Guía de Usuario – Cluster CNCA/CeNAT

Tutorial de uso de Kabré

Kabré es una palabra del lenguaje Ngäbe que significa *un puñado*. Este significado entra dentro de la actual composición del clúster, la cual cuenta con múltiples arquitecturas en paralelo. A lo largo de este tutorial usted aprenderá cómo Kabré se compone, cómo conectarse al clúster, enviar trabajos, recibir resultados y lo necesario sobre módulos de ambiente.

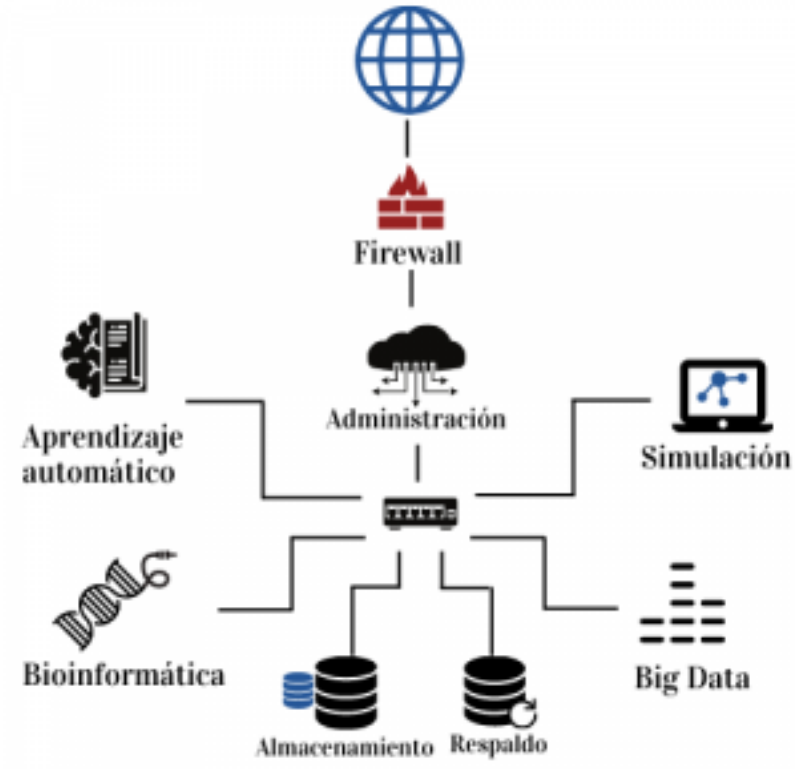
Requerimientos

Para completar este tutorial usted requerirá un cliente SSH. En plataformas Unix y Linux comúnmente existe un emulador de terminal capaz de establecer sesiones de SSH. En plataformas de Windows usted deberá descargar un programa que haga de cliente SSH, como Putty o similar.

Además, requerirá una cuenta activa en Kabré y credenciales válidas. Si no tiene una, por favor diríjase a la página de registro y llene sus datos para solicitar una nueva cuenta, o escríbanos a cnca@cenat.ac.cr explicando su situación.

Entendiendo la composición de Kabré

La siguiente imagen muestra un diagrama de red de Kabré. A continuación discutiremos un poco sobre los mayores componentes del mismo.



Nodos de acceso

Estos nodos, llamados también *login,* son un tipo de área de trabajo compartido. Cuando usted accede Kabré, se le asigna uno de los nodos login. Algunas tareas comunes que se realizan aquí son:

- Crear y editar archivos
- Crear directorios y mover archivos
- Copiar archivos desde y hacia su computadora
- Compilar código
- Enviar trabajos
- Administrar sus trabajos activos

Código paralelo o tareas pesadas corriendo en los nodos login son considerados comportamientos no deseados.

Nodos de aprendizaje automático (nukwa)

Nukwä significa *pájaro* en lenguaje Ngäbe. Hay 8 nodos nukwa, 4 de estos nodos cuentan con una GPU NVIDIA Tesla K40. El host tiene un Intel Xeon con 4 núcleos a 3,2 GHz sin hyper-threading y 16 GB de RAM. Además, hay otros 4 nodos nukwa con una GPU NVIDIA Tesla V100, 24 núcleos a 2.20GHz, 2 hilos por núcleo y 32 GB de RAM.

Solo las aplicaciones con un uso intensivo de GPU obtendrían una aceleración relevante con nodos nukwa.

