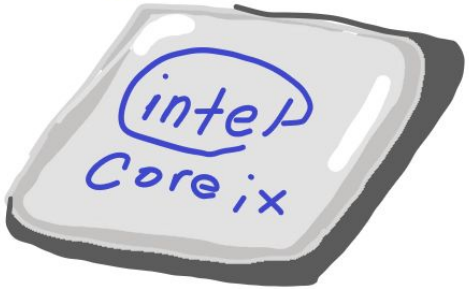
NLHPC
National Laboratory for High Performance Computing Chile

Curso Avanzado NLHPC

Ley de Moore



Aproximamente cada dos años se duplica el numero de transistores en un microprocesador

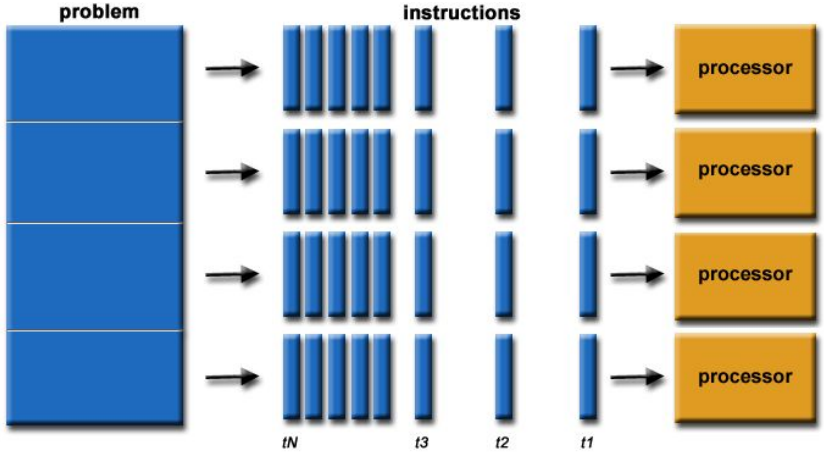
GORDON MOORE, ES CO- FUNDADOR DE INTEL

www.noticomic.com

NLHPC
National Laboratory for High Performance Computing
Chile

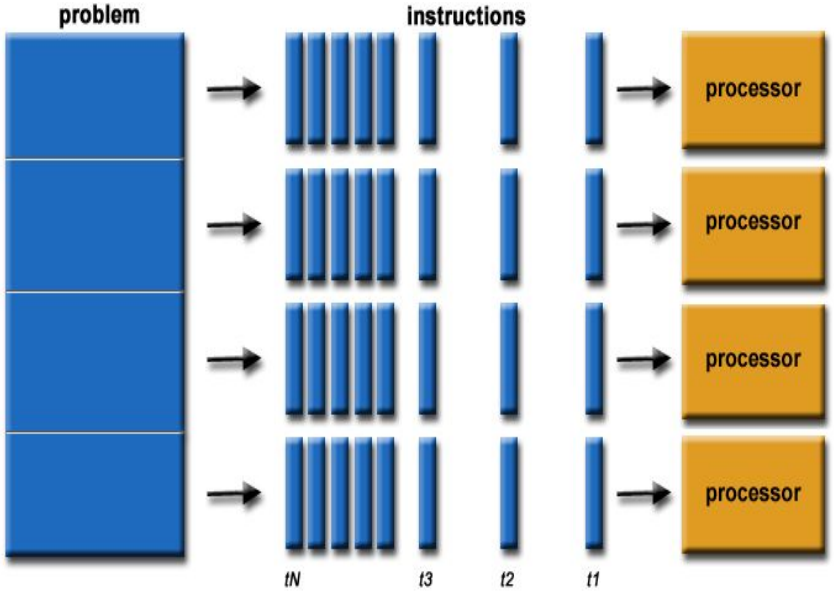
Computación Paralela

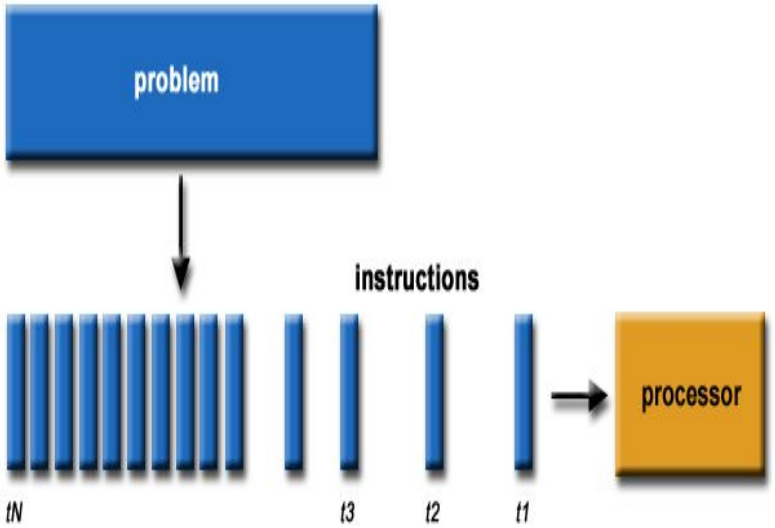
- Uso simultáneo de múltiples recursos computacionales
- Un problema se divide en partes discretas que se pueden resolver simultáneamente
- Cada parte se descompone en una serie de instrucciones
- Las instrucciones de cada parte se ejecutan simultáneamente en diferentes procesadores
- Se emplea un mecanismo global de control y coordinación



