https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 165 - 184

Manejo distribuido de matrices a gran escala en un equipo HPC

Large-scale distributed array management on an HPC system

Información del reporte:

Para citar este reporte técnico:

Durán Chavesti, A. (2026). Manejo distribuido de matrices a gran escala en un equipo HPC. *Cuadernos Técnicos Universitarios de la DGTIC, 4* (2), páginas (165 - 184). https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150

Adrián Durán Chavesti
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
adrian.chavesti@iimas.unam.mx
ORCID: 0000-0002-0542-1627

Resumen

La necesidad de computación de alto rendimiento se refiere a la práctica de utilizar recursos de cómputo más potentes que una computadora personal o servidor típico para abordar problemas complejos que requieren grandes cantidades de procesamiento. Una gran parte de las operaciones a resolver son matrices a gran escala. Por esta razón, se implementó un ambiente de clúster de servidores con la herramienta "Servidor Paralelo de MATLAB", que permitió mostrar cómo es que se pueden manejar matrices de gran tamaño de forma paralela y distribuida haciendo uso de un equipo de cómputo de alto rendimiento y el lenguaje de alto nivel MATLAB.

Se logró manejar y distribuir un par de matrices de 49,000 x 49,000 elementos. En contraste, una computadora personal típica con 16 GB de memoria sólo puede manejar un par de matrices de 23,000 x 23,000 elementos.

Este trabajo permitió verificar la facilidad y flexibilidad con la cual se puede escalar un problema usando servidores conectados en red con la herramienta "Servidor Paralelo de MATLAB". Esto se logra usando el mismo algoritmo y añadiendo más servidores a la configuración del clúster. En esta implementación, se usó un sistema de tres nodos conectados en una red local, por lo que, si quiere reproducir, se recomienda, antes de empezar la instalación, tener a la mano la licencia para poder distribuir MATLAB.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 166 - 184

El nivel de escalado del clúster va a estar relacionado directamente con el número de servidores que se tengan disponibles. En este caso, han sido beneficiados tres usuarios de la línea de investigación en cómputo de alto rendimiento. El sistema operativo usado fue Linux Ubuntu Server 24.04 LTS y el *software* MATLAB R2024a.

Palabras clave: Grupo de servidores, cómputo científico, MATLAB distribuido, escalado de matrices, supercómputo.

Abstract

The need for high-performance computing refers to the practice of using computing resources more powerful than a typical personal computer or server to address complex problems that require large amounts of processing power. A large part of the operations to be solved involve large-scale matrices; therefore, a server cluster environment was implemented with the "MATLAB Parallel Server" tool. This allowed them to demonstrate how large matrices can be handled in a parallel and distributed manner using high-performance computing equipment and the high-level language MATLAB.

It was possible to handle and distribute a pair of 49,000 x 49,000 matrices, whereas a typical personal computer with 16 GB of memory can only handle a pair of 23,000 x 23,000 matrices.

This work demonstrated the ease and flexibility with which a problem can be scaled using networked servers with the MATLAB Parallel Server tool, employing the same algorithm simply by adding more servers to the cluster configuration.

This implementation used a three-node system connected to a local network, so if one wishes to replicate it, it is recommended to have your MATLAB license on hand before starting the installation.

The cluster's scalability will be directly related to the number of available servers. In this case, three users from the high-performance computing research group benefited from the system. The operating system used was Ubuntu Server 24.04 LTS (Linux), and the software used was MATLAB R2024a.

Keywords: *Server cluster, scientific computing, distributed MATLAB, matrix scaling, supercomputing.*

1. INTRODUCCIÓN

La computación distribuida es una disciplina consolidada en la informática y la ingeniería. Ha evolucionado durante los últimos 40 años hasta convertirse en una de las metodologías más importantes para la implementación de los servicios de procesamiento de datos necesarios para prácticamente todas las actividades de la sociedad (Delfino, 2020).

Ingenieros y científicos utilizan MATLAB para analizar y diseñar sistemas y productos. El lenguaje de MATLAB, basado en matrices, es la forma más natural para expresar las matemáticas computacionales (MathWorks, s.f.).

El cálculo matricial es fundamental en numerosas aplicaciones analíticas con uso intensivo de datos, como es el caso de la minería de datos en redes sociales, los sistemas de recomendación, el procesamiento del lenguaje natural, el análisis numérico a gran escala y la física computacional. Por lo anterior, se considera

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 167 - 184

una parte importante de la construcción de plataformas computacionales para tales problemas. En el ámbito del *big data*, se ha creado una creciente demanda de implementaciones escalables de cálculo matricial en conjuntos de datos masivos.

La multiplicación de matrices es una operación dominante en muchas aplicaciones analíticas de *big data,* pero consume mucho tiempo. Por lo tanto, la optimización del rendimiento es un problema de investigación importante y fundamental (Gu *et al*., 2017). Asimismo, HouZhen *et al*. (2020) mencionan que las operaciones de matrices a gran escala son fundamentales y, tomando en cuenta a Lim *et al.* (2018), se deben optimizar los algoritmos clásicos para aprovechar todo el potencial de las nuevas características del *hardware*.

Se han propuesto técnicas de gestión automática del paralelismo para minimizar la sobrecarga relacionada con la creación y sincronización de hilos, manteniendo así altos niveles de rendimiento en cargas de trabajo distribuidas (Westrick *et al*., 2024). Es por ello que el *Toolbox Parallel Server* es una opción que se considera como gestor automático de distribución de trabajos. Asimismo, se pueden utilizar arreglos distribuidos en *Parallel Computing Toolbox* para ejecutar aplicaciones de *big data* utilizando la memoria combinada del clúster (MathWorks, s. f.).

Puede usarse el *software* Octave como una alternativa libre al *software* comercial MATLAB, éste último cuenta con una interfaz más pulida, amplias herramientas especializadas y soporte comercial, lo que lo hace excelente para resolver necesidades avanzadas de la industria donde su rendimiento optimizado y los kits de herramientas integrales (como procesamiento de señales y sistemas de control) son cruciales. Otra razón por la que se está usando MATLAB en este trabajo es el aprovechamiento de las licencias que ofrece la UNAM, evitando así la necesidad de uso de *software* libre.

