Cómo usar Matlab 2024a en el HPC de la Universidad de Cádiz

Introducción

La Universidad de Cádiz dispone actualmente de una licencia campus para usar el software Matlab. En virtud de esta licencia se ha instalado dicho software en el cluster HPC en su versión 2024a con el conjunto completo de herramientas que dicho software suministra.
El servidor de licencias dispensa tanto la licencia *«MATLAB Parallel Server»* como la versión *«MATLAB (Concurrent)»*.

Uso de MATLAB (Concurrent)

Para el uso de Matlab en su formato concurrente hemos de generar un script en el lenguaje propio de Matlab y grabarlo en un fichero con extensión .m.
Dicho script deberá hallarse en una carpeta de nuestra cuenta del cluster junto con todos los ficheros de entrada necesarios para su ejecución.
Las líneas para lanzar este script dentro del script sbatch (trabajo_matlab.sbatch v.g.) son estas:

```
module load Matlab/R2024a
matlab -nojvm -batch MATLAB_command
```

Donde `MATLAB_command` es el nombre de nuestro script .m.
Luego lanzamos el trabajo al modo usual en Slurm:

```
sbatch trabajo_matlab.sbatch
```

Uso de «MATLAB Parallel Server»

Introducción

Matlab en su versión «Parallel Server» se usa para lanzar y gestionar todo el proceso de ejecución de nuestros trabajos desde nuestro propio equipo, mediante comandos de Matlab.
Para usar este método será necesario:

- Configurar nuestra cuenta en el cluster para que cargue el módulo de Matlab automáticamente.
- Tener instalado Matlab R2024a en nuestro equipo.
- Añadir los scripts necesarios para la integración con Slurm.
- Y generar el perfil del cluster en nuestra instalación.

Pasos que detallaremos a continuación.

Configuración de la cuenta en el cluster

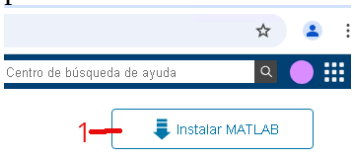
La configuración de nuestra cuenta en el cluster es muy simple: es sólo asegurarse de que las variables necesarias para ejecutar MATLAB están presentes en el momento de iniciar sesión, sin necesidad de hacer nada más.
Para ello basta con añadir la línea que carga el módulo correspondiente a nuestro fichero .bashrc.
Para ello iniciamos sesión ssh en el cluster y ejecutamos las ordenes:

```
~$ echo -e "module load Matlab/R2024a\n" >> ~/.bashrc
~$ tail -2 ~/.bashrc
module load Matlab/R2024a

~$
```

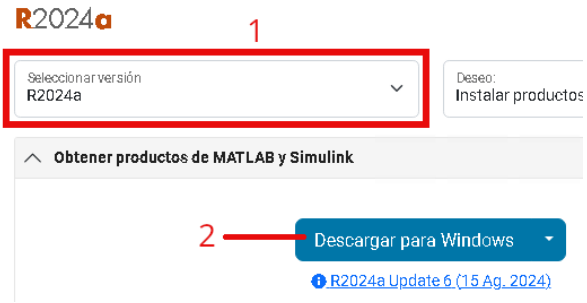
NOTA: La segunda orden es para comprobar que la línea se ha añadido al fichero correctamente.

Instalación de Matlab R2024a

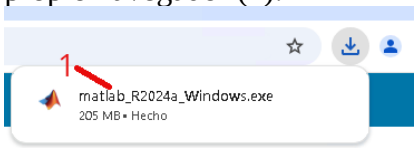
Para instalar Matlab hemos de conectarnos a la URL de Mathworks
https://www.mathworks.com/products/get-matlab.html
Nos pedirá que iniciemos sesión con nuestra cuenta de correo de la UCA. Una vez iniciada sesión, pinchamos en el enlace «`Instalar Matlab`» (1):



En la página que no sale debemos seleccionar la versión R2024a (1) y pinchar en «`Descargar para Windows`» (2):

