '/c/Users/ksaito/.ssh/id_rsa_cc':
Enter passphrase for key '/c/Users/ksaito/.ssh/id_rsa_cc':
MATLAB_HandsOn.zip                                    100%  244KB   1.5MB/s   00:00
```

- AOBA-B のサーバーにssh ログインしてzip を解凍します

ローカルPC のターミナルで実行

```
ssh front
```

フロントエンドサーバーのターミナルで実行

```
unzip ./MATLAB_HandsOn.zip
```

```
[w20641@front2 ~]$ unzip ./MATLAB_HandsOn.zip
Archive:  ./MATLAB_HandsOn.zip
   creating: MATLAB_HandsOn/
   creating: MATLAB_HandsOn/ex1/
   creating: MATLAB_HandsOn/ex2/
  inflating: MATLAB_HandsOn/ex2/monteCarloFunc.m
  inflating: MATLAB_HandsOn/ex2/monteCarloSample.m
  inflating: MATLAB_HandsOn/ex2/monteCarloSampleLiveScript.mlx
  inflating: MATLAB_HandsOn/ex2/monteCarloSampleLiveScript.pdf
  inflating: MATLAB_HandsOn/ex2/ReadMe.txt
  inflating: MATLAB_HandsOn/ex2/updateScatter.m
```

【演習１】１台ずつの処理 (1 ノード内実行)

- MATLAB スクリプトの作成

手動で作成

qsub →

MATLAB とParallel Computing Toolbox

ex1.m

```
p=parpool('local',4);
n = 500;
A = 100;
a = zeros(1,n);
parfor k = 1:n
a(k) = max(abs(eig(rand(A))));
end
delete(gcp('nocreate'))
```

- ジョブファイルの作成 *1)

ex1job.nqs

```
#!/bin/sh
matlab -batch "ex1"
```

ex1run.sh

```
#!/bin/sh#
PBS -l elapstim_req=00:10:00   # 最大経過時間
#PBS -q lx_edu -b 1   # 投入キュー名と利用ノード数1を指定
#PBS -jo -N lx_edu # 出力ファイル名を指定
cd $PBS_O_WORKDIR # 作業ディレクトリへの移動コマンド
ex1job.nqs # nqsに書かれたMATLABのジョブを実行
```

*1) サイバーサイエンスセンター「アプリケーションサービス」及び「ジョブの実行方法」を参考

【演習１】１台ずつの処理 (1 ノード内実行)

- 実行権限の付与

  フロントエンドサーバーのターミナルで実行

```
chmod u+x ex1*
```

- ジョブ(バッチリクエスト)の投入

```
qsub ex1run.sh
```

```
[w20641@front2 ex1]$ qsub ./ex1run.sh
プロジェクトコード: center01 にリクエストを投入します
Request 393810.job submitted to queue: lx_edu.
```

- ステータスの確認

```
reqstat
```

```
[w20641@front2 ex1]$ reqstat
RequestID   RequestName  User    PJCode     Que        Node ElapseLimit STT StartTime   Memory1  Memory2  ElapseTime  NodeTime
----------- ------------ ------ ---------- ---------- ---- ----------- --- ----------- -------- -------- ----------- -----------
393810.job  lx_edu       w20641 center01   lx_edu        1 00+00:10:00 RUN 11-16 10:19    1.80G        - 00+00:02:59 00+00:02:59
----------- ------------ ------ ---------- ---------- ---- ----------- --- ----------- -------- -------- ----------- -----------
```

- 結果の確認

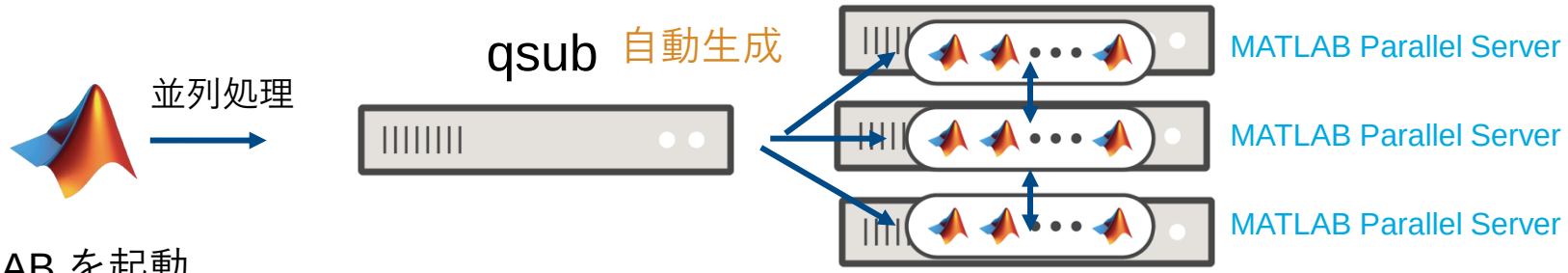
```
cat lx*
```

```
[w20641@front2 ex1]$ ls -l
合計 302
-rwxr--r-- 1 w20641 ccusers    135 11月  16 09:57 ex1.m
-rwxr--r-- 1 w20641 ccusers     30 11月  16 09:53 ex1job.nqs
-rwxr--r-- 1 w20641 ccusers    284 11月  16 09:47 ex1run.sh
-rw-r--r-- 1 w20641 ccusers    176 11月  16 10:22 lx_edu.o393810
```

```
[w20641@front2 ex1]$ cat lx*
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4).
Parallel pool using the 'local' profile is shutting down.
```

【演習２】複数台にまたがる処理

Parallel Server を使用するための設定



- MATLAB を起動

  フロントエンドサーバーのターミナルで実行