El objetivo es instalar y configurar un sistema paralelo y distribuido que ayude a investigadores, académicos y tesistas a resolver matrices de grandes proporciones. Para ello, se usa el lenguaje de programación MATLAB y su herramienta MATLAB *Parallel Server,* con la cual pueden escalar a un entorno de decenas de servidores.

2. DESARROLLO TÉCNICO

Paralelizar un código, aún en nuestros días, puede llegar a ser una tarea complicada y tardada. Para ello, existen herramientas altamente especializadas que permiten hacer menos difícil la vida del programador, tal como MATLAB. En este lenguaje, se realizaron pruebas de multiplicación de matrices cuadradas comúnmente encontradas en problemas de ingeniería, como son las de tipo diagonal y triangular. Para ubicar temporalmente este trabajo, las pruebas se realizaron en agosto de 2025. MATLAB nos proporciona dos herramientas muy poderosas para escalar un código, éstas son:

Parallel Computing Toolbox. Permite resolver problemas de computación y datos intensivos utilizando procesadores multinúcleo, GPU y clústers de computadoras. La herramienta ofrece construcciones de alto nivel (bucles *for* paralelos, tipos de matrices especiales y algoritmos numéricos paralelizados) y también tiene funciones habilitadas para procesamiento en paralelo (MathWorks, s.f.).

MATLAB Parallel Server. Permite escalar programas MATLAB y simulaciones Simulink a clústers y nubes. Puede crear prototipos en el escritorio y ejecutarlos en clústers y nubes sin necesidad de recodificar.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 168 - 184

MATLAB *Parallel Server* admite trabajos por lotes, cálculos paralelos interactivos y cálculos distribuidos con matrices grandes (MathWorks, s.f.).

2.1 METODOLOGÍA

A partir de las herramientas antes mencionadas, se procedió a instalar, configurar y poner en marcha un ambiente paralelo y distribuido de MATLAB. Además, se realizaron pruebas en las que se obtuvieron gráficas de *speedup* y eficiencia al escalar el código base. Cabe señalar que este ambiente no sólo se limita al manejo de matrices, sino que puede usarse para escalar simulaciones con la herramienta Simulink y solucionar problemas de *big data*.

2.2 INSTALACIÓN

La PC de comparación es una Dell XPS con las siguientes características:

Procesadores: 1 chip Intel Core i7-4770 3.4 GHz con 4 núcleos.

Memoria RAM: 16 GB.

Disco: 1 TB.

Tarjeta de red: 1 GbE.

Los servidores que se usaron son de la marca Dell PowerEdge R720 con las siguientes características:

Procesadores: 2 chips Intel Xeon E5-2640 2.5 GHz con 6 núcleos cada uno.

Memoria RAM: 64 GB.

Disco: 2 TB.

Tarjeta de red: 1 GbE.

Se requiere un tipo de licencia especial para instalar MATLAB *Parallel Server*; si el usuario no tiene esta licencia, tiene que ponerse en contacto con la Secretaría Técnica de su entidad o dependencia de adscripción.

Enseguida, se describen los pasos iniciales para la instalación:

- Crear un directorio para albergar el *software*.
- Montar la imagen de MATLAB en el sistema de archivos de Linux.
- En el directorio donde se montó el paquete, ejecutar el comando "*install*".

En el Anexo A, se describen con mayor detalle los pasos siguientes de la instalación.

2.3 CONFIGURACIÓN

Un *job* es una operación extensa y se divide en segmentos llamados tareas. La sesión de MATLAB donde se definen el *job* y sus tareas se denomina sesión de cliente. El cliente utiliza el *software Parallel Computing Toolbox* para definir trabajos y tareas. MATLAB *Parallel Server* ejecuta el trabajo evaluando cada una de las tareas y devolviendo el resultado a su sesión de cliente.

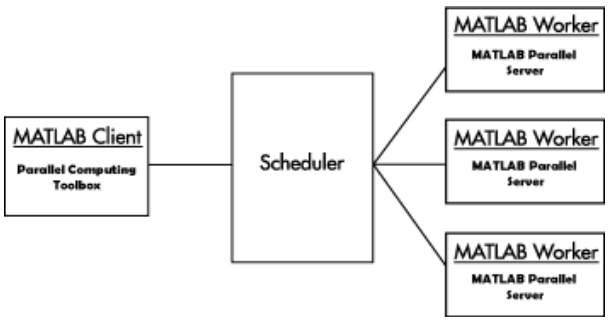
https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 169 - 184

Parallel Computing Toolbox permite ejecutar un clúster de trabajadores de MATLAB en la máquina local. MATLAB *Parallel Server* permite ejecutar tantos trabajadores de MATLAB en un clúster remoto de computadoras como lo permita la licencia. El Planificador de Trabajos de MATLAB es la parte del *software* del servidor que coordina la ejecución de trabajos y la evaluación de las tareas. Distribuye las tareas para su evaluación en las sesiones individuales de MATLAB del servidor, denominadas trabajadores.

En la Figura 1, se observa la configuración de MATLAB en paralelo. En este caso, se implementa en una PC y un servidor con procesadores multinúcleo.

Figura 1

Configuración básica de cómputo paralelo

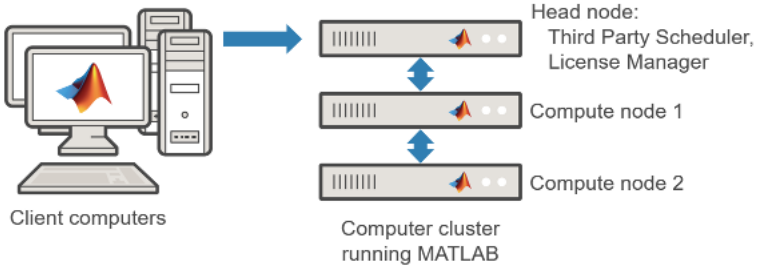


Nota. Recuperado de MathWorks. (s. f.).

En la Figura 2, se observa la configuración de MATLAB distribuido. En este caso, se implementa en varios servidores con procesadores multinúcleo a través de una red de comunicaciones.

