Introducción a la computación paralela utilizando MATLAB en el clúster TK-II HPC

Este documento proporciona los pasos para configurar MATLAB a enviar trabajos a un clúster, recuperar resultados y depurar errores.

INSTALLACIÓN y CONFIGURACIÓN - Cliente MATLAB en el escritorio

Los scripts de integración para el clúster se encuentran *aquí*

Descarga el archivo ZIP e inicia MATLAB. El archivo ZIP debe descomprimirse en la ubicación devuelta al llamar a

```
>> userpath
```

Configura MATLAB para ejecutar trabajos paralelos en el clúster llamando a `configCluster`. `configCluster` solo necesita ser llamado una vez por cada versión de MATLAB .

```
>> configCluster
```

La presentación al clúster requiere credenciales SSH. Se te pedirá que ingreses un nombre de usuario y una contraseña o un archivo de identidad (clave privada). El nombre de usuario y la ubicación de la clave privada quedarán guardados en MATLAB para sesiones futuras.

Los trabajos ahora se enviarán por defecto al clúster en lugar a la maquina local.

NOTA: Para enviar trabajos a la maquina local en lugar del clúster, ejecuta el siguiente:

```
>> % Obtener una referencia a los recursos locales
>> c = parcluster('local');
```

CONFIGURACIÓN de Trabajos

Antes de enviar el trabajo, se pueden asignar varios parámetros, como cola de procesos, correo electrónico, tiempo de ejecución, etc. A continuación, se muestra una lista parcial de parámetros.

Consulta `AdditionalProperties` para ver la lista completa.

```
>> % Obtener una referencia al clúster
>> c = parcluster;

[REQUERIDO]

>> % Especificar la partición
>> c.AdditionalProperties.Partition = 'partition-name';

>> % Especificar el tiempo de ejecución (p.ej., 1 día, 5 horas, 30
minutos)
>> c.AdditionalProperties.WallTime = '1-05:30';
```

[**OPCIONAL**]

```
>> % Especificar la cuenta
>> c.AdditionalProperties.AccountName = 'account-name';

>> % Especificar una restricción
>> c.AdditionalProperties.Constraint = 'feature-name';

>> % Solicitar notificaciones por correo electrónico sobre el estado
del trabajo
>> c.AdditionalProperties.EmailAddress = 'user-id@universidad.edu.mx';

>> % Especificar el número de GPUs (predeterminado: 0)
>> c.AdditionalProperties.GPUsPerNode = 1;
>> c.AdditionalProperties.GPUCard = 'gpu-card';

>> % Especificar la memoria a utilizar por núcleo (predeterminado:
4gb)
>> c.AdditionalProperties.MemPerCPU = '6gb';

>> % Especificar núcleos por nodo
>> c.AdditionalProperties.ProcsPerNode = 4;

>> % Asignar el nodo de manera exclusiva (predeterminado: falso)
>> c.AdditionalProperties.RequireExclusiveNode = true;

>> % Usar reservación
>> c.AdditionalProperties.Reservation = 'reservation-name';
```

Guarda los cambios después de modificar las AdditionalProperties para que los cambios anteriores permanezcan entre sesiones de MATLAB.

```
>> c.saveProfile
```

Para ver los valores de las opciones de configuración actuales, muestra `AdditionalProperties`.

```
>> % Para ver las propriedades actuales
>> c.AdditionalProperties
```

Elimina un valor cuando ya no sea necesario.

```
>> % Desactivar las notificaciones por correo electrónico
>> c.AdditionalProperties.EmailAddress = '';
>> c.saveProfile
```

Trabajos INTERACTIVOS - cliente de MATLAB en el clúster

Para ejecutar un trabajo de grupo interactivo en el clúster, continúe usando parpool como antes

```
>> % Obtenga un identificador del clúster
>> c = parcluster;

>> % Abra un grupo de 64 trabajadores en el clúster
>> pool = c.parpool(64);
```

En lugar de ejecutarse localmente en la maquina local, el grupo ahora podrá ejecutarse en varios nodos del clúster.

```
>> % Ejecutar un parfor sobre 1000 iteraciones
>> parfor idx = 1:1000
      a(idx) = rand;
   end
```

Elimine el grupo cuando ya no sea necesario.

```
>> % Elimine el grupo
>> pool.delete
```

TRABAJOS POR LOTES INDEPENDIENTE

Utilice el comando `batch` para enviar trabajos asincrónicos al clúster. El comando `batch` devolverá un objeto de trabajo que se utiliza para acceder al resultado del trabajo enviado. Consulte la documentación de MATLAB para obtener más ayuda sobre lotes (`batch`).

```
>> % Obtenga un identificador del clúster
>> c = parcluster;

>> % Enviar el trabajo para consultar dónde se ejecuta MATLAB en el
clúster
>> job = c.batch(@pwd, 1, {}, 'CurrentFolder','.');

>> % Query job for state
>> job.State

>> % Si el estado finaliza, recupera los resultados.
>> job.fetchOutputs{:}

>> % Elimine el trabajo después de que los resultados ya no sean
requeridos
>> job.delete
```

Para obtener una lista de trabajos en ejecución o completados, llame a `parcluster` para devolver el objeto del clúster. El objecto del clúster almacena una serie de trabajos que están en cola para

ejecutarse, se están ejecutando, se han ejecutado o han fallado. Recupere y vea la lista de trabajos como se muestra a continuación.

```
>> c = parcluster;
>> jobs = c.Jobs
>>
>> % Obtenga un identificador del segundo trabajo de la lista
>> job2 = c.Jobs(2);
```

Una vez seleccionado el trabajo, obtengo los resultados como se hizo anteriormente.