```
matlab -nodesktop
```

- クラスタープロファイルの設定

  MATLAB のコマンドウィンドウで実行

```
>> addpath('/mnt/stfs/ap/aoba-b_shared')
>> configCluster
```

- プロファイルの確認

```
>> c = parcluster
```

```
>> c = parcluster

c =

 Generic Cluster

    Properties:

                    Profile: aoba-b local R2022a
                   Modified: false
                       Host: front1
                 NumWorkers: 128
                 NumThreads: 1

          JobStorageLocation: /uhome/w20641/MdcsDataLocation/aoba-b/front1/R2022a/local
           ClusterMatlabRoot: /mnt/stfs/ap/ParallelServer/MATLAB.R2022a
             OperatingSystem: unix

    RequiresOnlineLicensing: false
      PluginScriptsLocation: /mnt/stfs/ap/aoba-b_shared/IntegrationScripts/aoba-b
       AdditionalProperties: List properties

     Associated Jobs:

            Number Pending: 0
             Number Queued: 0
            Number Running: 0
           Number Finished: 0
```

- キューの変更

```
>> c.AdditionalProperties.QueueName = 'lx_edu';
```

通常のキュー「lx」から、
講習会用のキューに変更します

サイバーサイエンスセンター「AOBA-B でのMATLAB Parallel Serverの使い方.pdf」より引用

【演習 2】複数台にまたがる処理
モンテカルロ・シミュレーション

モンテカルロ法 (モンテカルロ・シミュレーション)

- **乱数**を生成してシミュレーションや数値計算をおこなう手法の総称
- カジノで有名なモナコ公国の地名「モンテカルロ」から

- MATLAB スクリプトの作成

monteCarloSample.m

```
%% モンテカルロ法による円周率の計算
iteration = 10000;
withPlot = false;
[myPi, elapsedTime] = monteCarloFunc(iteration, withPlot);

fprintf('Estimated Pi: %2.7f¥n', myPi);
fprintf('Elapsed time: %8.2f seconds¥n', elapsedTime);
```

並列for 文で計算

```
parfor n = 1:iter
    x = R * rand(1);
    y = R * rand(1);
    if x^2 + y^2 < R^2
        count(n, 1) = 1;
    end
end
```

乱数

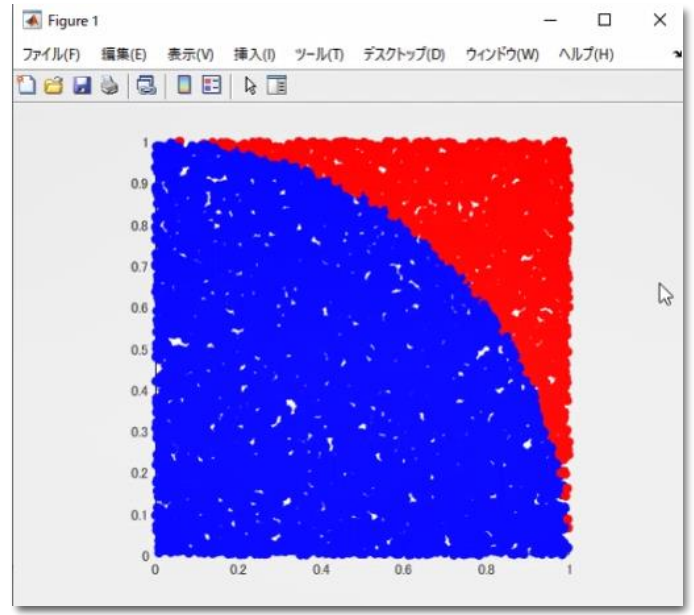
【演習２】複数台にまたがる処理
モンテカルロ・シミュレーション

- インタラクティブジョブで投入
  MATLAB のコマンドウィンドウで実行

```
>> monteCarloSample
```

```
[w20641@front2 ex2]$ matlab -nodesktop
MATLAB is selecting SOFTWARE OPENGL rendering.

                            < M A T L A B (R) >
                  Copyright 1984-2022 The MathWorks, Inc.
                 R2022a (9.12.0.1884302) 64-bit (glnxa64)
                              February 16, 2022


To get started, type doc.
For product information, visit www.mathworks.com.

>> ls
ReadMe.txt          monteCarloSample.m              monteCarloSampleLiveScript.pdf
monteCarloFunc.m    monteCarloSampleLiveScript.mlx  updateScatter.m

>> monteCarloSample
Starting parallel pool (parpool) using the 'aoba-b local R2022a' profile ...

additionalSubmitArgs =

    '-b 1 -T intmpi -q lx -l elapstim_req=24:00:00 -r n'


Estimated Pi: 3.1356000
Elapsed time:    15.16 seconds
```

← 円周率の近似値が得られました



GUI ありで実行した場合

参考情報

- **参考情報**
  - MATLAB Parallel Server の製品について
    https://jp.mathworks.com/products/matlab-parallel-server.html
  - MATLAB の並列処理について
    https://jp.mathworks.com/help/parallel-computing/getting-started-with-parallel-computing-toolbox.html
  - MATLAB のバッチ処理のサンプル
    https://jp.mathworks.com/help/parallel-computing/batch-processing.html
- **問い合わせ窓口**
  - MathWorks サポート窓口
    https://jp.mathworks.com/support/contact_us.html
  - コミュニティ Q&A サイト
    https://jp.mathworks.com/matlabcentral/answers/