Figura 2

Configuración básica de cómputo distribuido



Nota. Recuperado de MathWorks. (s. f.).

Para el acceso y configuración a la herramienta MATLAB *Parallel Server,* ejecutar "*admincenter*":

$> Directorio_Instalacion_MATLAB/toolbox/parallel/bin/admincenter

Los pasos de configuración detallados se muestran en el Anexo B.

Una vez configurada la herramienta, se procede a crear y probar un *script,* como se muestra a continuación.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 170 - 184

2.4 PRUEBA DE FUNCIONAMIENTO

Una vez que se ha configurado el clúster, se hace una prueba de funcionamiento. Para ello, se realizó una serie de operaciones usando números aleatorios que se muestra en la Figura 3.

Figura 3

Código usado para la prueba de funcionamiento

```
%% Prueba de funcionamiento de MATLAB parallel server

% Declaracion de variables.
n = 400;
A = 500;
a = zeros(1,n);

% Inicia el temporizador
tic

% "parfor" permite paralelizar un ciclo "for"
% Las operaciones que se encuentran dentro del ciclo "for" realizan lo
% siguiente en paralelo:
% Obtiene el maximo de los valores absolutos de un vector columna
% que contiene los eigen valores de la matriz cuadrada A de numeros
% aleatorios.
parfor i = 1:n
    a(i) = max(abs(eig(rand(A))));
end

% Termina el temporizador
toc

```

En la Figura 4, en la columna "*Status*", se puede observar que los trabajadores de MATLAB se encuentran ocupados (*busy*), con ello se asegura el buen funcionamiento de nuestro sistema distribuido.

Figura 4

Trabajadores distribuidos ocupados realizando operaciones

Workers

Start...
Stop...
Resume

Worker				MJS		
Name	Hostname	Status	Up Since	Connection	Name	Hostname
n49_worker04	n49	busy	2025-06-16 14:...	connected	maestro	n49
n49_worker03	n49	busy	2025-06-16 14:...	connected	maestro	n49
n49_worker06	n49	busy	2025-06-16 14:...	connected	maestro	n49
n49_worker05	n49	busy	2025-06-16 14:...	connected	maestro	n49
n52_worker03	n52	busy	2025-06-16 14:...	connected	maestro	n49
n52_worker04	n52	busy	2025-06-16 14:...	connected	maestro	n49
n52_worker05	n52	busy	2025-06-16 14:...	connected	maestro	n49
n53_worker02	n53	busy	2025-06-16 13:...	connected	maestro	n49
n53_worker01	n53	busy	2025-06-16 12:...	connected	maestro	n49
n53_worker04	n53	busy	2025-06-16 14:...	connected	maestro	n49
n53_worker03	n53	busy	2025-06-16 14:...	connected	maestro	n49
n53_worker05	n53	busy	2025-06-16 14:...	connected	maestro	n49
n52_worker01	n52	busy	2025-06-11 17:...	connected	maestro	n49
n52_worker02	n52	busy	2025-06-16 14:...	connected	maestro	n49
n49_worker02	n49	busy	2025-06-16 14:...	connected	maestro	n49

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 171 - 184

La descripción de las demás columnas de la Figura 4 se muestran a continuación:

Worker. Son los trabajadores o nodos de cálculo con sus respectivas características.

Hostname. Es el nombre de cada nodo.

Up Since. Muestra la hora en la cual el nodo está disponible dentro del clúster.

MJS (MATLAB Job Scheduler). Es el despachador de trabajos de MATLAB *Parallel Server.*

Connection. Muestra si el trabajador o *worker* está disponible y puede ser usado.

Name. Es el nombre del clúster creado con MATLAB.

2.5 MANEJO DE MATRICES

En este punto, se realizó una multiplicación de matrices diagonales y triangulares de 23.000 x 23,000 en una PC con 16 GB de memoria RAM. Antes de la multiplicación, se realizaron transformaciones previas de las matrices (transpuesta, inversa y elevadas al cuadrado), esto con el objetivo de mostrar el beneficio de usar varios servidores para resolver problemas complejos. Posteriormente, se escaló el problema para resolver un sistema de 49,000 x 49,000; esta magnitud de elementos ya no es posible manejarla en la PC, por lo que se seleccionaron todos los núcleos disponibles en tres servidores.

El operador de multiplicación de matrices '*' invoca la biblioteca BLAS (*Basic Linear Algebra Subprograms*), optimizada para realizar la multiplicación. Esta biblioteca es multihilo, lo que es una gran ventaja al usar todos los núcleos de un equipo. En la Figura 5, se muestra el código que se ejecutó en una PC (N = 23,000) y en un servidor (N = 49,000), donde "N" es cada una de las dimensiones de las matrices.

Figura 5

Multiplicación de matrices en paralelo sólo para un equipo de cómputo

```
%% Multiplicacion local de matrices de N x N.

% Elementos por fila y columna.
N = 49000;

% Total de elementos de las 3 matrices.
elementos = 3*N*N;

% Creacion de matrices diagonales.
A = diag(rand(1,N));
B = diag(rand(1,N));

%{
%Creacion de matrices triangulares.
A = triu(rand(N));
B = triu(rand(N));
%}

% Multiplicación de matrices con transformaciones previas:
% Transpuesta.
% Inversa.
% Al cuadrado.
% Multiplicación.
tic
C = (inv(A').^2)*(inv(B').^2);
toc

% Imprime las variables y sus detalles.
whos
```

En la Figura 6, se muestra el código que se ejecutó en un clúster de tres servidores. Con ello, el código se pudo escalar y ejecutar de forma distribuida usando la función "*distributed*".