Nodos de simulación (nu)

Nu significa *perro* en lenguaje Ngäbe. Estos nodos son el caballo de guerra de Kabré, 32 nodos Intel Xeon Phi KNL, cada uno con 64 núcleos a 1,3 GHz y 96 GB de RAM; cada núcleo tiene dos unidades AVX.

Si la aplicación se puede dividir en muchas partes pequeñas y utilizar vectorización, esta arquitectura puede proporcionar aceleraciones significantes.

Nodos de Big Data (andalan)

Andalan significa gallo en lenguaje Ngäbe. Dos de estos cinco nodos cuentan con un Intel Xeon, cada uno con 24 núcleos a 2,20 GHz, 2 hilos por núcleo y 64 GB de RAM. El tercer nodo cuenta con un Intel Xeon con 16 núcleos a 2,10 GHz, 2 hilos por núcleo y 64 GB de RAM. Un cuarto nodo cuenta con 10 núcleos, 2 hilos por núcleo, a 2.20 GHz y 32 GB de RAM. El último nodo tiene 24 núcleos a 2,40 GHz, 2 hilos por núcleo y 128 GB de RAM.

Los nodos andalan funcionan bastante bien para herramientas secuenciales que necesitan potencia de procesamiento.

Nodos bioinformáticos (dribe)

Dribe significa *alacrán* en lenguaje Ngäbe. Dribe tiene siete nodos, uno que cuenta con un Intel Xeon con 36 núcleos a 3,00 GHz, 2 hilos por núcleo y 1024 GB de

RAM, y los demás con 18 núcleos a 3,00 GHz, 2 hilos por núcleo y 512 GB de RAM.

Los nodos Dribe funcionan bien para herramientas que requieren una alta demanda de memoria.

Resumen de nodos computacionales

Interactuando con Kabré

En esta sección cubriremos las conexiones ssh, las claves ssh y cómo copiar archivos.

Conexiones SSH y claves SSH

Para empezar, abra un emulador de terminal y abra una sesión ssh escribiendo

$ ssh [user]@kabre.cenat.ac.cr

Recuerde cambiar [*user]* por su nombre de usuario. Escriba su contraseña cuando se le solicite. Se registrará en algún nodo de inicio de sesión. Este es un terminal típico de Linux, pruebe algunos comandos conocidos, como `ls`, `cd`, `mkdir` y otros.

Una clave SSH es un archivo que mantendrá su conexión segura y evita que escriba su contraseña cada vez que inicie sesión. Este archivo está comúnmente vinculado a una computadora, normalmente debe generar una para la computadora que utilizará para interactuar con Kabré.

Para generar una clave SSH, en su computadora local (computadora portátil, estación de trabajo ...) abra un terminal, vaya a su directorio de inicio y escriba

$ ssh-keygen -t rsa -C "your_email@example.com"

y siga las instrucciones. Si elige las opciones predeterminadas, la nueva clave estará en ~/.ssh/, ahora tiene que copiar la clave pública a Kabré, para hacerlo escriba

$ scp ~/.ssh/id_rsa.pub your_user@kabre.cenat.ac.cr:~

Ahora, dentro de una sesión ssh en Kabré, escriba:

$ cat id_rsa.pub >> .ssh/authorized_keys
$ rm id_rsa.pub

Alternativamente, puede ejecutar este procedimiento en un solo comando si está disponible en su estación de trabajo, de la siguiente manera:

$ ssh-copy-id your_user@kabre.cenat.ac.cr

Ahora, si abre un nuevo terminal y escribe ssh your_user@kabre.cenat.ac.cr se registrará sin solicitar la contraseña. Esto se debe a que Kabré tiene la clave ssh pública de su computadora. También es conveniente para su computadora tener la clave pública de Kabré, simplemente adjúntela a autorized_keys en su computadora local, así:

```
$ scp user@kabre.cenat.ac.cr:~/.ssh/id_rsa.pub .
$ cat id_rsa.pub >> ~/.ssh/autorized_keys
$ rm id_rsa.pub
```

Sistema de archivos de Kabré

Contamos con dos directorios: */home* and */work*.

