

## FUNDAMENTOS DE CHPC

Se presentará una visión general de un sistema computacional de alto rendimiento. El CHPC es un recurso importante pero limitado de la Universidad de Utah y sus afiliados, debe usarse de manera apropiada.

### Acceso a los sistemas HPC

Al obtener una cuenta HPC, se otorga acceso a todas las Plataformas de Computación de Alto Rendimiento de CHPC

- Disponible a través del programa ssh (shell seguro).
- Se permite solo el protocolo 2 para acceder al sistema (no se permiten claves ssh por motivos de seguridad).

Al provisionarse la cuenta al afiliado:

- Se abastecerá de los archivos de inicio de sesión necesarios para un entorno tcsh o bash. Dependiendo del shell escogido al crear la cuenta, se usará lo necesario para configurar el entorno para poder usar los clústeres.
- Los archivos de inicio de sesión proporcionados incluyen .tcshrc, .bashrc, .custom.sh, .custom.csh. Estos archivos usarán el sistema LMod para configurar el entorno.
- No modificar los archivos .tcshrc y .bash, usar los archivos .custom.sh y .custom.csh, junto con la creación de un archivo .aliases para configurar el entorno interactivo.

Los clústeres HPC actualmente validados son:

- Notchpeak Cluster:
  - ssh notchpeak.chpc.utah.edu
- Kingspeak Cluster:
  - ssh kingspeak.chpc.utah.edu
- Lonepeak Cluster:
  - ssh lonepeak.chpc.utah.edu

Y el cluster HPC para ambientes protegidos (datos protegidos):

- Redwood Cluster:
  - ssh redwood.chpc.utah.edu

Cuando se accede por ssh a alguno de los clusters anteriores, aterrizas en el respectivo **nodo de inicio de sesión** del cluster escogido. Se recomienda usar FastX para ingresar a los clústeres. El trabajo computacional sustancial se realiza en los nodos de cómputo que se acceden a través de un sistema por lotes. Todos los clústeres usan Slurm para administrar el sistema por lotes.

Los clusters anteriores se operan en un estilo de condominio, con nodos del CHPC y de grupos de investigación individuales.

- En notchpeak y redwood hay un sistema de asignación de tiempos en los nodos generales propiedad del CHPC.
- En kingspeak y lonepeak, la ejecución es sin asignación.
- Los nodos de propietarios están disponibles para todos los usuarios del CHPC para ejecutar como invitados, cuando no están siendo utilizados por el grupo propietario.

Se tiene un clúster restringido ash propiedad de un único grupo al que se le permite el acceso de invitados a través de ash-guest.chpc.utah.edu.

### Información del uso de disco

CHPC proporciona espacio de directorio personal, montado en NFS en todos los sistemas HPC. Este sistema de archivos no cuenta con respaldo.

Los usuarios son responsables de trasladar datos importantes a un lugar más permanente, como el servidor de archivos de su departamento de origen.

Los grupos individuales pueden adquirir espacio adicional en el directorio personal que incluye respaldo en cinta, así como espacio de grupo que puede incluir espacio archivado trimestral en cinta.

El CHPC proporciona varios niveles de espacio temporal en las diversas plataformas de HPC.

## **Documentación**

La descripción de los recursos CHPC es un buen punto de partida.

- La información común a todos los clústeres de HPC se proporciona en la [GUÍA DE USUARIO DEL CLÚSTER DE HPC](#).
- La información específica de un clúster determinado se encuentra en guías de usuario web-based para los [clústeres individuales](#)

También se tiene [documentación en relación a herramientas de programación, utilidades de acceso y de transferencia de archivos, sistema por lotes \(batch system\), y aplicaciones seleccionadas instaladas en los clústeres CHPC HPC](#).

Todo lo anterior debería ser suficiente para iniciarme con un recurso CHPC determinado. [Toda la documentación](#) estará disponible.