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 172 - 184

Figura 6

Operación de matrices distribuidas en tres servidores

```
%% Arreglos distribuidos

% Representan aquellos arreglos que se dividen entre los trabajadores
% (workers) en un grupo paralelo.
% Una matriz distribuida se parece a una matriz MATLAB
% normal en la forma en que indexa y manipula sus elementos, pero ninguno
% de sus elementos existe en el cliente.

% Dimension de filas y columnas.
N = 49000;

% Total de elementos de las 3 matrices.
elementos = 3*N*N;

% Crea dos matrices de N x N.
% Los elementos de la matriz se distribuyen en
% entre los workers, pertenecientes al cluster creado.
disp("Distribución ...")

% Creacion de matrices diagonales.
A1 = distributed(diag(rand(1,N)));
B1 = distributed(diag(rand(1,N)));

%{
%Creacion de matrices triangulares.
A1 = distributed(triu(rand(N)));
B1 = distributed(triu(rand(N)));
%}

% El calculo se ejecuta sobre los workers.
% La matriz resultado tambien es distribuida.
% Multiplicación de matrices con transformaciones previas:
% Transpuesta.
% Inversa.
% Al cuadrado.
% Multiplicación.
disp("Operaciones ...")
tic
C1 = (inv(A1').^2)*(inv(B1').^2);
toc

whos    % Imprime las variables y sus detalles
```

Se obtuvo la aceleración (*speedup*) y la eficiencia tanto para la prueba con la PC como para la prueba con el servidor. Estas gráficas son esenciales para verificar que un código sea escalable (Foster, 1995).

En la Figura 6, se muestra la creación de A1 y B1 (líneas 21 y 22, 26 y 27) como arreglos distribuidos, los cuales se han dividido entre los *workers*. Esto se muestra en la Figura 7. Las matrices se dividen por columnas y están listas para realizar la multiplicación.

Figura 7

Ejemplo de una matriz dividida entre el número de workers del clúster MATLAB

Fila	W1	W2	W3	…	W36
1	1, 2, 3, …, C	C+1, …, 2C	2C+1, …, 3C	…	35C+1, …, 36C
2	1, 2, 3, …, C	C+1, …, 2C	2C+1, …, 3C	…	35C+1, …, 36C
3	1, 2, 3, …, C	C+1, …, 2C	2C+1, …, 3C	…	35C+1, …, 36C
4	1, 2, 3, …, C	C+1, …, 2C	2C+1, …, 3C	…	35C+1, …, 36C
5	1, 2, 3, …, C	C+1, …, 2C	2C+1, …, 3C	…	35C+1, …, 36C
⋮	⋮	⋮	⋮	⋮	⋮
N	1, 2, 3, …, C	C+1, …, 2C	2C+1, …, 3C	…	35C+1, …, 36C

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 173 - 184

Donde:

C = N/nw

C = Número columnas que le corresponden a cada *worker*.

N: Número de filas o columnas de las matrices.

nw: Número de *workers*.

3. RESULTADOS

La primera prueba se realizó en una PC con 16 GB de memoria RAM, por lo que la dimensión máxima de las matrices fue de 23,000 x 23,000. En la Tabla 1, se observa el número de elementos de las matrices, la memoria usada y el tiempo de ejecución.

Tabla 1

Datos que se obtuvieron al ejecutar una multiplicación de matrices en paralelo en una PC

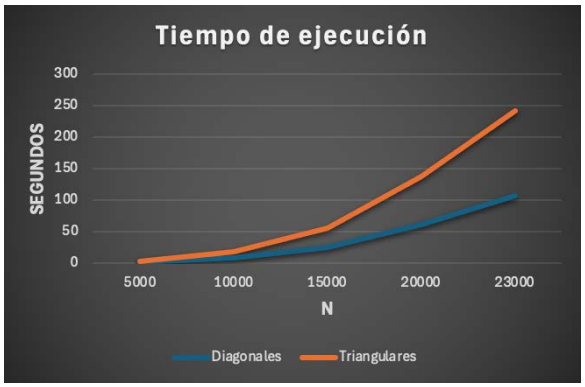
Tamaño de las matrices	Elementos totales de las matrices (x 10^9)	Memoria usada por las matrices (GB)	Memoria total usada por el sistema (GB)	Tiempo de ejecución (s). Matrices diagonales	Tiempo de ejecución (s). Matrices triangulares
5,000 x 5,000	0.075	0.60	2.90	1.1	2.5
10,000 x 10,000	0.300	2.40	4.70	8.0	18.1
15,000 x 15,000	0.675	5.40	7.80	24.9	56.4
20,000 x 20,000	1.200	9.60	12.10	60.9	138.1
23,000 x 23,000	1.587	12.7	15.20	106.9	242.4

En la Figura 8, se observa el tiempo de ejecución de una multiplicación de matrices en una PC.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 174 - 184

Figura 8

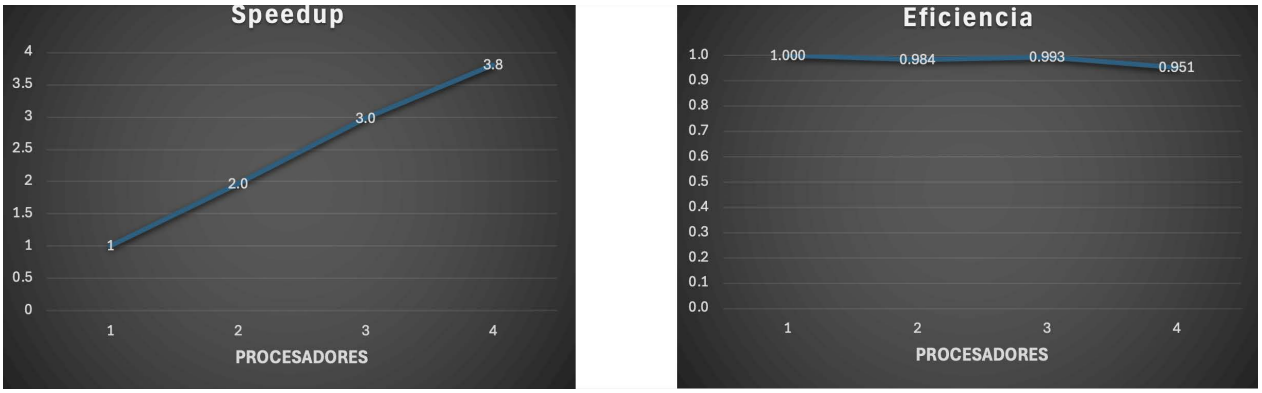
Tiempo de ejecución de multiplicación de matrices en una PC



En la Figura 9, se observa el *speedup* y eficiencia del procesamiento en paralelo usando de uno a cuatro núcleos del procesador Intel Core i7-4770 de la PC.