NLHPC
National Laboratory for High Performance Computing Chile

Computación Paralela Vs Computación Serial

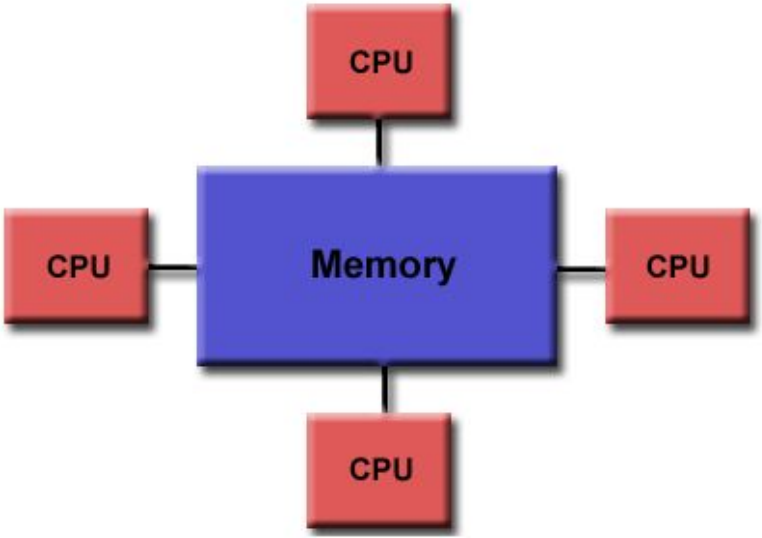




NLHPC
National Laboratory for High Performance Computing Chile

Arquitecturas de memoria de computación Paralela: Memoria Compartida

- Los procesos comparten un espacio de memoria común
- Escriben y leen de manera asíncrona
- No es necesario especificar cómo se comunican los datos entre las tareas
- Se usan semáforos o locks para controlar el acceso a la memoria compartida



NLHPC
National Laboratory for High Performance Computing Chile

OpenMP

- Es una interfaz de programación de aplicaciones para la programación multiproceso de memoria compartida
- Permite añadir concurrencia: Operar actividades al mismo tiempo
- Se compone de:
  - Directivas de compilación
  - Rutinas de biblioteca
  - Variables de entorno
- Modelo de programación portable y escalable
- Proporciona a los desarrolladores una interfaz simple y flexible para el desarrollo de aplicaciones paralelas

Ejecución de Simulaciones en Slurm - Trabajos paralelos (OpenMP)

- Los trabajos paralelos diseñados para ejecutarse en un sistema multi-core (shared memory) requieren especificar el número de núcleos (-c 44) a utilizar:

```
#!/bin/bash
# ----------------SLURM Parameters----------------
#SBATCH -J paralelo_openmp
#SBATCH -p general
#SBATCH -n 1
#SBATCH -c 44
#SBATCH --mem=192000
#SBATCH --mail-user=eguerra@nlhpc.cl
#SBATCH --mail-type=ALL
#SBATCH -o paralelo_openmp_%j.out
#SBATCH -e paralelo_openmp_%j.err

#-----------------Toolchain---------------------------
ml purge
ml intel/2019b
# ----------------Módulos-----------------------------
ml Programa
# ----------------Comandos----------------------------
./binario entrada
```

NLHPC
National Laboratory for High Performance Computing Chile

Ejercicio 1

1. Descarga el siguiente código: pi_omp.c en tu directorio de trabajo.

2. Teniendo en cuenta que es un código OpenMP, compílalo mediante el comando `icc pi_omp.c -o pi_omp -qopenmp`.

3. Crea un script de ejecución para lanzarlo con sbatch, con las siguientes consideraciones: a) La partición a lanzar es slims. b) Ejecuta 1 proceso. c) Este único proceso debe tener 20 hilos de ejecución. d) Supón que cada hilo ocupa 2300MB de memoria RAM. e) El binario a ejecutar es `pi_openmp`.

4. Anota el script que has usado.

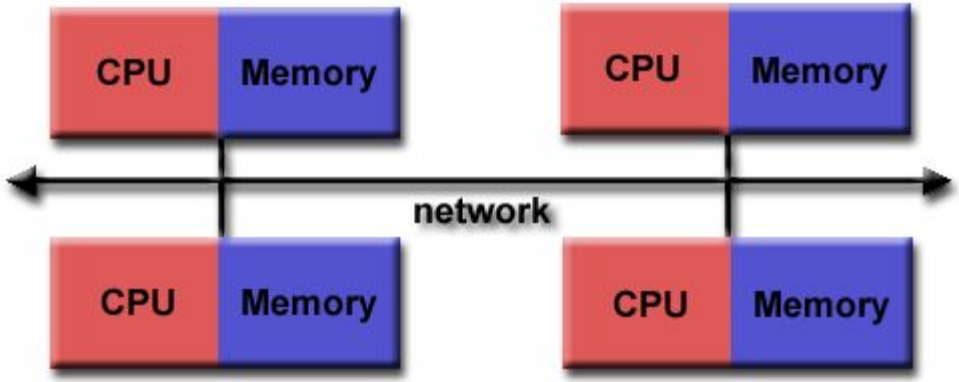
5. Anota la salida de la ejecución.