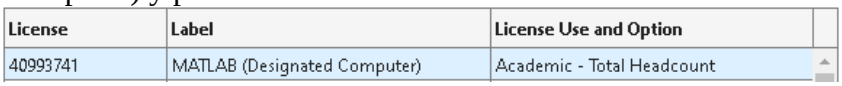


Una vez se complete la descarga debemos lanzar el instalador, cosa que podemos hacer desde el propio navegador (1):



Nos preguntará si permitimos que se realicen cambios en el dispositivo, a lo que respondemos pinchando en «`Sí`». Y nos aparecerá el primer paso de la instalación que consiste en iniciar sesión en nuestra cuenta de Matlab con la dirección de correo de la UCA.
Cuando hayamos iniciado sesión se nos pedirá que aceptemos el acuerdo de licencia, lo que hacemos marcando la opción «`Yes`» (1) y pinchando en «`Next`» (2):

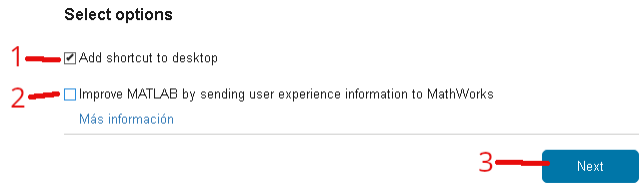
El siguiente paso es escoger el tipo de instalación. Escogemos el «MATLAB (Designated Computer) y pinchamos en «Next»:



License	Label	License Use and Option
40993741	MATLAB (Designated Computer)	Academic - Total Headcount

En la selección de carpeta de destino dejamos el valor por defecto («`C:\Program Files\MATLAB\R2024a`») y pinchamos en «`Next`».
El siguiente paso es seleccionar los componentes que se instalarán. Por defecto están todos marcados. Si queremos ahorrar espacio y sabemos que no vamos a usar algunas Toolboxes podemos desmarcarlas, asegurándonos siempre que la «`Parallel Computing Toolbox`» está marcada. Pinchamos en «`Next`»:

El penúltimo paso es decidir sobre las opciones de instalación. Yo suelo marcar que se me instale un enlace en el escritorio (1) y que no envíe informes a Mathworks (2). Pinchamos en «`Next`» (3):



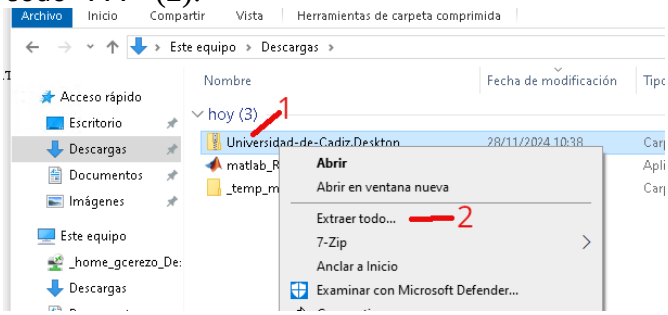
Al pinchar en «`Begin Install`» comenzará la instalación. Cuando termine podemos pinchar en «`Close`».

Añadir los scripts necesarios a nuestro equipo local

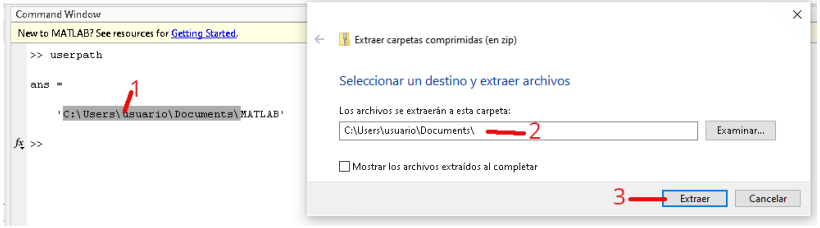
Lo primero es descargarnos los scripts necesarios desde la página web de supercomputación: https://supercomputacion.uca.es/wp-content/uploads/2023/06/Universidad-de-Cadiz.Desktop.zip
Para saber dónde descomprimir este archivo, en nuestro equipo local, iniciamos MATLAB y ejecutamos la orden «`userpath`»:

```
>> userpath

ans =

    'C:\Users\usuario\Documents\MATLAB'
```