Figura 9

Speedup y eficiencia en la PC



La segunda prueba se realizó en un solo servidor, la dimensión máxima de las matrices fue de 49,000 x 49,000, el número de elementos pasó de 1.587 x 10^9 a 7.2 x 10^9, más de cuatro veces que en una PC. En la Tabla 2, se observa el número de elementos de las matrices, la memoria usada y el tiempo de ejecución.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 175 - 184

Tabla 2

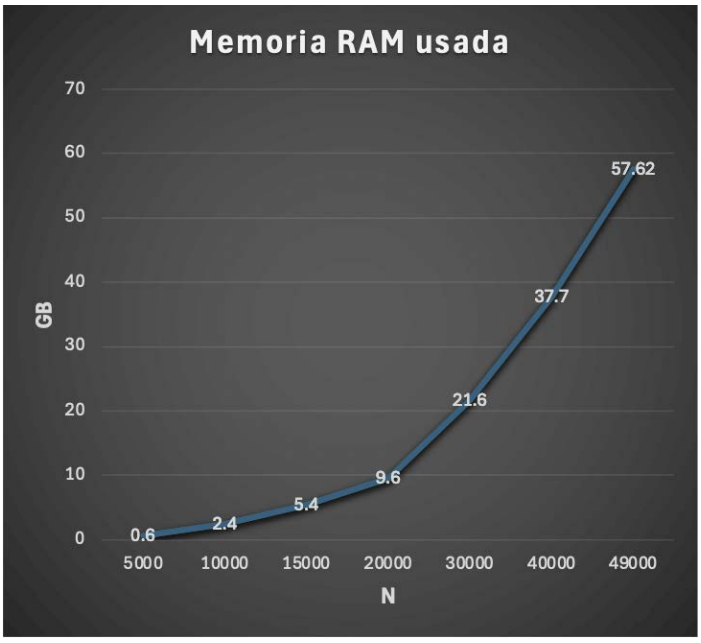
Datos que se obtuvieron al realizar operaciones con matrices en un solo servidor

Tamaño de las matrices	Elementos totales de las matrices (x 10^9)	Memoria usada por las matrices (GB)	Memoria total usada por el sistema (GB)	Tiempo de ejecución (s). Matrices diagonales	Tiempo de ejecución (s). Matrices triangulares
5,000 x 5,000	0.075	0.60	4.24	0.6	1.3
10,000 x 10,000	0.300	2.40	5.96	4.5	10.1
15,000 x 15,000	0.675	5.40	8.80	14.7	33.2
20,000 x 20,000	1.200	9.60	12.70	34.2	77.6
30,000 x 30,000	2.700	21.60	24.10	116.4	264.0
40,000 x 40,000	4.800	37.70	40.10	341.4	774.2
49,000 x 49,000	7.200	57.62	60.23	977.3	2215.9

Como se observa en la Figura 10, la memoria usada se incrementa exponencialmente con la dimensión de las matrices. En un equipo de cómputo personal ordinario (PC), sólo es posible multiplicar matrices de hasta 23,000 x 23,000; sin embargo, se observa en la Figura 11 cómo es posible realizar un procesamiento de matrices de mayor dimensión. En la Figura 12, se puede observar la comparativa entre los tiempos de ejecución entre la PC y el servidor.

Figura 10

Memoria usada en un servidor en la multiplicación de matrices con dimensión variable



https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 176 - 184

Figura 11

Tiempo de ejecución de la multiplicación de matrices de una dimensión de 49,000 x 49,000

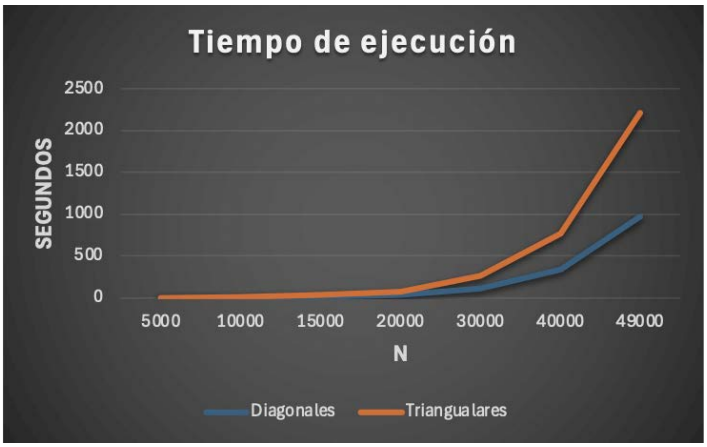
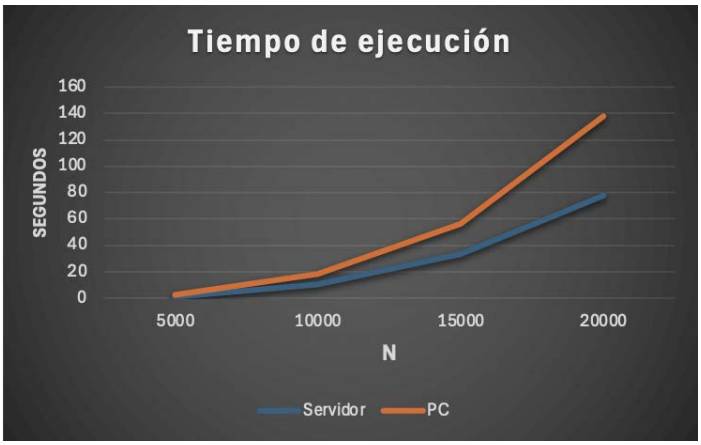


Figura 12

Comparativa de los tiempos de ejecución entre la PC y el servidor



En la tercera prueba, se usaron tres servidores con 12 procesadores cada uno. En la Tabla 3, se puede apreciar el tiempo de ejecución de la multiplicación de matrices, con transformaciones previas, de una dimensión de 49,000 x 49,000 con diferente cantidad de procesadores distribuidos, así como su *speedup* y eficiencia.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 177 - 184

Tabla 3

Tiempo de ejecución, speedup y eficiencia

Procesadores	Tiempo de ejecución (s). Matrices diagonales	Tiempo de ejecución (s). Matrices triangulares	Speedup	Eficiencia
1	11663	26445	1	1
4	2920	6621	3.99	0.999
8	1469	3330	7.93	0.992
12	977	2216	11.93	0.995
16	736	1668	15.84	0.990
20	592	1342	19.70	0.985
24	495	1123	23.56	0.982
28	430	974	27.12	0.969
32	369	838	31.60	0.988
36	332	752	35.13	0.976

En la Figura 13, se observa la gráfica de tiempo de ejecución usando diferente número de procesadores.