## **ÍNDICE DE ENTRENAMIENTO**

- Recursos computacionales
  - Conceptos básicos e introducción
  - Ambiente y empleo
  - Slurm (Programación de trabajos)
- Almacenamiento y archivos
- Guía para Servicios y Software
  - Presentaciones
  - Paquetes and software installation
  - Rclone

## **RECURSOS COMPUTACIONALES**

### **Conceptos básicos e introducción**

Se introducirá brevemente a CHPC, se muestran los servicios y herramientas ofrecidas por CHPC: visión general de la formación y documentación, herramientas para conectarse a recursos, una revisión de las asignaciones, e introducciones a otro software.

La misión de CHPC es resguardar las diversas necesidades computacionales de investigación. Para esto, Utah University ofrece muchos recursos, incluyendo servidores computacionales (ejecutables en Linux y Windows), en un ambiente general y protegido; máquinas virtuales, almacenamiento y movimiento de datos; y un networking de alta velocidad avanzado.

Podemos hacer uso de estos recursos si tenemos grandes tareas, donde necesitemos usar múltiples servidores, donde necesitemos una sola computadora poderosa o si tenemos un dataset extenso para almacenar, o aplicaciones específicas para instalar y correr. En general, CHPC puede ayudar si:

- Necesitas procesar en paralelo
- Necesitas ejecutar varias cosas simultáneamente.
- Necesitas una sola gran computadora.
- Tienes mucha data para almacenar o procesar.
- Debes utilizar un ambiente protegido (Protected Environment), si cuentas con IRB-governed Protected Health Information (PHI)
- Si tienes necesidades informáticas de investigación no cubiertas por recursos locales (del departamento, por ejemplo)

### **Conectando a recursos HPC**

Los servidores informáticos CHPC y otros recursos están instalados en Downton Data Center (DDC), el cual opera 24/7/365, y también alberga otros recursos computacionales usados por Utah University.

Los recursos en el DDC son variados. Hay demasiados “clusters” los cuales son una colección de servidores con similar hardware. Los servidores por sí mismos son grandes computadoras, típicamente presentando múltiples CPUs con varios núcleos y una cierta cantidad de memoria. Algunos quizá tienen GPUs o un hardware específico para cierto trabajo computacional.

También hay nodos con otros propósitos además del trabajo de investigación, como gestionadores de recursos para asignar tareas a servidores individuales dentro de los clusters.

La red (Network) cambia la transferencia de datos entre servidores y externamente (con internet).

Los sistemas de almacenamiento albergan datos de investigación y respaldos (back up) en directorios de inicio y de grupo.

Los recursos individuales trabajan juntos para garantizar el trabajo computacional de los investigadores.

El staff de CHPC también provee una amplia variedad de servicios, desde diagnósticos de hardware y mantenimiento hasta apoyo de networking a usuarios y gestión de software

### **Interactuando con los recursos CHPC**

La mayoría de los usuarios usan los clusters por trabajo de investigación, estos se pueden alcanzar con herramientas de software como SSH y FastX. Una vez que tienes una cuenta de usuario, tú puedes acceder a los nodos de inicio de sesión (login nodes) directamente con tales herramientas (SSH y FastX). Luego, preparar grandes tareas llamadas “jobs” para enviar al planificador, Slurm.

Cuando los recursos que solicitas se vuelven disponibles, tu “job” ha comenzado.

Trabajos gráficos se pueden hacer a través de jobs interactivos con FastX o reenvío X. Es también posible usar los nodos frisco para tareas breves requiriendo software gráfico.

### **Asignaciones (allocations)**

Los allocation de los clústeres HPC se actualizan trimestralmente. Hay tres tipos de allocation en CHPC:

- Quick allocations
  - Para nuevos PI quienes no han tenido un allocation antes, y proporciona hasta 20 000 horas de core (núcleo).
- Small allocations
  - Además de proporcionar 20 000 horas de core (núcleo), está destinado para PIs quienes han recibido un allocation antes.

Quick y Small allocations son revisados internamente por el staff CHPC y no por el comité de asignación.

- Regular allocations
  - Para cualquier cosa mayor a 20 000 horas de core, son revisados por el comité de asignación.

### Opciones de almacenamiento

La data generada o requerida cuando trabajas en recursos CHPC puede ser almacenada en directorios de inicio (home) o grupos de filesystems (sistemas de archivos).