Pinchamos con el botón derecho del ratón sobre el archivo descargado (1) y elegimos «`Extraer todo ...`» (2):



Del resultado del comando «`userpath`» copiamos el camino sin el «`MATLAB`» final (1), lo copiamos en el campo para la carpeta de destino (2) y pinchamos en «`Extraer`» (3):

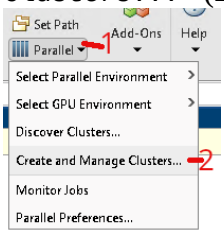


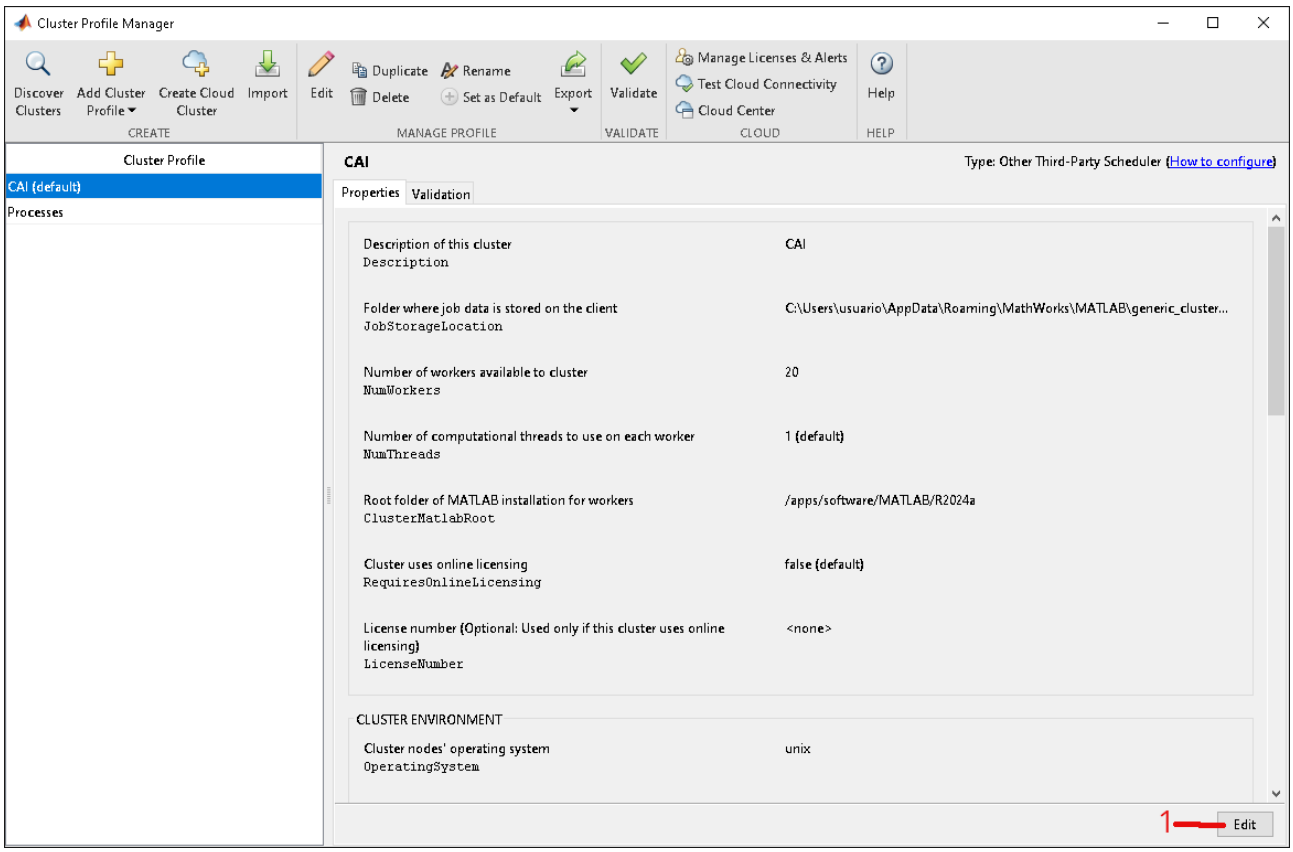
Generar el perfil del cluster

Ahora debemos generar el perfil de nuestro cluster. Para ello ejecutamos la orden *«configCluster»*. Este script está personalizado para nuestra Universidad, por lo que lo único que pedirá será nuestro nombre de usuario en el cluster («`uDNI`»):

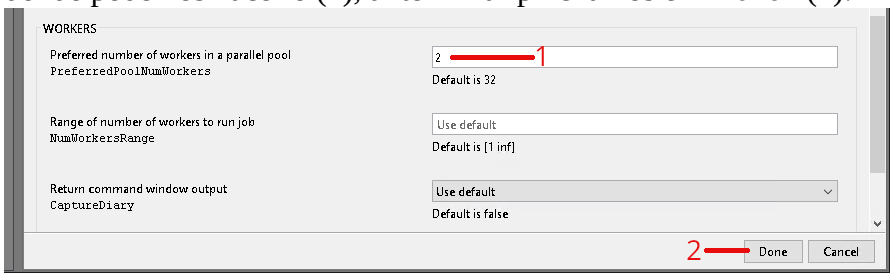
```
>> configCluster
Username on CAI (e.g. uDNI): u********
Complete.  Default cluster profile set to "CAI".
>>
```

Una vez hecho esto, en la lista de perfiles de clusters de supercomputación aparecerá el perfil *«CAI»* como perfil por defecto: pinchamos en «`Parallel (1)` → `Create and Manage Clusters...`» (2) y obtenemos:





Si queremos cambiar algún valor podemos pinchar en «`Edit`» y la pantalla cambia a un formulario donde podemos hacerlo (1), al terminar pinchamos en «`Done`» (2):



