JEMARO

With the support of the
Erasmus+ Programme
of the European Union

Parallelization in MATLAB

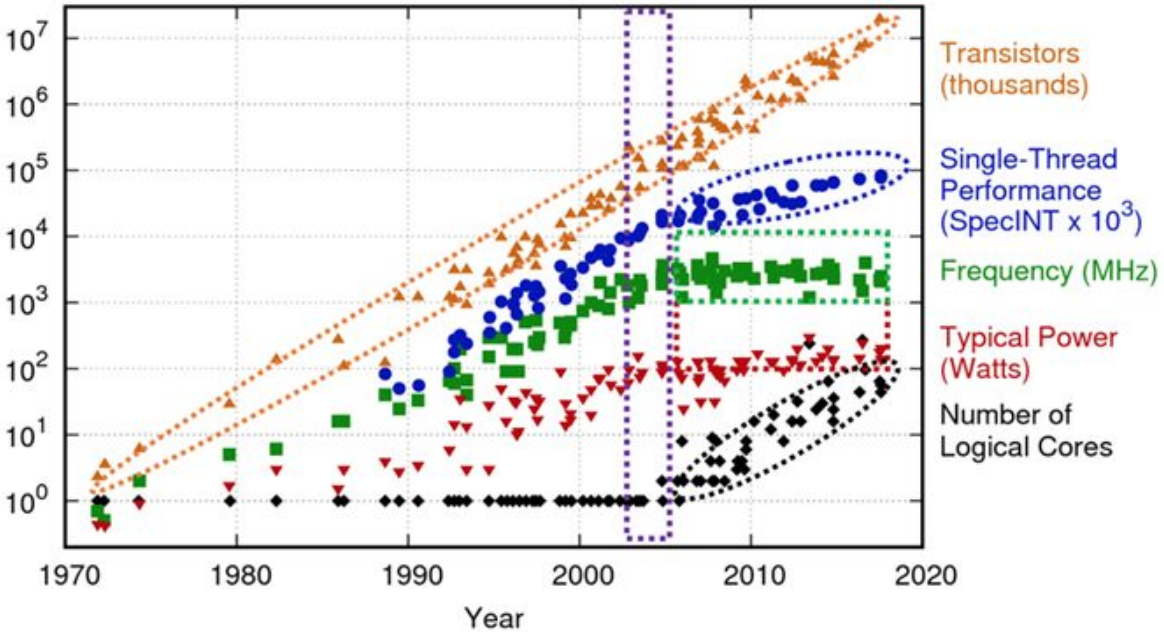
Ammar AlKhaled
Litong Huang

Supervisor: Prof. Jasmin Jahic

Course: Concurrent Computing in Robotics
Team number: Group 3
Presentation 2

Table of Contents

1.Introduction



https://education.dellemc.com/content/dam/dell-emc/documents/en-us/2019KS_Yellin-Saving_The_Future_of_Moores_Law.pdf.