El espacio más allá del directorio de inicio estándar deben ser adquiridos por los grupos de investigación. Algunos espacios son además respaldados.

El almacenamiento de archivos está disponible tanto en el entorno general como en el protegido.

El scratch storage (almacenamiento de rasguños) es una alternativa si no quieres guardar tu data por un periodo largo.

### Slurm

- El sistema usado para enviar jobs a recursos computacionales en clusters Linux es el programador Slurm, con el cual se puede interactuar desde los nodos de inicio.
- CHPC usa Slurm para programar jobs en clústeres Linux.
- Los jobs de Slurm pueden ser corridos interactivamente o con job scripts.
- Es un job scheduler usado en CHPC para proporcionar acceso a nodos computacionales en clúster CHPC HPC.
  - Así es como se ingresa al batch system para poner en cola los jobs.
  - Así es como ingresas a nodos computacionales directamente mediante sesiones interactivas.
  - Para interconectar los nodos computacionales, debes pasar por Slurm.

### Cuentas y particiones

- Una cuenta es un grupo de acceso, que te brinda acceso a diferentes recursos en el cluster.
- Una partición es una unidad lógica que divide el cluster en diferentes unidades utilizables, basados en cualidades o rasgos/traits de diferentes nodos.
  - La configuración de particiones varía entre diferentes ambientes de computación.
  - En CHPC tenemos cuatro particiones que usamos en nuestro main: allocated clusters (kingspeak & ember).
    - kingspeak & ember (allocated clusters)
    - Cuatro particiones:
      - kingspeak, ember (para uso allocated)
      - kingspeak-freecycle, ember-freecycle: al correr sin alocación.
      - baggins-kp, baggins-em (owner nodes): para grupos que también tienen nodos privados, la sintaxis es [group name]-(kp/em), dependiendo si estás en el cluster kingspeak o ember
      - kingspeak-guest, ember-guest (owner-guest): permite correr los owner nodes como guest, donde tus jobs son preemptible.
    - Cuentas:
      - allocated o freecycle: PI name
      - owner nodes: el mismo nombre que el de la partición

- owner-guest (para usar owner-nodes como) como guest (invitado).
- lonepeak and tangent (freecycle clusters):
  - Solo tienen una partición cada una (lonepeak, tangent).
  - Usa tu nombre de cuenta estándar.
  - Es non-preemptible, pero la disponibilidad inmediata no está garantizada: es una especie de orden de llegada (dentro de lo razonable).
  - tangent es un servidor experimental basado en la nube. Lanzar tus jobs puede ser un poco raro.

En slurm, hay comandos básicos que necesitas conocer en orden de cargar jobs, cancelar jobs, y ver el estado de jobs corriendo.

squeue: muestra una lista de todos los jobs que están ejecutándose en el cluster actual, en ese instante. Esto no es tan útil, pues solamente estás interesado en tus jobs que están corriendo.

```
-----
Last login: Thu May 23 13:33:34 2024 from 187.191.38.52
[u6059911@kingspeak2:~]$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
13253531	carbon-kp	run_ecli	u6047901	R	9-03:27:44	1	kp371
13258185	gertz-kp	ondemand	u6023898	R	1:07	1	kp154
13258186	kingspeak	ondemand	u1142170	PD	0:00	1	(Resources)
13257819	kingspeak	alpha_cp	u6032169	PD	0:00	1	(Priority)
13257820	kingspeak	alpha_cp	u6032169	PD	0:00	1	(Priority)
13257821	kingspeak	alpha_cp	u6032169	PD	0:00	1	(Priority)
13258179	kingspeak	ESfilter	u6033116	PD	0:00	1	(Priority)
13258184	kingspeak	3.2-IQTR	u6039728	PD	0:00	1	(Priority)
13258183	kingspeak	3.2-IQTR	u6039728	PD	0:00	1	(Priority)
13258084	kingspeak	0_run_de	u1130312	PD	0:00	1	(Priority)
13258085	kingspeak	0_run_de	u1130312	PD	0:00	1	(Priority)
13258087	kingspeak	0_run_de	u1130312	PD	0:00	1	(Priority)

squeue -u [username]: muestra únicamente tus jobs. Al no tener jobs corriendo, no tendrás mucha información para ver. La información que esto te da es el JOB ID, la partición que estás usando, el nombre del job, tu usuario, el estado del job, cuanto tiempo ha estado funcionando este job, los nodos que has estado usando, el número de nodos, y la lista de nodos. Esto es útil porque da información de si las cosas están funcionando.