Directorio	Cuota	Propósito	Respaldo
/home/userid	10GB	scripts y datos importantes	mensual
/work/userid	100GB	datos temporales	NO

Para saber la cantidad de espacio disponible que queda en cada directorio, puede usar el siguiente comando:

```
$ df -h /home/userid
```

Or:

```
$ df -h /work/userid
```

NOTAS:

- El usuario es responsable de administrar los directorios de **inicio y de trabajo.** El primero está destinado a programas y datos confidenciales, mientras que el segundo está destinado a datos masivos y temporales.
- La capacidad del directorio de trabajo se puede ampliar a petición. Debe proporcionarse una justificación clara del espacio adicional necesario. Envíe su solicitud a jumana@cenat.ac.cr.
- Aunque se realiza una copia de seguridad del directorio de inicio mensualmente, recomendamos a los usuarios que trabajen con un sistema de control de versiones (v.g. git) para sus scripts y códigos fuente.

Copiar archivos entre su computadora y Kabré

El comando scp es similar al comando cp, copia archivos de un origen a un destino a través de una sesión SSH. Tiene la siguiente sintaxis:

```
$ scp [user]@[host][path]origin_file [user]@[host][path]destiny_file
```

Tenga en cuenta que este comando es solo para un solo archivo, si necesita cargar un directorio completo, agregue la opción -r al comando, como se muestra a continuación:

```
$ scp -r [user]@[host][path]origin_directory [user]@[host][path]destiny_directory
```

Los valores por defecto son:

- user: su usuario local
- host: host local
- path: directorio de trabajo actual

El comando scp debe ejecutarse en su máquina local, no en Kabré. Tal vez su aplicación genere muchos archivos de visualización y desee descargar esos archivos a su computadora, recuerde que ~ significa el directorio de inicio y * coincide con cualquier secuencia de caracteres, usando estos conceptos:

$ scp user@kabre.cenat.ac.cr:~/application/output/*.viz ~/app/results/visualization

O tal vez desee cargar un archivo de parámetros para usar en una simulación, debería hacer lo siguiente:

$ scp ~/research/app/parameters.dat user@kabre.cenat.ac.cr:~/application/input

Dar permisos a otro usuario sobre un directorio

El primer paso es conocer el uid (identificación de usuario) del usuario con el que desea compartir la propiedad, para obtener el uid, ejecute el siguiente comando:

$ id username

A continuación, una vez conocido el uid, se puede proceder a otorgar permisos sobre un solo archivo o directorio, aquí se puede elegir qué permisos otorgar, podrían ser: R (para lectura), W (para escritura) y / o X (para ejecutar).

Si desea otorgar todos los permisos al usuario cuyo uid es “[uid]” sobre el directorio “[/ ruta / directorio]” (recuerde sustituir estos valores por los que necesita), puede usar el siguiente comando:

$ nfs4_setfacl -a A::[uid]:RWX [/path/directory]

Note que esto daría todos los permisos (ya que usamos RWX) a [uid] solo al directorio [/ ruta / directorio], pero no a sus subdirectorios (estas son las carpetas dentro de / ruta / directorio). Tenga cuidado al otorgar todos los permisos a otro usuario, ya que tendrá la capacidad de eliminar y modificar libremente.

Si desea incluir todos los subdirectorios, este mismo comando se puede ejecutar de forma recursiva simplemente agregando -R, así:

$ nfs4_setfacl -R -a A::[uid]:RWX [/path/directory]

Por último, si desea eliminar los permisos RWX a un “[uid]” sobre el directorio “[/ ruta / directorio]”, puede usar:

$ nfs4_setfacl -a D::[uid]:RWX [/path/directory]

Entendiendo el sistema de colas de Kabré

Los nodos de inicio de sesión son adecuados para tareas ligeras, como se mencionó anteriormente: editar archivos, compilar, copiar archivos, etc. Se espera que las tareas pesadas se ejecuten en los nodos Nu, Nukwa, Andalan o Dribe. Para hacer cumplir un reparto justo de recursos entre los usuarios, su tarea debe enviarse a un sistema de cola. Es como formarse en el banco, una vez que su tarea llega al principio de la fila, se le otorgarán todos los recursos solicitados y se ejecutará hasta que se complete o hasta que consuma su franja de tiempo.