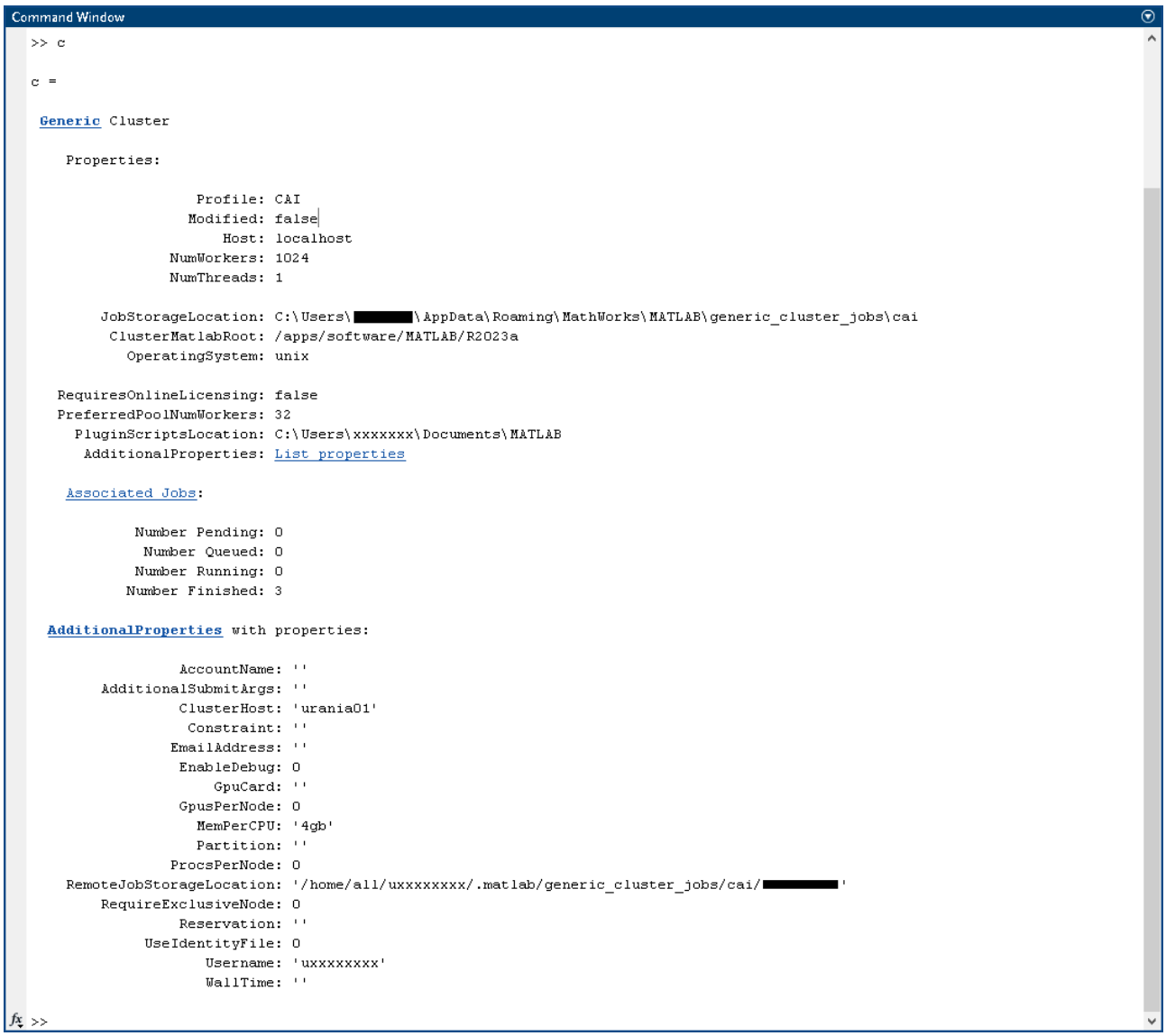
Al final de este documento se añade, como apéndice un listado completo de todos los valores del perfil del cluster. Puede compararlo con el suyo y alterar algún valor en función de sus necesidades.

Configuración de los trabajos

Antes de enviar los trabajos podemos especificar varios parámetros para pasárselos a nuestros trabajos como la cola, el email para avisarnos, la duración esperada, etc.
Lo primero es obtener un manejador para el cluster:

```
c=parcluster('CAI');
```

Aquí si ejecutamos «c» y pinchamos en «list properties» veremos todo lo que podemos configurar para nuestros trabajos:



Para definir, por ejemplo, una duración de 3 días y 6 horas haríamos:

```
>> c.AdditionalProperties.WallTime = '3-06:00';
```

Para especificar que vamos a usar dos gpus debemos hacer:

```
>> c.AdditionalProperties.GpusPerNode = 2;