```
[u6059911@kingspeak2:~]$ squeue -u u6059911
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
13258186	kingspeak	ondemand	u1142170	PD	0:00	1	(Resources)

squeue --account = owner-guest: muestra información de las cuentas de los owner-guest del grupo.

```
[u6059911@kingspeak2:~]$ squeue --account=owner-guest
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
13258132	kingspeak	ondemand	u1360742	R	1-00:19:28	1	kp112
13258131	kingspeak	ondemand	u1360742	R	1-00:28:22	1	kp387
13258130	kingspeak	ondemand	u1360742	R	1-00:40:57	1	kp386
13257809	kingspeak	3CX46H2C	u6056291	R	2-01:03:08	8	kp[274-281]
13257766	kingspeak	1CX46H1C	u6056291	R	2-02:15:28	8	kp[258-265]
13257596	kingspeak	8CX50H2S	u6056291	R	2-03:18:00	8	kp[372-379]
13257482	kingspeak	5CX50H1C	u6056291	R	2-03:38:42	8	kp[119-126]
13257454	kingspeak	2CX46H1S	u6056291	R	2-06:31:23	8	kp[234-237,246-249]

squeue --partition = [partition\_name]: Da la misma información que lo anterior, pero es útil si se desea estudiar una dada partición.

```
[u6059911@kingspeak2:~]$ squeue --partition=kingspeak-guest
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(REASON)
13258132	kingspeak	ondemand	u1360742	R	1-00:20:02	1	kp112
13258131	kingspeak	ondemand	u1360742	R	1-00:28:56	1	kp387
13258130	kingspeak	ondemand	u1360742	R	1-00:41:31	1	kp386
13257809	kingspeak	3CX46H2C	u6056291	R	2-01:03:42	8	kp[274-281]
13257766	kingspeak	1CX46H1C	u6056291	R	2-02:16:02	8	kp[258-265]
13257596	kingspeak	8CX50H2S	u6056291	R	2-03:18:34	8	kp[372-379]
13257482	kingspeak	5CX50H1C	u6056291	R	2-03:39:16	8	kp[119-126]
13257454	kingspeak	2CX46H1S	u6056291	R	2-06:31:57	8	kp[234-237,246-249]

sinfo: muestra el estado de diferentes particiones

```
[u6059911@kingspeak2:~]$ sinfo
```

PARTITION	AVAIL	TIMELIMIT	NODES	STATE	ODELIST
kingspeak*	up 3-00:00:00		1	drain*	kp028
kingspeak*	up 3-00:00:00		1	mix	kp166
kingspeak*	up 3-00:00:00		51	alloc	kp[001-010,012-023,025-027,029-032,110-111,158-160,162-165,167,196-199,223,384-385]
kingspeak-shared	up 3-00:00:00		1	drain*	kp028
kingspeak-shared	up 3-00:00:00		1	mix	kp166

Estado de partición idle: Es de los más interesantes. Cuando observas una partición en el estado idle, puedes ver cuantos nodos están disponibles a un tiempo dado (en el caso de kingspeak, nodos para asignación o allocation).