6. Repite el ejercicio variando el número de hilos, ¿qué ocurre con el tiempo de ejecución?

NLHPC
National Laboratory for High Performance Computing Chile

Arquitecturas de memoria de computación Paralela: Memoria Distribuida

- Esta arquitectura se basa en múltiples procesadores con su propia memoria física privada
- Las tareas pueden operar solo con información local
- Se prefiere esta arquitectura ya que se pueden añadir múltiples unidades de procesamiento
- Requiere una red de comunicación para conectar la memoria entre procesadores
- Las tareas intercambian datos por medio del paso y recepción de mensajes



NLHPC
National Laboratory for High Performance Computing Chile

MPI

- Es una especificación para programación de pasos de mensajes
- Proporciona una librería de funciones para C, C++ o Fortran (Existen implementaciones para Python y R)
- Características:
  - Estandarización
  - Portabilidad (Multiprocesadores, multicomputadoras)
  - Buenas prestaciones
  - Múltiples implementaciones (MPICH, **OpenMPI**, **IMPI**, LAM-MPI, MVAPICH)
- El usuario escribirá su aplicación como un proceso secuencial del que se lanzarán varias instancias que cooperan entre sí

Ejecución de Simulaciones en Slurm - Trabajos paralelos (MPI)

- En el caso de trabajos en paralelo con MPI, es necesario especificar la cantidad de tareas en total (`-n 132`) y cuantas queremos que entren por nodo (`--ntasks-per-node=44`):

```
#!/bin/bash
# -----------------SLURM Parameters----------------
#SBATCH -J paralelo_mpi
#SBATCH -p general
#SBATCH -n 132
#SBATCH --ntasks-per-node=44
#SBATCH --mem=192000
#SBATCH --mail-user=eguerra@nlhpc.cl
#SBATCH --mail-type=ALL
#SBATCH -o paralelo_mpi_%j.out
#SBATCH -e paralelo_mpi_%j.err

#-----------------Toolchain-----------------------------
ml purge
ml intel/2019b
# -----------------Módulos-----------------------------
ml  WRF/4.1-dm+sm
# -----------------Comandos-----------------------------
srun wrf.exe
```

NLHPC
National Laboratory for High Performance Computing Chile

Ejercicio 2

1. Descarga el siguiente código: pi_mpi.c en tu directorio de trabajo.

2. Teniendo en cuenta que es un código MPI, compílalo mediante el comando `mpiicc pi_mpi.c -o pi_mpi`.

3. Crea un script de ejecución para lanzarlo con sbatch, con las siguientes consideraciones: a) La partición a lanzar es slims. b) Ejecuta 20 procesos. c) Deben entrar 10 procesos por cada nodo. d) Supón que cada tarea ocupa 2300MB de memoria RAM. e) El binario a ejecutar es pi mpi.

4. Anota el script que has usado.

5. Anota la salida de la ejecución.

6. Repite el ejercicio variando el número de procesos, ¿qué ocurre con el tiempo de ejecución?

NLHPC
National Laboratory for High Performance Computing Chile

Ejecución de Simulaciones en Slurm - Trabajos paralelos Híbrido (MPI+OpenMP)

- En el caso de trabajos híbridos MPI+OpenMP, es necesario declarar el número de procesos MPI o tareas (`-n 2`) y además el número de cpus a asignar por cada una de esas tareas (`-c 44`):

```
#!/bin/bash
# ----------------SLURM Parameters----------------
#SBATCH -J hibrido
#SBATCH -p general
#SBATCH -n 2
#SBATCH --ntasks-per-node=1
#SBATCH -c 44
#SBATCH --mem=192000
#SBATCH --mail-user=eguerra@nlhpc.cl
#SBATCH --mail-type=ALL
#SBATCH -o hibrido_%j.out
#SBATCH -e hibrido_%j.err

#-----------------Toolchain---------------------------
ml purge
ml intel/2019b
# ----------------Módulos-----------------------------
ml Programa
# ----------------Comandos----------------------------
srun ./programa
```

NLHPC
National Laboratory for High Performance Computing Chile

Ejercicio 3

1. Descarga el siguiente código: hello_hybrid.c en tu directorio de trabajo.

2. Teniendo en cuenta que es un código MPI-OpenMP híbrido, compílalo mediante el comando `mpiicc hello_hybrid.c -o hello_hybrid -qopenmp`.

3. Crea un script de ejecución para lanzarlo con sbatch, con las siguientes consideraciones: a) La partición a lanzar es slims. b) Ejecuta 2 procesos. c) Cada proceso debe de tener 20 hilos de ejecución. d) Supón que cada proceso ocupa 2300MB de memoria RAM. e) El binario a ejecutar es `hello_hybrid`.