>> c.AdditionalProperties.Partition = 'gpu';
```

La propiedad «`AdditionalSubmitArgs`» sirve para añadir cualquier opción no especificada aquí.

Si queremos que estas modificaciones estén disponibles entre diferentes sesiones MATLAB, nuestros valores por defecto, debemos salvar las modificaciones con:

```
>> c.saveProfile;
```

Gestión de trabajos

Lanzar un trabajo

Para lanzar un trabajo usamos el método «`batch`» de parcluster, que devuelve un objeto del tipo trabajo que debemos guardar para poder acceder a los resultados:

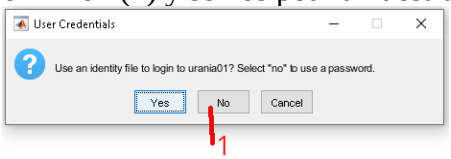
```
>> job = c.batch(función,N,X1,X2,...);
```

(**NOTAS**:

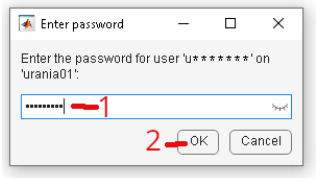
- El script «función» es copiado al nodo que ejecutará el trabajo. No hay que incluir la extensión «.m».
- N es el número de argumentos devueltos por el script.
- X_1, X_2, … son los argumentos de entrada al script.