Hay algunos aliases que puedes encontrar en la documentación de CHPC, bajo la página de software Slurm.

srun: carga un job interactivo en sistemas CHPC.

sbatch: se utiliza para iniciar batch scripts alistandose en cola para trabajar más adelante. Observe.

```
[u0123458@kingspeak1:qe]$ squeue -u u0123458
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(REASON)
455450	kingspeak	run-qe.s	u0123458	CG	1:16	1	kp141

```
[u0123458@kingspeak1:qe]$ squeue -u u0123458
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(REASON)
455453	kingspeak	run-qe.s	u0123458	PD	0:00	1	(Priority)

```
[u0123458@kingspeak1:qe]$ sbatch run-qe.slurm
```

Submitted batch job 455453

```
[u0123458@kingspeak1:qe]$ squeue -u u0123458
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(REASON)
455453	kingspeak	run-qe.s	u0123458	PD	0:00	1	(Priority)

```
[u0123458@kingspeak1:qe]$ scancel 455453
```

```
[u0123458@kingspeak1:qe]$ squeue -u u0123458
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(REASON)
455454	kingspeak	run-qe.s	u0123458	R	0:02	1	kp141

```
[u0123458@kingspeak1:qe]$ sbatch run-qe.slurm
```

Submitted batch job 455454

```
[u0123458@kingspeak1:qe]$ squeue -u u0123458
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	ODELIST(REASON)
455454	kingspeak	run-qe.s	u0123458	R	0:02	1	kp141

```
[u0123458@kingspeak1:qe]$ scontrol show job 455454
```

En la imagen, en el directorio qe, se tiene el script (.slurm) run-qe.slurm, y sbatch [script\_name] someterá un batch job, si el script anterior está configurado correctamente (si no está configurado correctamente, Slurm es muy bueno avisando si hiciste un buen trabajo o no).

squeue -u [user\_name], puedes ver los job en la cola. Mostrará también los estados del job (PD o pending, R o running, CG o canceling job). TIME muestra el tiempo que el job ha estado funcionando. Para trabajos que corren hasta finalizar, es usual que expiren del squeue. Si tienes problemas con tu job script, quizá veas terminar más rápido a tu job, lo cual puede ser malo para el usuario.

Se pueden cancelar jobs usando `scontrol [JOB_ID]`, y usar seguidamente `squeue -u [user_name]` para verificar que el job ha sido cancelado.

El comando `scontrol show job [JOB_ID]` sirve para ver información adicional y detallada del job en particular que se escoja.