Actualmente, hay diferentes colas para cada componente en Kabré, eso significa que no puede mezclar nodos Nukwa y nodos Nu en un solo trabajo, por ejemplo. La siguiente tabla muestra todas las colas disponibles:

Partición (Cola)	Plataforma	Número de nodos	Asignación de tiempo
nu	Xeon Phi KNL	1	72 h
nu-debug	Xeon Phi KNL	1	8 h
nu-wide	Xeon Phi KNL	12	24 h
nu-long	Xeon Phi KNL	1	744 h
nukwa	GPU	1	72 h
nukwa-debug	GPU	1	8 h
nukwa-wide	GPU	2	24 h
nukwa-long	GPU	1	168 h
andalan	Xeon	1	72 h
andalan-debug	Xeon	1	8 h
dribe	Xeon	1	72 h
dribe-long	Xeon	1	744 h
dribe-debug	Xeon	1	8 h

El proceso de envío de un trabajo en Kabré se puede dividir en cuatro pasos: escribir un archivo SLURM, poner su trabajo en cola, monitorear los trabajos y recuperar los resultados.

Escribiendo un archivo SLURM

Este archivo de configuración le dice al sistema de colas todo lo que necesita saber sobre su trabajo, para que pueda colocarse en la cola correcta y ejecutarse. Vamos a probarlo con un ejemplo de trabajo mínimo. A continuación se muestra un código C que se aproxima al valor de pi utilizando un método de Montecarlo. Inicie sesión en Kabré, copie el texto en un archivo y guárdelo con el nombre pi_threads.c.

```
#include <pthread.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <stdio.h>

typedef struct {
```

```
 int num_of_points;
 int partial_result;
} pi_params;

void * calc_partial_pi(void * p){
 pi_params * params = p;
 int count = 0;
 double r1, r2;
 unsigned int seed = time(NULL);

for(int i = 0; i < params->num_of_points; ++i){
 r1 = (double)rand_r(&seed)/RAND_MAX;
 r2 = (double)rand_r(&seed)/RAND_MAX;
 if(hypot(r1, r2) < 1)
 count++;
 }
 params->partial_result = count;
 pthread_exit(NULL);
}


int main(int argc, char * argv[]){

if(argc != 3){
 printf("Usage: $ %s num_thread num_points\n", argv[0]);
 exit(0);
 }

int num_threads = atoi(argv[1]);
 int num_points = atoi(argv[2]);
 int num_points_per_thread = num_points / num_threads;

pthread_t threads[num_threads];
 pi_params parameters[num_threads];

for(int i = 0; i < num_threads; i++){
 parameters[i].num_of_points = num_points_per_thread;
 pthread_create(threads+i, NULL, calc_partial_pi, parameters+i);
 }

for(int i = 0; i < num_threads; i++)
 pthread_join(threads[i], NULL);

double approx_pi = 0;
 for(int i = 0; i < num_threads; i++)
 approx_pi += parameters[i].partial_result;
 approx_pi /= (num_threads * num_points_per_thread) / 4;

printf("Result is %f, error %f\n", approx_pi, fabs(M_PI-approx_pi));

}
```

Actualmente, se encuentra en un nodo de inicio de sesión, por lo que está bien

compilar el código allí, hágalo escribiendo:

$ gcc -std=gnu99 pi_threads.c -lm -lpthread -o pi_threads

El siguiente es un archivo SLURM de ejemplo. Todas las líneas que comienzan con #SBATCH son comandos de configuración para el sistema de colas. Las opciones que se muestran aquí son las más comunes y posiblemente las únicas que necesitará.

Configuración	Descripción
–job-name=<nombre del trabajo>	Nombre específico del trabajo
–output=<nombre del resultado>	El nombre con el que sale el trabajo
–partition=<nombre de partición>	En qué cola debería ejecutarse
–ntasks=<numero>	Numero de procesos a ejecutar
–time=<HH:MM:SS>	Duración aproximada del trabajo

El cuerpo de un archivo SLURM es un código bash. Copie el ejemplo en un archivo llamado pi_threads.slurm

```
#!/bin/bash
#SBATCH --job-name=pi_threads
#SBATCH --output=result.txt
#SBATCH --partition=nu
#SBATCH --ntasks=1
#SBATCH --time=00:10:00

module load gcc/7.2.0

srun ./pi_threads 64 100000000000
```

Nota: Los argumentos de la línea de comando 64 y 100000000000 son los parámetros específicos necesarios para ejecutar pi threads.

Ahora, desde la línea de comando, invoque al remitente de colas:

$ sbatch pi_threads.slurm

¡Y eso es todo! Su trabajo se pondrá en cola y se ejecutará, en este caso, en un nodo Xeon Phi.