4. Anota el script que has usado.

5. Anota la salida de la ejecución.

NLHPC
National Laboratory for High Performance Computing Chile

Ejecución de simulaciones en Slurm - Parámetros

Comando SLURM	Descripción
--mem-per-cpu=<megabytes>	Memoria requerida para el trabajo por CPU (en MegaBytes). El valor predeterminado es 1024 MB.
-p <partition>, --partition=<partition>	Enviar un trabajo a una partición específica.
-n, --ntasks=<cantidad de tareas>	Número de tareas que serán asignadas para el trabajo.
-c <cpus>	Esta es la cantidad de CPU que necesita su trabajo. Tenga en cuenta que SLURM es relativamente generoso con las CPU, y el valor especificado aquí es el número "mínimo" de CPU que se asignará a su trabajo. Si hay CPU adicionales disponibles en un nodo más allá de lo solicitado, su trabajo recibirá esas CPU hasta que otros trabajos las necesiten. El valor predeterminado es 1 CPU. Intentar usar más CPU de las que se le asignaron dará como resultado que sus procesos adicionales se turnen en la misma CPU (ralentizando su trabajo).
-J <name>, --jobname=<name>	Especifica un nombre a tu trabajo.
--mail-type=BEGIN,END,FAIL,ALL **and** --mail-user=<emailAddress>	Enviar por correo electrónico cuando su trabajo comienza / termina / falla. Puede especificar varios valores para esto (separados por comas) si es necesario.
-o <STDOUT_log>, --output=<STDOUT_log>	Redirija la salida a los archivos de registro que especifique. Por defecto, ambos, STDOUT and STDERR son enviados a este archivo. Puedes especificar %j como parte del nombre de archivo de registro para indicar la ID del trabajo (como ejemplo, "#SBATCH -o ouptut_%j.o" redirigiría la salida a "output_123456.o").
-e <STDERR_log>, --error=<STDERR_log>	Redireccionar STDERR a un archivo separado. Funciona exactamente igual que "-o".
-t <days-hours:minutes:seconds>	Walltime para tu trabajo. La duración del Walltime es el tiempo que espera que su trabajo se ejecute.
-a, --array=<índices>	Envía una lista (arreglo) de trabajos identicos. Solo aplica para sbatch.

NLHPC
National Laboratory for High Performance Computing Chile

Ejecución de Simulaciones en Slurm - Trabajos secuenciales

- En este ejemplo consideramos el caso de enviar una tarea utilizando 1 core:

```
#!/bin/bash
# ----------------SLURM Parameters----------------
#SBATCH -J secuencial
#SBATCH -p slims
#SBATCH -n 1
#SBATCH --ntasks-per-node=1
#SBATCH -c 1
#SBATCH --mem=48000
#SBATCH --mail-user=eguerra@nlhpc.cl
#SBATCH --mail-type=ALL
#SBATCH -o secuencial_%j.out
#SBATCH -e secuencial_%j.err

#-----------------Toolchain---------------------------
ml purge
ml intel/2019b
# ----------------Módulos-----------------------------
ml  Python/3.8.2
# ----------------Comandos----------------------------
python3 programa.py
```

NLHPC
National Laboratory for High Performance Computing Chile

Ejecución de Simulaciones en Slurm - Trabajos secuenciales (arrays)

- Un job array en Slurm es una colección de trabajos idénticos que sólo difieren entre sí por un parámetro. La dimensión del arreglo se define con el parámetro `--array=1-n`, donde n es la cantidad de jobs. Se puede definir la cantidad de procesos simultáneos de la siguiente forma: 1-100%10. A continuación un ejemplo:

```
#!/bin/bash
# ----------------SLURM Parameters----------------
#SBATCH -J secuencial_array
#SBATCH -p slims
#SBATCH -n 1
#SBATCH --ntasks-per-node=1
#SBATCH -c 1
#SBATCH --mem=48000
#SBATCH --mail-user=eguerra@nlhpc.cl
#SBATCH --mail-type=ALL
#SBATCH --array=1-100
#SBATCH -o secuencial_array_%A_%a.out
#SBATCH -e secuencial_array_%A_%a.err

#----------------Toolchain----------------------------
ml purge
ml intel/2019b
# ----------------Módulos----------------------------
ml  Python/3.8.2
# ----------------Comandos----------------------------
python3 programa.py $SLURM_ARRAY_TASK_ID
```

NLHPC
National Laboratory for High Performance Computing Chile

Ejercicio 4

1. Descarga el siguiente código: average.py en tu directorio de trabajo.

2. Crea un script de ejecución para lanzarlo con sbatch, con las siguientes consideraciones: a) La partición a lanzar es slims. b) Ejecuta 1 tarea. c) Esta única tarea debe tener asignados 2 core. d) Supón que cada tarea ocupa 2300MB de memoria RAM. e) Debes utilizar la versión de Python más reciente. f) La dimensión del arreglo es de 100 y deben entrar de a 10 simultáneos