Figura 13

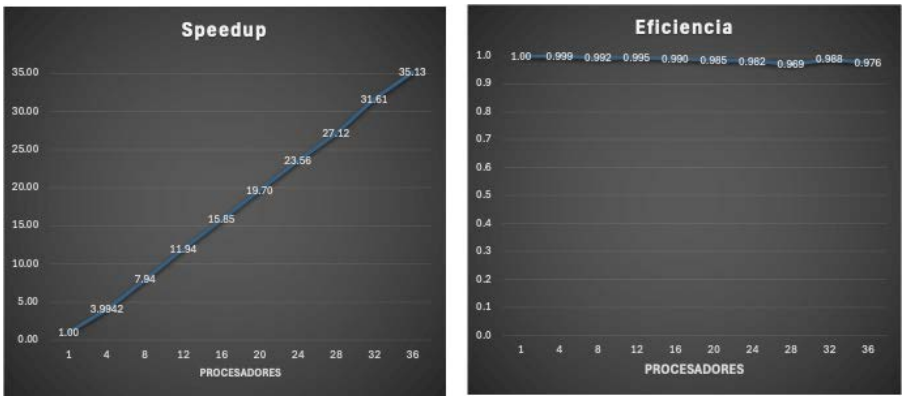
Tiempo de ejecución para diferente número de procesadores

En la Figura 14, se observa el *speedup* y la eficiencia del manejo de matrices usando 36 núcleos de 3 servidores del clúster de MATLAB.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 178 - 184

Figura 14

Speedup y eficiencia en el clúster MATLAB



Nota. Cabe señalar que esta configuración de clúster ha sido utilizada por tres usuarios, entre ellos un investigador, un tesista y un técnico académico.

En este caso, las gráficas ya no son tan lineales como los resultados de la PC debido al proceso de comunicación de los *workers* a través de la red. También se observa que MATLAB *parallel server* es una herramienta útil para escalar un código de forma distribuida.

4. CONCLUSIONES

Los investigadores del IIMAS de la UNAM necesitaban recursos computacionales que una computadora de escritorio no puede ofrecerles. Por lo anterior, se puso en marcha un clúster de servidores con MATLAB distribuido entre ellos a través de una red de comunicaciones que permitió satisfacer las necesidades de cómputo científico. En este caso, han sido beneficiados tres usuarios de la línea de investigación en cómputo de alto rendimiento.

El *software* utilizado, junto con los recursos de *hardware*, permitieron ejecutar procesos para la resolución de problemas basados en operaciones de matrices. Lo anterior permitió utilizar automáticamente los múltiples núcleos presentes de los servidores sin necesidad de configuración adicional por parte de los usuarios. Esta abstracción del código permitió una mayor flexibilidad para los investigadores que no desean emplear tanto tiempo en verificar las comunicaciones entre núcleos en los tres nodos conectados a la red. Si se compara el tiempo y facilidad de configuración y escalamiento de MATLAB con otro *software*, como Octave, la ventaja es del primero. Se usó MATLAB para la distribución de procesos en el clúster aprovechando la facilidad de licencia que está disponible institucionalmente.

La instalación de MATLAB en una sola computadora es muy sencilla; sin embargo, es importante tomar en cuenta que, para la instalación y configuración en un ambiente distribuido en red, se necesita una licencia especial y tener un nodo específico para instalarla. En este trabajo, se usaron tres nodos de cálculo; sin embargo, una vez instalado y configurado el primer nodo de trabajo, se puede escalar el tamaño del clúster con relativa facilidad.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 **Vol. 4, Núm. 2.** abril-junio 2026, págs. 179 - 184

REFERENCIAS

Delfino, M. (2020). Distributed Computing. En C.W. Fabjan & H. Schopper (Eds.), *Particle Physics Reference Library* (pp. 613-644). Springer. https://doi.org/10.1007/978-3-030-35318-6_14

Foster, I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley. https://edoras.sdsu.edu/~mthomas/docs/foster/Foster_Designing_and_Building_Parallel_Programs.pdf

Gu, R., Tang, Y., Tian, C., Zhou, H., Li, G., Zheng, X., & Huang, Y. (2017). Improving Execution Concurrency of Large-Scale Matrix Multiplication on Distributed Data-Parallel Platforms, *IEEE Transactions on parallel and distributed systems, 28*(9), 2539 – 2552.

HouZhen, W., Yan, G., & HuanGuo, Z. (2020). A Method of Ultra-Large-Scale Matrix Inversion Using Block Recursion. *Information, 11*(11), 523. https://doi.org/10.3390/info11110523

Lim, R., Lee Y., Kim, R., & Choi, J. **(**2018). An implementation of matrix–matrix multiplication on the Intel KNL processor with AVX-512. *Cluster Computing. 21*, 1785–1795. https://doi.org/10.1007/s10586-018-2810-y

MathWorks. (s. f.). *Distributed arrays*. Recuperado de https://la.mathworks.com/help/parallel-computing/distributed-arrays.html

MathWorks. (s.f.). *Get started with MATLAB*. Recuperado de https://la.mathworks.com/help/matlab/getting-started-with-matlab.html

MathWorks. (s. f.). *MATLAB Parallel Server* [Documento digital]. Recuperado de https://la.mathworks.com/help/pdf_doc/matlab-parallel-server/matlab-parallel-server.pdf

MathWorks. (s. f.). *Parallel Computing Toolbox* [Documento digital]. Recuperado de https://la.mathworks.com/help/pdf_doc/parallel-computing/parallel-computing.pdf

Westrick, S., Fluet, M., Rainey, M., & Acar U. (2024). Automatic Parallelism Management. *Proceedings of the ACM on Programming Languages, 8*, 1118 – 1149. https://doi.org/10.1145/3632880