```
[u0123458@kingspeak1:qe]$ scontrol show job 455454
JobId=455454 JobName=run-qe.slurm
  UserId=u0123458(67335) GroupId=chpc(1722)
  Priority=16 Nice=0 Account=owner-guest QOS=kingspeak-guest
  JobState=RUNNING Reason=None Dependency=(null)
  Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
  RunTime=00:00:16 TimeLimit=01:00:00 TimeMin=N/A
  SubmitTime=2015-07-16T15:52:54 EligibleTime=2015-07-16T15:52:54
  StartTime=2015-07-16T15:52:58 EndTime=2015-07-16T16:52:58
  PreemptTime=None SuspendTime=None SecsPreSuspend=0
  Partition=kingspeak-guest AllocNode:Sid=kingspeak1:3846
  ReqNodeList=(null) ExcNodeList=(null)
  NodeList=kpl41
  BatchHost=kpl41
  NumNodes=1 NumCPUs=20 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
  Socks/Node=* NtasksPerN:B:S:C=0:0:*:1 CoreSpec=*
  MinCPUsNode=1 MinMemoryNode=0 MinTmpDiskNode=0
  Features=(null) Gres=(null) Reservation=(null)
  Shared=0 Contiguous=0 Licenses=(null) Network=(null)
  Command=/uufs/chpc.utah.edu/common/home/u0123458/qe/run-qe.slurm
  WorkDir=/uufs/chpc.utah.edu/common/home/u0123458/qe
  StdErr=/uufs/chpc.utah.edu/common/home/u0123458/qe/qe-455454
  StdIn=/dev/null
  StdOut=/uufs/chpc.utah.edu/common/home/u0123458/qe/qe-455454
```

## **Slurm Batch Scripting**

Se hablará sobre Slurm batch jobs y batch scripting.

### **¿Cuándo deberías usar el batch system?**

Debes usar el batch system cuando tienes un trabajo grande que está ejecutándose en paralelo, que es de cálculo de trabajo pesado, y cualquier cosa que sea computación de alto rendimiento.

Para usar el batch system, realmente tienes que tener un conocimiento de scripting. Vamos a repasar los fundamentos de lo que necesitas poner en un batch script para ejecutar en un batch system.

### **¿Qué incluye un batch script?**

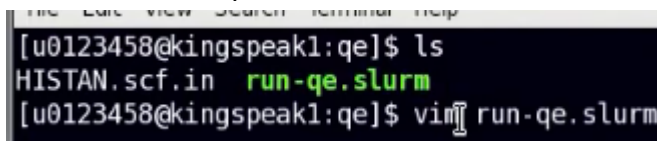
- SBATCH directives
  - Lo primero que pones en un batch script bajo un Slurm system es un conjunto de directivas SBATCH.
  - SBATCH directives dice al Scheduler que usar en términos de número de nodos, cuantos núcleos por nodo, cuantas tareas a través de todos estos nodos, la cuenta y la partición que quieres usar, y cuanto tiempo vas a ejecutar.
  - Toda esta información es útil para decir al Scheduler cómo se verá el job.
- Environment setup
  - Una vez tienes los SBATCH directives configurados, necesitas configurar tu entorno.
  - En CHPC, lo que esto implica es cargar algunos módulos, quizá configurando algunos PATH variables adicionales para poder correr un programa en



específico escrito por el usuario mismo. Basicamente, configurar tu entorno para que puedas correr tu trabajo.

- Scratch setup
  - Muchos jobs necesitan un scratch directory desde el que ejecutarse. Es una buena práctica configurar tu scratch directory en tu batch script antes de ejecutarlo.
- Copy data to scratch (for heavy IO loads)
  - Si anticipas tener varios escritos, deberías escribir tus escritos en scratch.
  - Si estás haciendo algo más pequeño y ligero, no va a impactar a los file servers. Se puede ejecutar desde el home directory, pero es generalmente no recomendable.
- Run your job (MPI if need be)
  - Si estás corriendo un batch script, quieres correr tu trabajo, y quieres ejecutarlo en paralelo con MPI si es necesario.
  - Obviamente, quieres asegurarte de que ejecutes tu trabajo en tu batch script, de otra manera no estarías haciendo nada.
- Copy data from scratch
  - Quieres asegurarte que si tu tienes data que estás escribiendo en tu scratch, lo copies en una ubicación segura. Ya sea en tu home directory o en tu group space.
  - De cualquier manera, si estás escribiendo en scratch, asegurate de copiar desde scratch para que no pierdas data en scratch.
  - El scratch file server en CHPC son depurados luego de 30 días.
- Remove temporary files and scratch data
  - Si tu programa escribe algunos archivos temporales, o si tienes leftover scratch data (datos sobrantes), deberías removerlos para que puedas ser un buen usuario.

Mostramos el script así:



```
[u0123458@kingspeak1:qe]$ ls
HISTAN.scf.in  run-qe.slurm
[u0123458@kingspeak1:qe]$ vim run-qe.slurm
```



Lo cual muestra:

```
#!/bin/bash
#SBATCH --time=1:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=16
#SBATCH --account=owner-guest
#SBATCH --partition=kingspeak-guest
#SBATCH -o qe-%j

NAME=HISTAN.scf

DATADIR=/uufs/chpc.utah.edu/common/home/u0123458/qe

SCRATCH=/scratch/kingspeak/serial/u0123458/qe/

module purge
module load intel impi qe

mkdir -p $SCRATCH

#copy data to scratch here if you need to

mpirun -np $SLURM_NTASKS pw.x -inp $DATADIR/$NAME.in > $DATADIR/$NAME.$SLURM_JOB_ID.out
#srun pw.x -inp $DATADIR/$NAME.in > $DATADIR/$NAME.$SLURM_JOB_ID.out - depends on MPI