)

La primera vez que lancemos un trabajo nos pedirá si queremos usar un «Identity File». Pinchamos en «No» (1) y se nos pedirá nuestra clave en el cluster:



La introducimos (1) y pinchamos en «Ok» (2):



A partir de ahora, siempre que usemos este perfil, se nos pedirá esta clave (una vez por sesión).
Con esta acción el trabajo se está ejecutando en el cluster y podemos cerrar nuestro equipo.
Para reanudar el trabajo debemos volver a obtener un manejador para el cluster y solicitar la lista de trabajos en ejecución:

```
>> c = parcluster;
>> jobs = c.Jobs;
```

Si tenemos varios trabajos en ejecución y queremos comprobar el número dos podemos hacer:

```
>> job = c.Jobs(2);
>> job.State
```

Que nos dirá si está en ejecución o ha terminado («`finished`»). Para obtener los resultados, si pasamos una función («`@pwd`» por ejemplo) usamos «`fetchOutputs`» y si pasamos un script usamos «`load`».

```
>> job.fetchOutputs{:}
>> load(job,'x');
```

Si hubiese problemas, eliminamos el trabajo para liberar los recursos:

```
>> job.delete;
```

Lanzar un trabajo paralelo

Vamos a suponer este script:

```
function [t, A] = parallel_example(iter)

if nargin==0
    iter = 8;
end

disp('Start sim')

t0 = tic;
parfor idx = 1:iter
    A(idx) = idx;
    pause(2)
    idx
end
t = toc(t0);

disp('Sim completed')

save RESULTS A

end
```

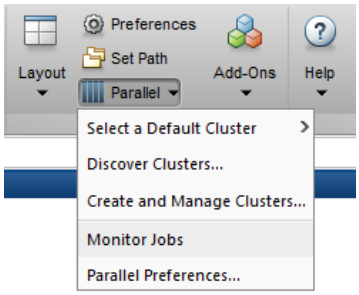
Esta vez, al usar batch especificamos un «`pool`» (en este caso será de cuatro nodos):

```
>> job = c.batch(@parallel_example, 1, {16}, 'Pool',4, ...);
```

Por lo demás el trabajo es igual al anterior.

Monitorizar los trabajos

Alternativamente a la expresión «`job.State`» podemos usar la interfaz gráfica para ver el estado de nuestros trabajos:



Ahí podemos ver su estado e incluso eliminarlos pinchando con el botón derecho del ratón sobre el trabajo y seleccionando «`Delete`».

Depuración de errores

Para ver si han ocurrido errores podemos obtener lo que hubiera sido la salida por pantalla con el comando:

```
>> diary(job)
```

Si queremos ver la salida de depuración usamos la expresión:

```
>> c.getDebugLog(job)
```

Por último, puede ser interesante saber el ID del trabajo en el cluster. Se obtiene con la expresión:

```
>> schedID(job)
```

Una vez obtenido ese número se pueden usar los comandos apropiados en el cluster para obtener información.

Más información

- Parallel Computing Coding Examples
- Parallel Computing Documentation
- Parallel Computing Overview
- Parallel Computing Tutorials
- Parallel Computing Videos
- Parallel Computing Webinars

Ejemplo de ejecución de un trabajo paralelo

```
>> c=parcluster('CAI');
>> j=batch(c,@version,1,{}, 'pool', 3, 'CurrentFolder', '.');

additionalSubmitArgs =

    '--ntasks=4 --cpus-per-task=1 --ntasks-per-core=1 --mem-per-cpu=4gb'

>> j.State

ans =

    'running'

>> j.State

ans =

    'finished'

>> j.fetchOutputs{:}

ans =

    '24.1.0.2537033 (R2024a)'

>>
```