Moore's Law
Dennard's Scaling

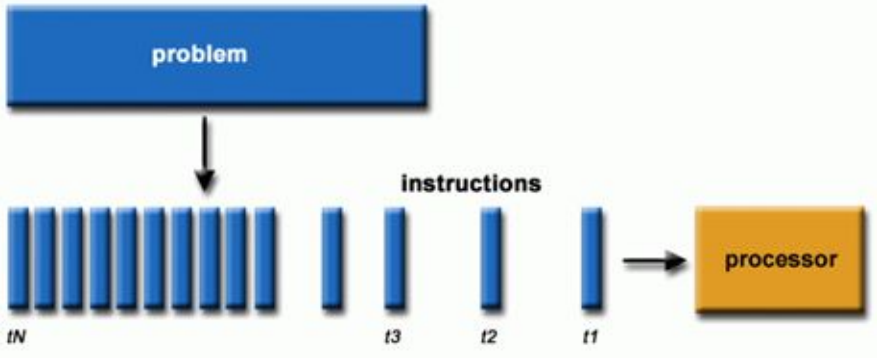
Solution

Shifting to

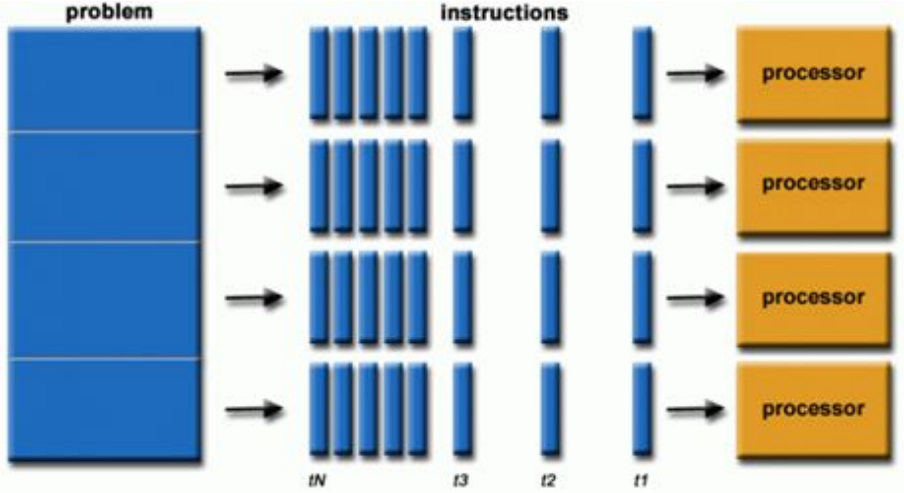
Multicores
Parallelism

1.Introduction

Sequential Computing



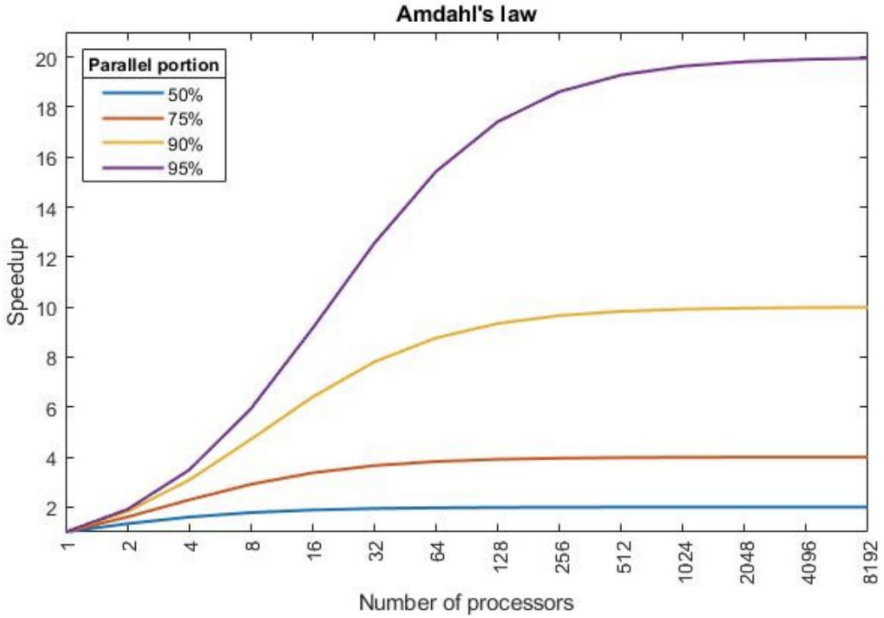
Parallel Computing



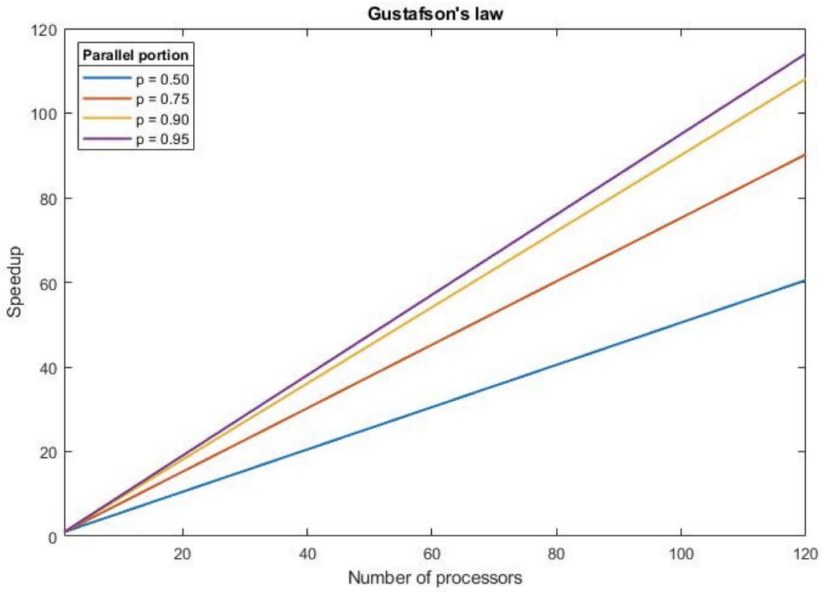
https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial

1.Introduction

Amdahl's Law: Speed-Up



Gustafson's Law: Scalability



http://dspace.unive.it/bitstream/handle/10579/17783/857086-1255168.pdf?sequence=2

2.Parallelization in MATLAB

parpool: Create a parallel pool of workers to do parallel computing.

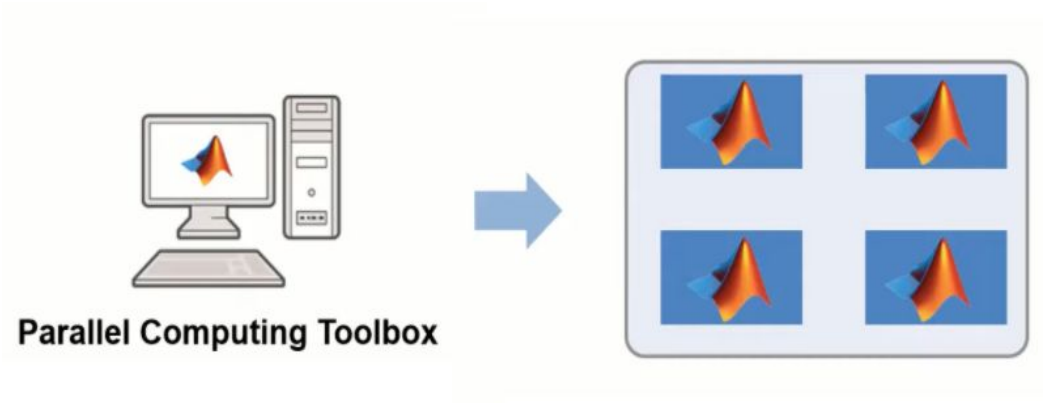
parpool('Environment', WorkersNumber)

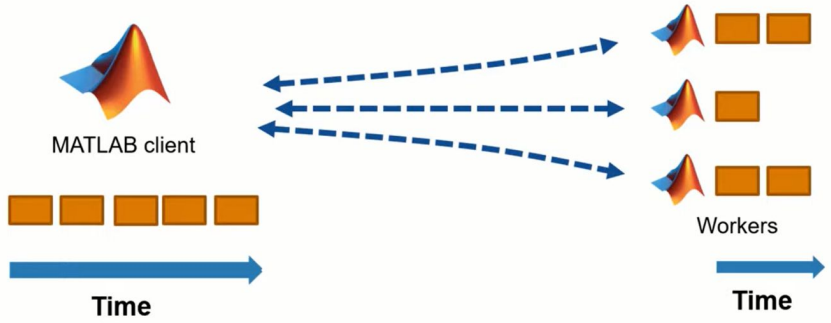
delete(gcp('nocreate')): delete previously created parallel pool

parfor: Execute for loop in parallel by workers in parallel pool.

The general form of a parfor statement is:

```
parfor loopvar =initval:endval
   <statements>
END
```





https://www.mathworks.com/help/parallel-computing/choose-between-thread-based-and-process-based-environments.html#:~:text=Process%2Dbased%20environments%20have%20the,MATLAB%20client%20does%20not%20crash.

3. Sequential Vs Parallel Computing

Example 1.1

```
primeNums = primes(uint64(2^17));
compositeNums = primeNums.*primeNums(randperm(numel(primeNums)));
factors = zeros(numel(primeNums),2);


  %====Normal calculation (no parallel computing)===%
  delete(gcp('nocreate'));
  tic;
for i = 1:numel(compositeNums)
    factors(i,:)=factor(compositeNums(i));
end
  toc

  %============Parallel computing (using 4 cores) ============%
  % using parfor for multicore in thread-based environment
  delete(gcp('nocreate'));
  parpool('local',4);

  tic;
parfor i = 1:numel(compositeNums)
    factors(i,:) = factor(compositeNums(i));
end
  toc
```

Sequential	*Parallel*
5.097076 seconds.	3.775082 seconds.