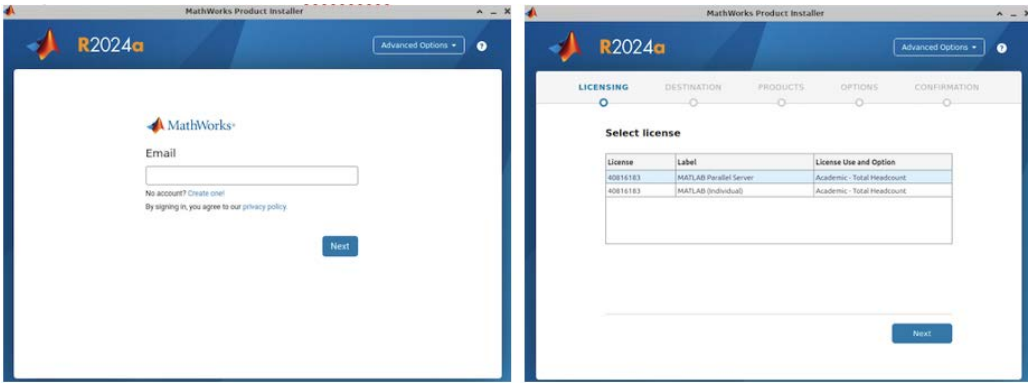
https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 180 - 184

ANEXO A. PASOS SIGUIENTES DE LA INSTALACIÓN

La Figura 15 y Figura 16 muestran la secuencia de instalación de *MATLAB Parallel Server.*

Figura 15

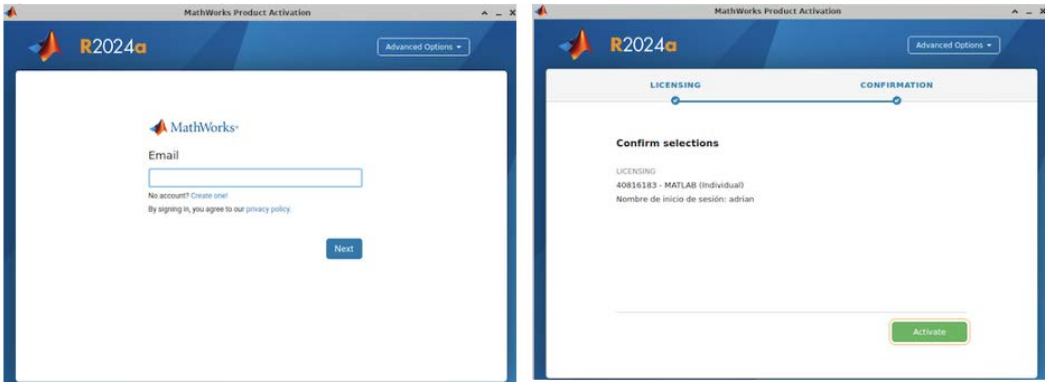
Solicitud de correo institucional y selección de la licencia MATLAB Parallel Server



Nota. En un paso posterior, se seleccionará el archivo de licencia.

Figura 16

Activación de MATLAB



Al ejecutar MATLAB, es importante activarlo con la licencia de red. Para información detallada de la instalación, referirse al manual de MATLAB *parallel server.*

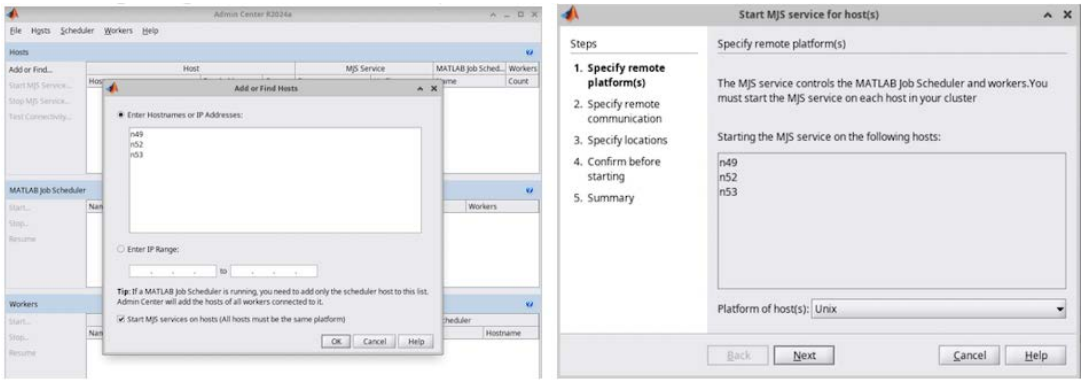
https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 181 - 184

ANEXO B. PASOS DE CONFIGURACIÓN

De la Figura 17 a la Figura 23, se muestra la secuencia de configuración de MATLAB *Parallel Server.*

Figura 17

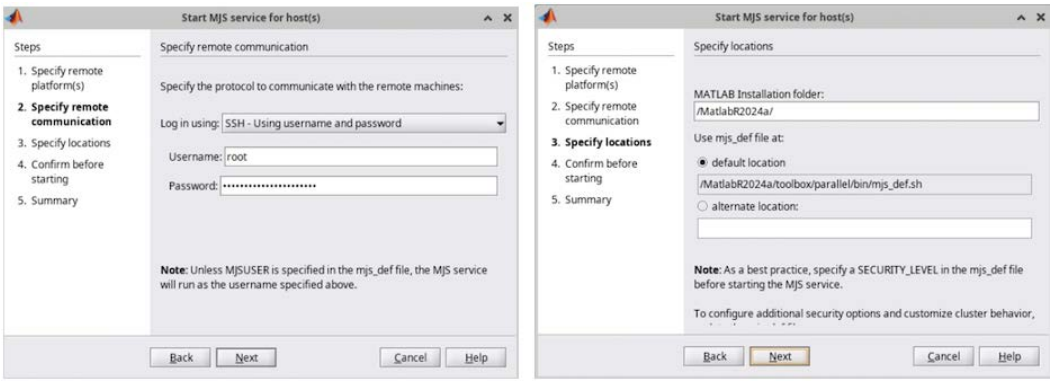
Nodos y sistema operativo



Nota. En el apartado "*Add or Find*…", se agregan los equipos que ya tengan instalado MATLAB *Parallel Server* (en este caso, n49, n52 y n53) y se verifica la plataforma.