Mientras «`j.State`» devuelva `'running'` podemos ver el trabajo con los comandos de Slurm:

```
11:04:54 u********@urania02.uca.es:~$ squeue
   JOBID PARTITION     NAME         USER ST       TIME  NODES NODELIST(REASON)
    4904    normal     Job2    u********  R       0:14      1 urc11
```

Apéndice: valores del perfil

NOTA: cliente designa al equipo desde el que el usuario lanza los trabajos. En este escrito se supone que es un equipo Windows y los PATHS aparecen en el formato típico de ese S.O.
https://supercomputacion.uca.es/wp-content/uploads/2023/06/Universidad-de-Cadiz.Desktop.zip

	Description of this cluster (Description)	CAI
	Folder where job data is stored on the client (JobStoreLocation)	C:\Users\USUARIO[i]\AppData\Roaming\MathWorks\MATLAB\generic_cluster_jobs\cai
	Number of workers availabel to cluster (NumWorkers)	1024[ii]
	Number of computational threads to use on each worker (NumThreads)	1(default)[iii]
	Root folder of MATLAB installation for workers (ClusterMatlabRoot)	/apps/software/MATLAB/R2024a
	Cluster uses online licensing (RequiresOnlineLicensing)	false(default)
	License number (Optional: Used only if this cluster uses online licensing) (LicenseNumber)	<none>
CLUSTER ENVIRONMENT		
	Cluster nodes' operating system (OperatingSystem)	unix
	Job storage location is accesible from client and cluster nodes	false

	(HasSharedFilesystem)		
SCHEDULER PLUGIN			
	Folder containing scheduler plugin scripts (PluginScriptsLocation)	C:\Users\USUARIO\Documents\MATLAB\IntegrationScripts\cai[iv]	
	Additional properties for plugin scripts (AdditionalProperties)		
		AccountName	
		AdditinalSubmitArgs	
		ClusterHost	urania01
		Constraint	
		EmailAddress	usuario@uca.es[v]
		EnableDebug	false
		GPUsPerNode	0[vi]
		MemPerCPU	4gb
		Partition	normal[vii]
		ProcsPerNode	0
		RemoteJobStorageLocation	/home/all/USUARIO[viii]/.matlab/generic_cluster_jobs/cai/MAQUINALOCAL
		RequireExclusiveNode	false
		Reservation	XXX[ix]
		UseIdentityFile	
		Username	uDNI[x]
		WallTime	
FILES AND FOLDERS			
	Automatically send code files to cluster. Data files or folders must be listed in the AttachedFiles property (AutoAttacheFiles)	true(default)	
	Manually specify files and folders to copy from client to cluster nodes (One entry per line) (AttachedFiles)	<none>	
	Manually specify folders to add to the workers' search path (One entry per line) (AdditionalPaths)	<none>	
WORKERS			
	Preferred number of workers in a parallel pool (PreferredPoolWorkers)	32	
	Range of number of workers to run job (NumWorkersRange)	[1 inf](default)	
	Return command window output (CaptureDiary)	false(default)	
	Manually specify environment variables to copy from client to workers (One entry per line) (EnvironmentVariables)	<none>	

i USUARIO debe sustituirse por el nombre del usuario en el equipo cliente.
ii Se corresponde con el número de licencias concurrentes para este cluster. Sería el número máximo de procesos que se pueden ejecutar a la vez en el cluster entre todos los usuarios .
iii Número de tareas que se pueden ejecutar a la vez por cada unidad de cálculo (que equivale a CPU en la nomenclatura de SLURM y se corresponde con los CORES físicos). Puesto que en el cluster no se permite el multithreading, debe valer 1.
iv Es la carpeta donde hemos descomprimido el zip con los scripts Matlab que nos descargamos.
v Dirección de correo en la que Slurm notificará el inicio/fin del trabajo.
vi 0 para las colas normal y fat; 0, 1 o 2 para la cola gpu.
vii O fat, o gpu.
viii Sustituir USUARIO por el nombre del usuario en el cluster
ix El nombre de la reserva creada en Slurm si procede. Normalmente estará en blanco.
x Cambiar por el nombre del usuario en el cluster.