4. Performance Speed-Up

Example 1.2

1,2,4 cores

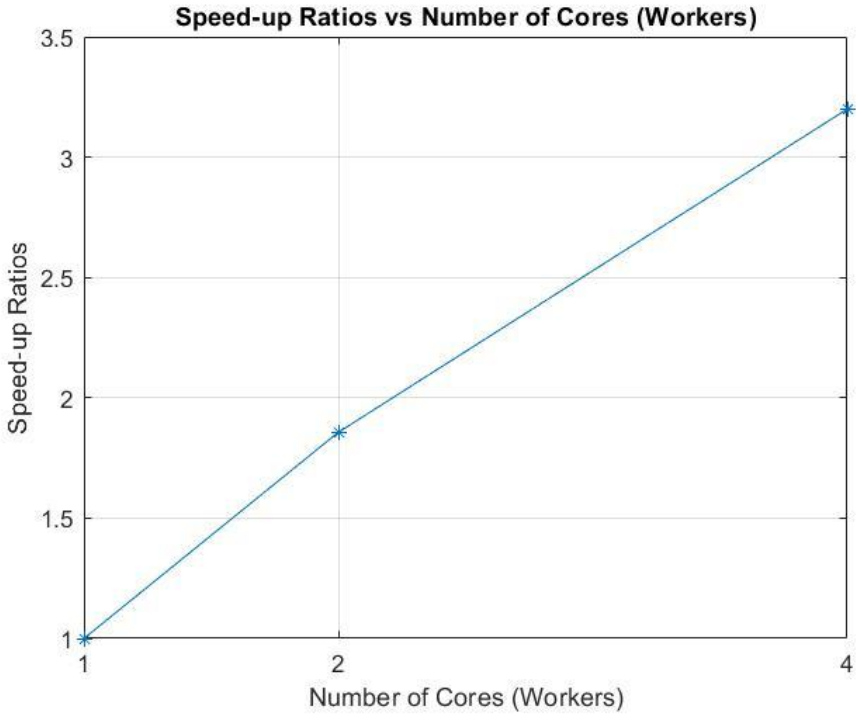
```
delete(gcp('nocreate'));
parpool('local',4);

tic;
parfor i = 1:numel(compositeNums)
    factors(i,:) = factor(compositeNums(i));
end

toc
```

```
Parallel pool using the 'local' profile is shutting down.
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 1).
Parallel pool using the 'local' profile is shutting down.
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 2).
Parallel pool using the 'local' profile is shutting down.
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4).
```

1 core:	2 cores:	4 cores:
51.2585	27.6084	16.0189



4. Performance Speed-Up

Example 2.1

$$A = \begin{bmatrix} 2 & -5 & 3 \\ 0 & 7 & -2 \\ -1 & 4 & 1 \end{bmatrix} \qquad \det(A - \lambda I) = 0$$