Monitoreando sus trabajos activos

Una forma pasiva de monitorear sus trabajos es indicarle a SLURM que envíe un correo electrónico cuando haya terminado. Esto se puede configurar en el archivo SLURM usando las siguientes opciones:

Configuración	Descripción

–mail-user=<email>	Dónde enviar alertas de correo
–mail-type= <BEGIN|END|FAIL|REQUEUE|ALL>	Cuando enviar alertas de correo

#SBATCH --mail-user=example@mail.com
#SBATCH --mail-type=END,FAIL

Una forma pasiva de monitorear sus trabajos es indicarle a SLURM que envíe un correo electrónico cuando haya terminado. Esto se puede configurar en el archivo SLURM usando las siguientes opciones:

Configuración	Descripción
squeue -u <username>	Verificar trabajos para un usuario específico
sinfo	Mostrar todos los nodos (con atributo)
scontrol show job <job_id>	Estado de un trabajo en particular
scancel <job_id>	Eliminar trabajo

Para mostrar detalles sobre el estado de un trabajo:

$ squeue -j <jobid>

Para mostrar detalles sobre el estado de un trabajo:

$ watch -n <num_seconds> squeue -j <jobid>

Estados de trabajo válido

Para comprender los códigos de estado de trabajo que puede encontrar, verifique lo siguiente:

Código	Estado
CA	Cancelado
CD	Completado
CF	Configurando
CG	Completando
F	Fallo
NF	Fallo de nodo
PD	Pendiente
R	Corriendo
TO	Timeout

Recuperando resultados

De forma predeterminada, cada trabajo generará un archivo de salida con un nombre como en el siguiente ejemplo:

result.txt

Puede copiar este archivo a su computadora local o ejecutar otro script para procesar posteriormente la salida.

Trabajos interactivos

A veces desea tener acceso directo a algún nodo. Usar ssh directamente es una mala práctica, porque el sistema de cola podría enviar el trabajo de otra persona para que se ejecute en el nodo que está utilizando actualmente. La forma educada de solicitar acceso directo es a través de un trabajo interactivo que lo pondrá en una venta interactiva en un nodo de cómputo. Esto le permite experimentar con diferentes opciones y variables que proporcionarán retroalimentación inmediata.

Para solicitar una cola interactiva en un nodo Nu, puede utilizar:

$ salloc

Como alternativa, el siguiente comando abre un trabajo interactivo en un nodo Nukwa, Andalan o Dribe y reserva una GPU para su uso:

$ srun --partition=[node] --pty --preserve-env $SHELL

Tenga en cuenta en el comando anterior que debe sustituir [node] por: nukwa-debug, andalan-debug o dribe-debug, según sus necesidades.

Módulos de ambiente

Diferentes usuarios tienen diferentes necesidades, a veces esas necesidades podrían ser conflictivas, por ejemplo, varias versiones de la misma biblioteca. Estas situaciones se resuelven con módulos de entorno. Un caso típico son las diferentes versiones de Python. Para ejemplificar, pregunte por la cola en la que se debe ejecutar escribiendo:

$ SBATCH --partition=nu

Continúe y escriba $ python, debe ingresar al intérprete de Python predeterminado, el encabezado debe ser así:

```
Python 2.7.5 (default, Nov 6 2016, 00:28:07)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
```

Además del intérprete predeterminado, puede ejecutar Intel Distribution para Python, una compilación ajustada específicamente con paquetes que se usan comúnmente en informática científica. Para obtener intel python, escriba:

module load intelpython/3.5

Ahora, escriba de nuevo $ python, obtendrá un encabezado diferente:

```
Python 3.5.2 |Intel Corporation| (default, Oct 20 2016, 03:10:33)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-15)] on linux
Type "help", "copyright", "credits" or "license" for more information.
Intel(R) Distribution for Python is brought to you by Intel Corporation.
Please check out: https://software.intel.com/en-us/python-distribution
>>>
>>>
```

Para comprobar qué módulos ya están cargados, escriba

$ module list

Para obtener una lista de todos los módulos disponibles, escriba

$ module avail

Detrás de escena, el comando del módulo solo configura rutas, alias y otras variables de entorno, por lo que los módulos se cargan solo para la sesión de shell actual. Puede solicitar módulos específicos en sus trabajos, simplemente agregue las líneas “module load module_name” al cuerpo del archivo SLURM, debajo de todas las líneas #SBATCH y antes de ejecutar su programa.