Figura 18

Usuario y ruta de instalación



Nota. En este paso, se configura el usuario que ejecuta el *software* y el directorio donde está instalado MATLAB.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 182 - 184

Figura 19

Confirmación de los datos y verificación de inicio de los servicios del MJS (MATLAB Job Scheduler)

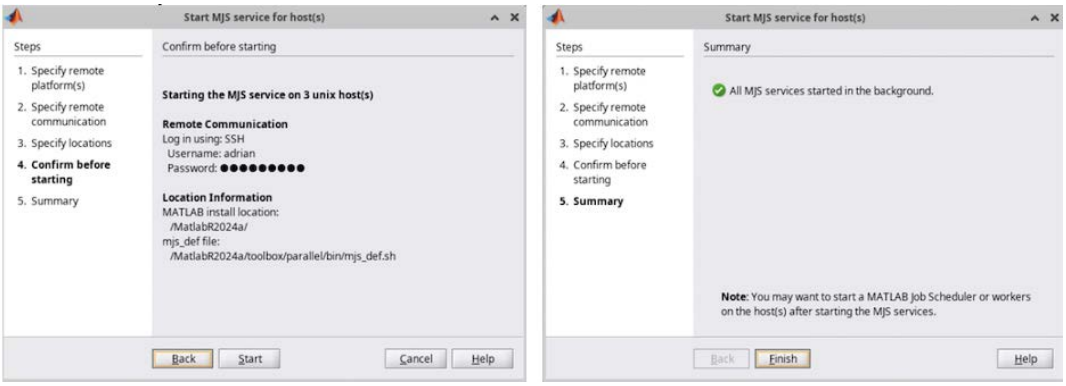
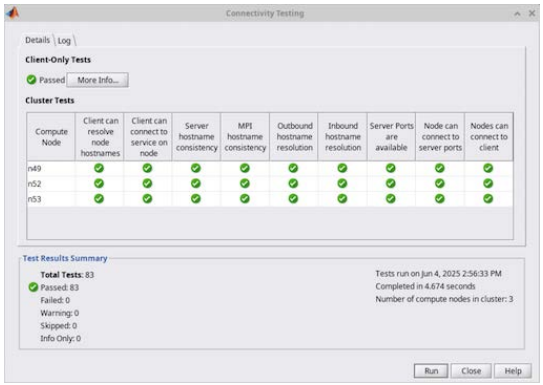


Figura 20

Prueba de conectividad (Test Connectivity)



Nota. Al finalizar la sección de "*Hosts*", se prueba la conexión entre los nodos y se observa la verificación.

Se prosigue con la configuración de las secciones "MATLAB *Job Scheduler*" y "*Workers*". En la Figura 21, se observa que los servicios están ejecutándose.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 183 - 184

Figura 21

Vista de "admincenter" cuando la configuración funciona correctamente

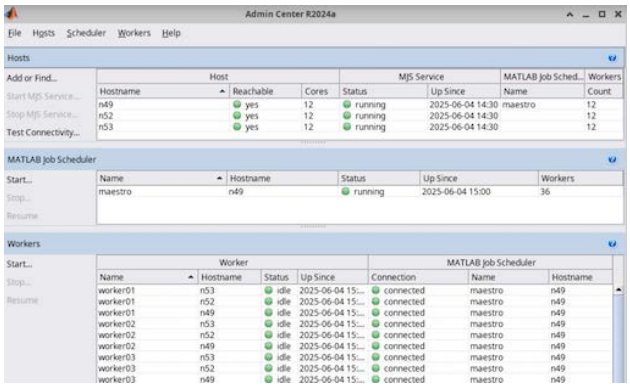
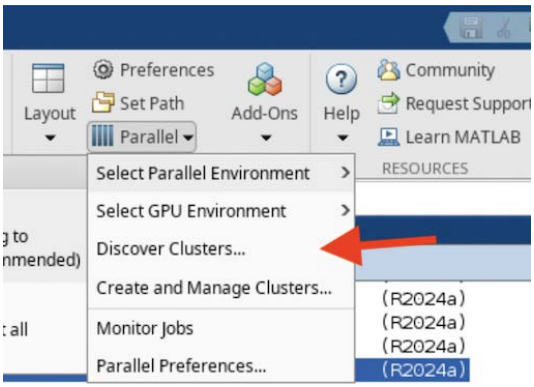


Figura 22

Agregar el perfil del nuevo clúster de MATLAB

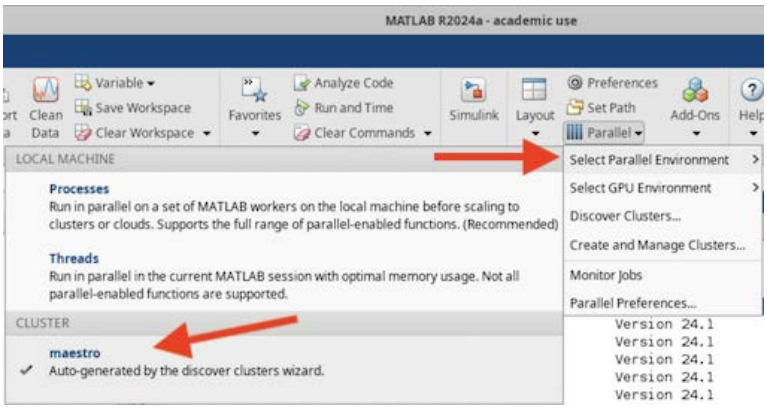


Nota. En la pestaña "*Parallel*" y sección "*Discover Clusters*…", se agrega la configuración del nuevo clúster.

https://doi.org/10.22201/dgtic.30618096e.2026.4.2.150 Vol. 4, Núm. 2. abril-junio 2026, págs. 184 - 184

Figura 23

Selección del nuevo clúster "maestro"



Nota. Finalmente se verifica la selección del nuevo clúster. En este caso, llamado "maestro".