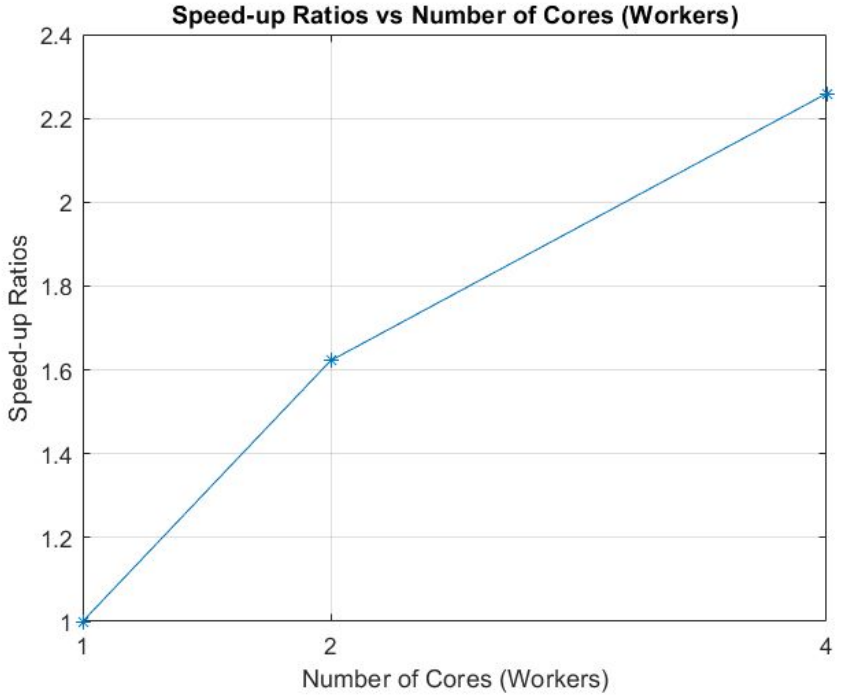
```
L = 1000;
c = [];

delete(gcp('nocreate'))
pool = parpool(1);
tic;
parfor i = 1:12
   c(i) = max(eig(rand(L)));
end
time(1) = toc;

time =

   10.1577    6.2620    4.4965
```

1,2,4 cores

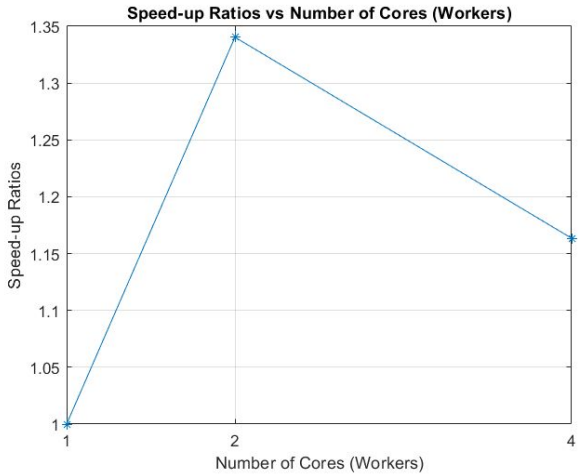


Example 2.2

Low Values (L=45)

```
time =
```

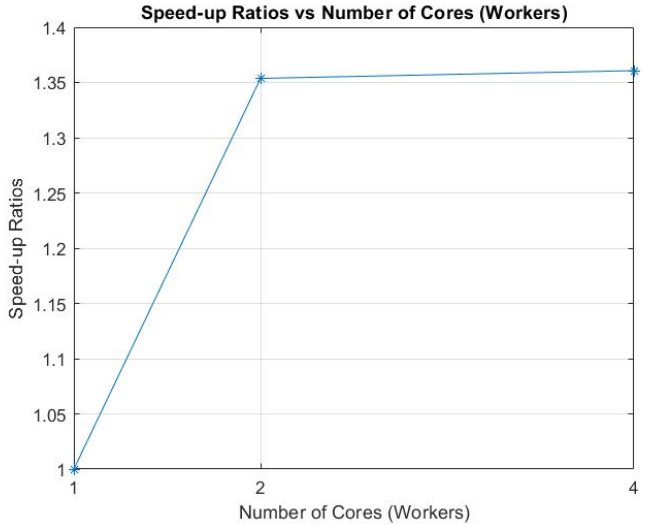
1 core:	2 cores:	4 cores:
0.2635	0.1966	0.2265



High Values (L=4000)