`fetchOutputs` se utiliza para recuperar argumentos de salida de funciones; si llama a `batch` con un script, utiliza `load` en su lugar. Los datos que se han escruto en archivos en el clúster deben recuperarse directamente del sistema de archivos (por ejemplo, a través de sftp).

```
>> % Obtenga todos los resultados del segundo trabajo de la lista
>> job2.fetchOutputs{:}
```

TRABAJOS POR LOTES EN PARALELO

`Batch` también puede enviar trabajos en paralelo. Usemos el siguiente ejemplo para trabajo paralelo, que esta guardado como `parallel_example.m`.

```
function [sim_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
sim_t = toc(t0);

disp('Sim completed')

save RESULTS A

end
```

Esta vez, cuando utilice el comando `batch`, especifique también un argumento de MATLAB Pool.

```
>> % Obtenga un identificador del clúster
>> c = parcluster;
```

```
>> % Envíe un trabajo de grupo por lotes utilizando 4 trabajadores
para 16 simulaciones
>> job = c.batch(@parallel_example, 1, {16}, 'Pool',4, ...
       'CurrentFolder','.');

>> % Ver el estado actual del trabajo
>> job.State

>> % Obtener los resultados después de recuperar un estado finalizado
>> job.fetchOutputs{:}

ans =

     8.8872
```

El trabajo fue ejecutado en 8.89 segundos utilizando 4 trabajadores. Tenga en cuenta que estos trabajos siempre solicitaran N+1 núcleos de CPU, ya que se requiere un trabajador para administrar el trabajo por lotes y el grupo de trabajadores. Por ejemplo, un trabajo que necesita ocho trabajadores solicitara nueve núcleos de CPU.

Ejecuta la misma simulación, pero aumente el tamaño del grupo. Esta vez, para recuperar los resultados más tarde, realice un seguimiento del ID del trabajo.

Nota: Para algunas aplicaciones, habrá un rendimiento decreciente al asignar demasiados trabajadores, ya que los gastos generales pueden exceder el tiempo de cálculo.

```
>> % Obtenga un identificador del clúster
>> c = parcluster;

>> % Envíe un trabajo de grupo por lotes utilizando 8 trabajadores
para 16 simulaciones
>> job = c.batch(@parallel_example, 1, {16}, 'Pool',8, ...
       'CurrentFolder','.');

>> % Obtenga el ID del trabajo
>> id = job.ID

id =

     4

>> % Borrar trabajo del espacio de trabajo (como si se hubiera salido
de MATLAB)
>> clear job
```

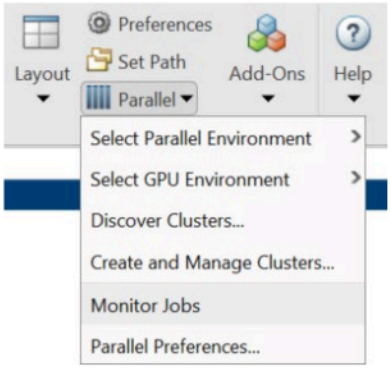
Con un identificador del clúster, el método `findJob` busca el trabajo con el ID de trabajo especificado.

```
>> % Obtenga un identificador del clúster
>> c = parcluster;
```

```
>> % Encuentra el trabajo anterior
>> job = c.findJob('ID', 4);

>> % Recuperar el estadio del trabajo
>> job.State

ans =

     finished

>> % Obtener los resultados
>> job.fetchOutputs{:};

ans =

     4.7270
```

El trabajo ahora se ejecuta en 4.83 segundos con ocho trabajadores. Ejecuta el código con distinto número de trabajadores para determinar el número ideal de trabajadores.

Alternativamente, para recuperar los resultados del trabajo a través de una interfaz gráfica de usuario, utilice Job Monitor (Parallel > Monitor Jobs).



FUNCIONES DE AYUDA

Función	Descripción
clusterFeatures	Lista de características/limitaciones del clúster
clusterGpuCards	Lista de tarjetas GPU del clúster
clusterPartitionNames	Lista de nombres de particiones del clúster
disableArchiving	Modifique el archivo de archivos para resolver el problema de duplicación de archivos
fixConnection	Restablecer la conexión del clúster (por ejemplo, después de la

	reconexión de la VPN)
willRun	Explicar por qué el trabajo está en la cola

DEPURACIÓN

Si un trabajo en serie produce un error, llama al método `getDebugLog` para ver el archivo de registro de errores. Al enviar un trabajo independiente, especifica la tarea.

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

Para trabajos en Pool, solo especifica el objeto de trabajo.

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

Al solucionar problemas de un trabajo, el administrador del clúster puede solicitar el ID del programador del trabajo. Esto se puede obtener llamando a `getTaskSchedulerIDs`.

```
>> job.getTaskSchedulerIDs()

ans =

      25539
```

PARA APRENDER MAS

Para aprender más sobre la caja de herramientas de computación paralela de MATLAB, consulte estos recursos:

- Resumen de la computación paralela
- Documentación de Computación Paralela
- Ejemplos de Programación en Computación Paralela
- Tutoriales de Computación Paralela
- Videos de Computación Paralela
- Seminarios Web de Computación Paralela