4. Anota el script que has usado.

5. Anota la salida de la ejecución.

NLHPC
National Laboratory for High Performance Computing Chile

Ejecución de Simulaciones en Slurm - Trabajos paralelos (GPU)

- Los trabajos que utilizarán las GPUs deben indicar la cantidad a utilizar (`--gres=gpu:1`) Cada nodo tiene 2 GPUs:

```
#!/bin/bash
# ----------------SLURM Parameters----------------
#SBATCH -J GPU
#SBATCH -p gpus
#SBATCH -n 44
#SBATCH --gres=gpu:1
#SBATCH --ntasks-per-node=44
#SBATCH -c 1
#SBATCH --mem=192000
#SBATCH --mail-user=eguerra@nlhpc.cl
#SBATCH --mail-type=ALL
#SBATCH -o GPU_%j.out
#SBATCH -e GPU_%j.err

#----------------Toolchain---------------------------
ml purge
ml fosscuda/2019b
# ----------------Módulos-----------------------------
ml  NAMD/2.13-mpi
# ----------------Comandos----------------------------
srun namd2
```

NLHPC
National Laboratory for High Performance Computing Chile

Ejercicio 5

1. Descarga los siguientes archivos en tu directorio de trabajo: mulBy2.cu, Makefile

2. Teniendo en cuenta que es un código en cuda y que tenemos un Makefile, compílalo mediante el comando **make** en tu directorio de trabajo

3. Crea un script de ejecución para lanzarlo con sbatch, con las siguientes consideraciones: a) La partición a lanzar es gpus. b) Ejecuta 1 proceso. c) Cada proceso debe de tener 1 core. d) Supón que cada proceso ocupa 2300MB de memoria RAM. e) El binario a ejecutar es `mulBy2`.

4. Anota el script que has usado.

5. Anota la salida de la ejecución.

NLHPC
National Laboratory for High Performance Computing Chile

Dependencias de trabajos

Las dependencias de trabajos se utilizan para aplazar el inicio de un trabajo hasta que se satisfagan las dependencias especificadas. Se especifican con la opción `--dependency` en el siguiente formato:

```
sbatch --dependency=<type:job_id[:job_id][,type:job_id[:job_id]]> ...
```

Los tipos de dependencias son las siguientes:

`after:jobid[:jobid...]`
el trabajo puede empezar después de que los trabajos especificados comiencen
`afterany:jobid[:jobid...]`
el trabajo puede empezar después de que los trabajos especificados terminen
`afternotok:jobid[:jobid...]`
el trabajo puede empezar después que los trabajos especificados terminan fallidamente
`afterok:jobid[:jobid...]`
el trabajo puede empezar después que los trabajos especificados terminan exitósamente

NLHPC
National Laboratory for High Performance Computing Chile

Ejemplo de dependencias de trabajos - Ejemplo

La manera más simple de usar una dependencia del tipo afterok:

```
[prueba@leftraru1 ~]$ sbatch job1.sh
Submitted batch job 21363626
[prueba@leftraru1 ~]$ sbatch --dependency=afterok:21363626 job2.sh
```

Ahora cuando job1.sh termine correctamente, el job2.sh entrará en ejecución. Si job1.sh termina fallidamente, job2.sh no entrará en ejecución nunca pero sí quedará en cola (debe cancelarse manualmente el trabajo).

NLHPC
National Laboratory for High Performance Computing Chile

Programación de tareas usando crontab

Crontab es un planificador de tareas donde se guarda un listado de comandos a ejecutar en un tiempo determinado por el usuario.

Para acceder a crontab utilice el siguiente comando:

```
[prueba@leftraru1 ~]$ crontab -e
```

Esto le permitirá editar su archivo crontab, la estructura base es la siguiente:

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  *     command to be executed
```

NLHPC
National Laboratory for High Performance Computing Chile

Programación de tareas usando crontab - Ejemplo

Primero debemos especificar el tiempo, en este caso la tarea será ejecutada todos los días cada 20 minutos, luego debemos obtener el archivo `/etc/profile` que contiene las definiciones comunes para los usuarios, posterior a ello, declaramos el comando sbatch para iniciar nuestra tarea con el gestor de recursos slurm, para finalizar, indicamos el script enviado por el usuario y guardamos los cambios en el archivo:

```
*/20 * * * * . /etc/profile ; sbatch /home/prueba/ejemplo/script.sh
```

Para revisar el listado de tareas existentes en nuestro crontab, ejecutamos el comando:

```
[prueba@leftraru1 ~]$ crontab -l
*/20 * * * * . /etc/profile ; sbatch /home/prueba/ejemplo/script.sh
```

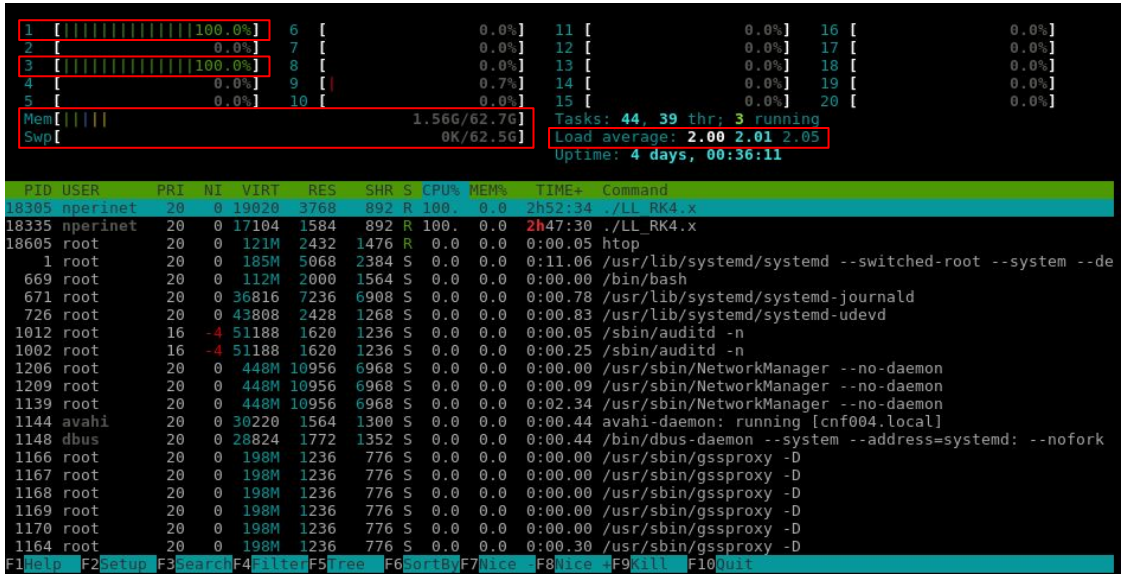
Para borrar el contenido de nuestro crontab:

```
[prueba@leftraru1 ~]$ crontab -r
```

NLHPC
National Laboratory for High Performance Computing Chile

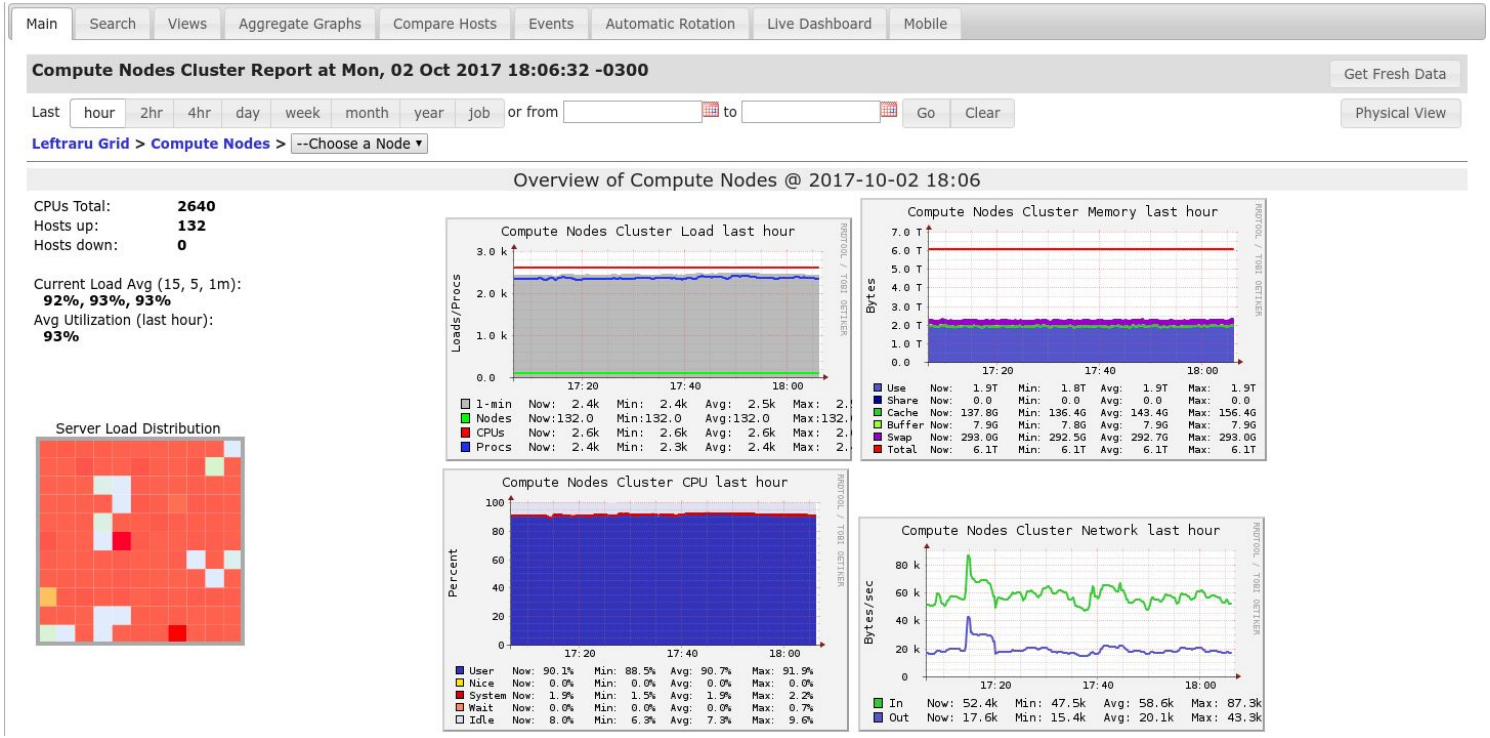
Monitoreo de Simulaciones - htop

- Puede ingresar a través de ssh a un nodo en donde tenga una tarea en ejecución y ejecutar **`htop`:**



Monitoreo de simulaciones - Ganglia

- Para monitorear nuestras tareas en ganglia, deben ir a http://monitor.nlhpc.cl/ganglia/



Monitoreo de Simulaciones - Ganglia

Ejercicio 6

1. ¿Cuánto espacio de almacenamiento tienes ocupado en tu cuenta de usuario? ¿Qué comando debes usar para obtener dicha información?

2. Si quieres lanzar un programa en MPI en la infraestructura del NLHPC y tienes 120 cores disponibles, ¿cuánto deberías de reservar/usar?

NLHPC
National Laboratory
for High Performance
Computing
Chile

Instalación y Compilación de aplicaciones - Compiladores

- La compilación es el proceso por el cual se traducen las instrucciones escritas en un determinado lenguaje de programación a lenguaje máquina
- Un compilador es un programa de computadora que “traduce” las instrucciones o código fuente desde un lenguaje a otro (usualmente código máquina)
- En ambientes HPC, se busca sacar el máximo rendimiento al hardware con las aplicaciones, por lo que el 99% de los programas se encuentra compilado
- En nuestro caso, disponemos de los siguientes set de herramientas de compilación:
  - **Intel Toolchain**
  - **GNU Toolchain**
  - Estos 2 set de herramientas de compilación permiten compilar programas escritos en C, C++, Fortran
- Recomendamos siempre hacer uso de las herramientas de Intel, ya que se encuentran optimizadas para utilizar las características de sus microprocesadores en toda su extensión

NLHPC
National Laboratory for High Performance Computing Chile

Instalación y Compilación de aplicaciones - Flags

- Los flags de optimización se usan para:
  - Disminuir la cantidad de mensajes de depuración
  - Aumentar los niveles de aviso de errores
  - Optimizar el código producido
- Algunos de los flags de optimización utilizados en nuestras aplicaciones:
  - **`-O3`**: Habilitar la optimización del compilador, donde 3 representa el nivel (máximo)
  - **`-ipo`**: La optimización interprocedural le permite al compilador analizar el código para determinar donde este puede beneficiarse de optimizaciones específicas
  - **`-no-prec-div`**: Mejora la precisión de las divisiones de punto flotante
  - **`-static-intel`**: Hace que las bibliotecas proporcionadas por intel se vinculen estáticamente
  - **`-fPIC`**: Determina si el compilador genera código independiente de la posición
  - **`-qopenmp`**: Permite que el paralelizador genere código multiproceso basado en directivas OpenMP

NLHPC
National Laboratory for High Performance Computing Chile

Instalación y Compilación de aplicaciones - Ejemplos de compilación

- La compilación más básica de un programa en C sería la siguiente:
  - **`icc hello_world.c -o hello_world`**
  - Esta compilación generará un ejecutable llamado “hello_world” el cual puede ser ejecutado de la siguiente forma: **`./hello_world`**
- Si queremos compilar un programa en C que soporte MPI, la compilación sería la siguiente:
  - **`mpiicc hello_world.c -o hello_world -O3 -axCORE-AVX512,AVX,SSE4.2 -ipo -no-prec-div -static-intel -fPIC -qopenmp`**
- La opción **`-axCORE-AVX512,AVX,SSE4.2`** la utilizamos para que nuestro ejecutable pueda ejecutarse en los distintos tipos de nodos que tenemos actualmente
- Compiladores disponibles: **gcc/icc/mpiicc** (C) **g++/icpc/mpiicpc** (C++), **gfortran/ifort/mpiifort** (Fortran)

NLHPC
National Laboratory for High Performance Computing Chile

Instalación de módulos en Python y R

- En el caso de que queramos instalar módulos de Python localmente en nuestro directorio utilizaremos el siguiente comando:
  - Cargar el módulo de Python: **`ml Python/3.8.3`**
  - **`pip install programa --no-binary :all: --user`**
  - El parámetro **`--no-binary :all:`** intentará compilar el módulo a instalar
  - El parámetro **`--user`** instalará en nuestro directorio el módulo
- Para instalar paquetes de R localmente en nuestro directorio:
  - Cargamos el módulo de R: **`ml R/4.0.0`**
  - **`R CMD INSTALL -l /home/prueba/R/library programa`**
  - Para decirle a R que utilice los paquetes instalados por mi: **`library("paquete", lib.loc="/home/prueba/R/library")`**

NLHPC
National Laboratory for High Performance Computing
Chile

Problemas Frecuentes - Exceso de memoria

- Debemos tener en cuenta que la memoria RAM que SLURM reserva por defecto es de 1000MB (1GB). Un típico error de cancelación de tareas es por utilizar más de la memoria reservada:

```
/tmp/slurmd/job136839939/slurm_script: line 15: 23547 Killed                ./programa.sh
slurmstepd: error: Detected 1 oom-kill event(s) in step 136839939.batch cgroup. Some of your processes
may have been killed by the cgroup out-of-memory handler.
```

  - Si su tarea ocupa más de la memoria por defecto (1GB), se puede utilizar el siguiente parámetro en SLURM:
    - **`#SBATCH --mem=2300`**
  - Esto hará que SLURM reserve 2300MB (2.3GB) por la totalidad del trabajo
- Para más detalles de nuestra infraestructura visitar: http://wiki.nlhpc.cl/index.php/Recursos_Computacionales_NLHPC

NLHPC
National Laboratory for High Performance Computing Chile

Problemas Frecuentes - Subutilización de CPU y RAM

- Nuestro objetivo en el cluster es velar por el uso óptimo de los recursos, por lo que estamos continuamente monitoreando el uso de los nodos. En el caso de la subutilización de CPU, la condición para que una tarea sea cancelada es que la mitad del total de cores reservados muestre un porcentaje de uso menor o igual a un 30% en un lapso de 4 horas. Para el caso de la memoria RAM, lo que se busca es garantizar el uso de al menos un 70% del recurso reservado.
- El procedimiento en ambos casos es el siguiente:
  - Se envía una notificación a las 2 horas de ejecución de la tarea que subutilice recursos
  - Si mantiene el comportamiento en las próximas 2 horas, se envía una notificación informando la cancelación de la tarea
  - En el correo de cancelación se adjuntan los gráficos del uso de CPU y Memoria por cada nodo con el fin de que puedan realizar los cambios respectivos en las próximas tareas

NLHPC
National Laboratory for High Performance Computing Chile

Problemas Frecuentes - Ejemplo subutilización de CPU

- Tenemos la tarea de un usuario que reserva 4 cpu y 4800MB de memoria. La notificación recibida es la siguiente:

Estimada/o Pablo,

Durante los procesos rutinarios de monitoreo hemos observado el comportamiento de la tarea **18197288** (**test_general**), iniciada por el usuario **pflores**.

Actualmente presenta un comportamiento que podemos clasificar como: subutilización de recursos, dado que **4** de sus cores presentan menos del **2%** de utilización en las últimas 2 horas.

Los datos recopilados son:

JobID	Start	End	Elapsed	Partition	NodeList	NCPUS	ReqMem
18197288	2020-05-24T15:03:24	Unknown	02:44:08	slims	cn042	4	4800Mn

Consideramos esta información relevante, ya que su tarea se comporta en forma poco eficiente, al no utilizar óptimamente los recursos reservados dentro del clúster.

- Si este comportamiento se repite en las siguientes 2 horas, la tarea será cancelada
- Al revisar el archivo report.log de la notificación, tenemos lo siguiente:

```
* cpu= 9 avg=0.75 std=0.43 / all avg=0.79 std=0.41
* cpu= 8 avg=1.08 std=0.40 / all avg=1.07 std=0.36
* cpu= 7 avg=1.08 std=0.40 / all avg=1.07 std=0.36
* cpu= 6 avg=1.00 std=0.00 / all avg=1.00 std=0.00
```

- Debido a que el uso de CPU es tan bajo, la recomendación es:
  - Reservar siempre lo realmente utilizado, en este caso 1 cpu

NLHPC
National Laboratory for High Performance Computing
Chile

Problemas Frecuentes - Ejemplo subutilización de Memoria

- Tenemos la tarea de un usuario que reserva 12 cpu y 2.5GB por cada cpu, osea un total de 30GB:

Estimada/o Eugenio,

Durante los procesos rutinarios de monitoreo hemos observado el comportamiento de la tarea **18202593** (**ORCA-a-bpa-e**), iniciada por el usuario **eguerra**.

Actualmente presenta un comportamiento que podemos clasificar como: subutilización de recursos, dado que desde el inicio de la ejecución presenta menos del **70%** de utilización de memoria.

Los datos recopilados son:

JobID	Start	End	Elapsed	Partition	NodeList	NCPUS	ReqMem
18202593	2020-05-25T14:50:13	Unknown	02:10:47	slims	cn075	12	2.50Gc

- Si este comportamiento se repite en las siguientes 2 horas, la tarea será cancelada
- Al revisar el archivo report.log de la notificación, tenemos lo siguiente:

```
* mem max:16861 resv:30720
```

- Debido a que el uso de memoria sólo llega a 16.8GB:
  - Se debe reservar la memoria que realmente se utiliza:
  - **#SBATCH --mem=18000**

NLHPC
National Laboratory for High Performance Computing Chile

NLHPC
National Laboratory for High Performance Computing Chile

¡Gracias por participar!

www.nlhpc.cl
2022