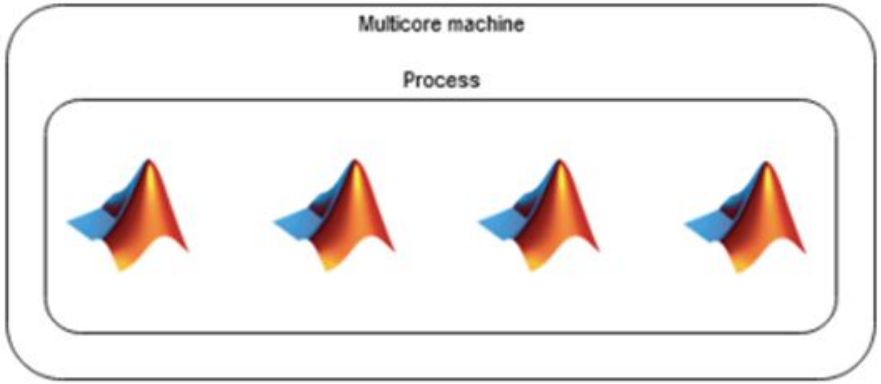
```
time =
```

1 core:	2 cores:	4 cores:
310.6736	229.4980	228.3127



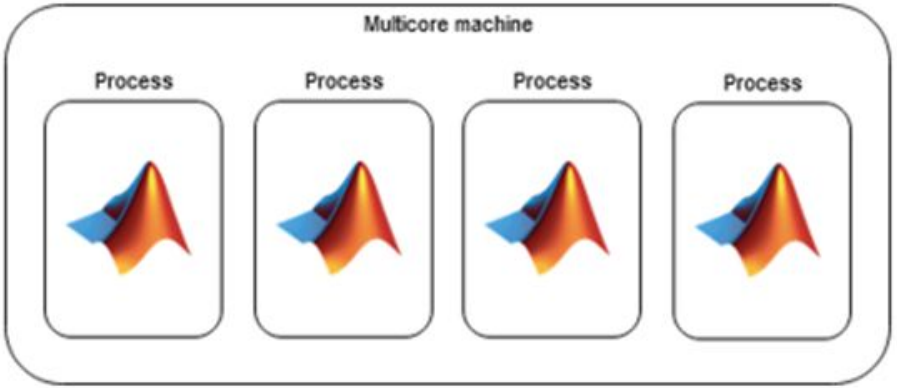
6. Thread-Based and Process-Based Environments

Thread-Based Environments



- **Thread-Based Environments** can share memory, so they can access numeric data without copying → more memory efficient.

Process-Based Environments



- **Process-Based Environments** are more robust in the event of crashes. If a process worker crashes, then the MATLAB client does not crash.
- For large data usage in Process Based Environments, you can use cluster features, such as *batch* function.

https://www.mathworks.com/help/parallel-computing/choose-between-thread-based-and-process-based-environments.html#:~:text=Process%2Dbased%20environments%20have%20the,MATLAB%20client%20does%20not%20crash.

6. Thread-Based and Process-Based Environments

Example 3

```
% Calculation (random numbers generation)
X = rand(3000, 3000);

% Parallel Computing ("Process-based environment on local machine")
delete(gcp('nocreate'))
pool = parpool('local');
% Execution time calcutation
ticBytes(pool);
%Time for trasnfering data (in Process Based Environment)
tProcesses = timeit(@() fetchOutputs(parfeval(@sum,1,X,'all')))
tocBytes(pool)

% Parallel Computing ("Thread-based environment on local machine")
delete(gcp('nocreate'))
pool = parpool('threads');
%
ticBytes(pool);
%Time for trasnfering data (in Thread Based Environment)
tThreads = timeit(@() fetchOutputs(parfeval(@sum,1,X,'all')))
tocBytes(pool)
```

```
tProcesses =

    0.2594
```

	BytesSentToWorkers	BytesReceivedFromWorkers
1	1.08e+09	34830
2	0	0
3	0	0
4	0	0
Total	1.08e+09	34830

```
Parallel pool using the 'local' profile is shutting down.
Starting parallel pool (parpool) ...
Connected to the parallel pool (number of workers: 4).

tThreads =

    0.0156
```

	BytesSentToWorkers	BytesReceivedFromWorkers
Total	0	0

```
Without data transfer, this example is 16.61x faster.
```

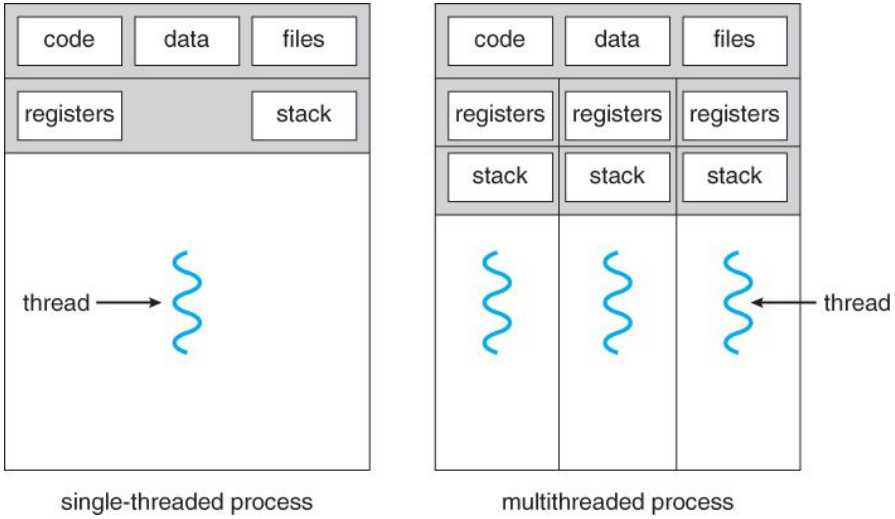
7. Single-Threaded vs Multithreaded

Multithreading Advantages:

- Scalability
- Responsiveness
- Resource sharing
- Economy

MATLAB Limitations:

- Cannot change the thread numbers in MATLAB
- By Default: Multithreaded
- Cannot compared the results of Single-threaded vs Multithreaded



https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/4_Threads.html

8. Summary

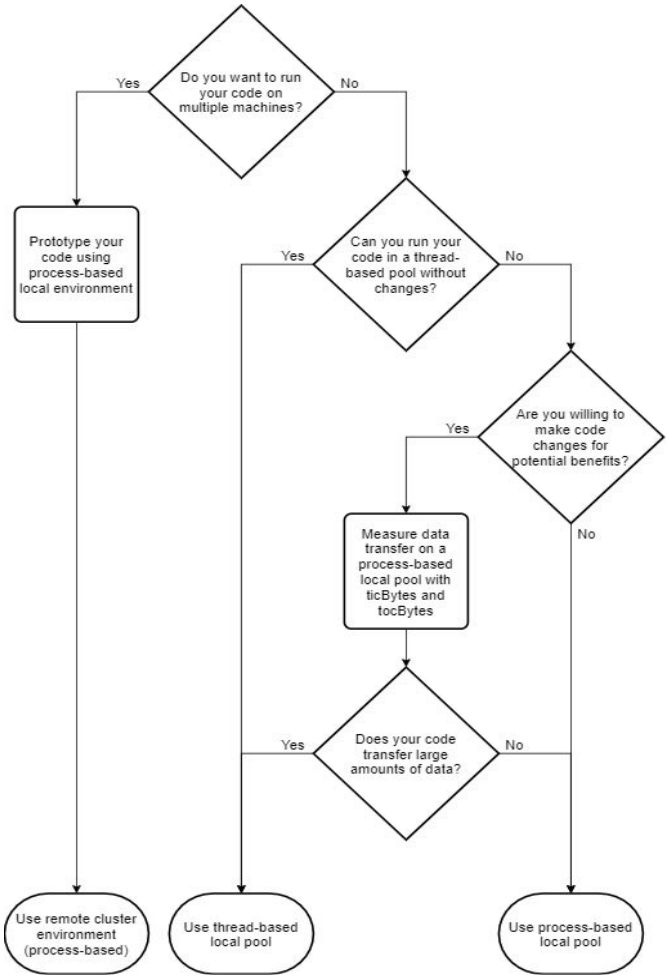
Notes if using MATLAB for Parallelism :

- Parallel Computing: Multicores are performing better than single core, but to an extend.

MATLAB Limitations:

- By Default: Multithreaded, Multicores
- Cannot change the thread numbers in MATLAB
- Cannot compared the results of Single-threaded vs Multithreaded

- **Thread-Based Environments** can share memory, so they can access numeric data without copying → more memory efficient.
- **Process-Based Environments** are more robust in the event of crashes. If a process worker crashes, then the MATLAB client does not crash.
- For large data usage in Process-Based Environments, you can use **cluster** features, such as *batch* function.



https://www.mathworks.com/help/parallel-computing/choose-between-thread-based-and-process-based-environments.html#:~:text=Process%2Dbased%20environments%20have%20the,MATLAB%20client%20does%20not%20crash.

JEMARO

With the support of the Erasmus+ Programme of the European Union

Thank you for listening