#copy data from scratch here if you need to
```

- ¿Qué debo introducir aquí? Como se mencionó anteriormente, se desea introducir una larga serie de SBATCH directives. Antes de eso, debes asegurarte de que introduzcas en tu asunto (sheebang).
- En `#!/bin/bash` podemos ver que esto es un bash script. Si estás corriendo un bash script, debes asegurarte de usar bash. Mientras que si estás usando csh, debes asegurarte de que estás usando tsch.
- De cualquier manera, si no colocas la línea del tipo `#!/bin/bash` entonces Slurm no será capaz de determinar cómo debe correr el script.
- En los SBATCH directives, el número de tasks están extendidos por la cantidad de nodos. Para “n” tasks sobre “m” nodos, se tendrán n/m tasks por nodo.
- Luego, se nota que se determinan el account y el partition.
- Finalmente, se nota el flag “-o”, el cual le dice al scheduler cuál es el nombre de salida de su salida estándar. Si el script tuviera errores, querrás escribirlos en la salida estándar para que de una manera puedas ver esos errores, esto hace el flag “-o”.
- qe- es un nombre, tú puedes escribir cualquier nombre que tú quieras.
- %j es una sintaxis especial en Slurm, y solamente funciona en Slurm, el cual usará el job number en lugar de cualquier otra cosa; así, adjuntará el job number a tu output name.

El siguiente paso es configurar tu ambiente.

- Cuando estás usando módulos, se recomienda que cuando uses batch scripts, empieces haciendo un module purge. Esto te inicia con un ambiente limpio y hace las cosas más predecibles cuando estás ahí. Slurm transferirá las variables de entorno a su entorno actual.
- module load intel impi qe configura el ambiente en el que quiero estar ejecutando.

- Las variables configuradas desde NAME hasta SCRATCH dependen totalmente del usuario.

### **Slurm Interactive Jobs**

Tutorial sobre cómo correr jobs interactivos.

¿Qué es un job interactivo?

- Es un job que brinda acceso de consola a uno de los nodos informáticos del Utah CHPC.
- En lugar de correr tu job en un batch script, simplemente te conectarías a tu sesión interactiva y correr tu trabajo como si estuvieras en un nodo interactivo. Y usar todas las capacidades del nodo a tu contenido.

¿Cuándo correr un job interactivo?

Software requiere GUI para correr en multi-core, multi-threaded fashion.

Pruebas de un MPI/Parallel/Multi-threaded software.

Si quieres jugar con un nodo de computación multi-core. Para aprender cosas y tener una idea de cómo se desempeñan.

### **Conectando con los recursos HPC**

Se verá cómo conectarnos a los recursos de los clústeres HPC en CHPC.

Cuando te conectas a un cluster, te conectas a lo que se llama “nodo interactivo”.

Un nodo interactivo es un lugar orientado al público, para usuarios que trabajos de peso ligero: managing, files, data, compilación y desarrollo de software, visualzation, etc.

Los nodos interactivos son recursos compartidos, debes ser cuidadoso con ellos.

No queremos hacer nada que requiere una significativa carga computacional. Para eso, necesitamos enviar jobs via el sistema batch, para que puedan correr en nodos de computación del cluster.

- Todos los clústeres en CHPC (kingspeak-ember-tangent-lonepeak) son servidores basados en Linux, y el camino estandar de conectarse a un servidor basado en Linux es por SSH (“secure shell”). SSH te da acceso al terminal del cliente, o linea de comandos, que nos permite trabajar.
- Hay otras opciones para conectarse como FastX el cual es un diferente tipo de software, que tiene muchas características graficas coonstruidas. El cual se verá próximamente.
- Todos necesitan saber cómo conectarse mediante SSH.
- Como conectarse via SSH depende en cual OS (Operative System) es instalado en tu Client Workstation. En Mac y Linux, el cliente SSH está incorporado.
- Para aplicaciones gráficas, necesitarás configurar un servidor X11 como Xming o habilitar el reenvío de X11, dependiendo de cual sea tu sistema operativo (OS).

### **En Mac o Linux:**

En cuanto tienes la terminal abierta, es realmente directo conectarse. Para ingresar, tu nombre de usuario de CHPC en Utah es tu uNID.

En mi caso, mi uNID es: u6059911, por lo que para ingresar mediante la terminal por SSH, debo hacer:

```
ssh u6059911
```

Luego, debes añadir el símbolo @ para luego usar la dirección del clúster al que te estás conectando. El hostname es el nombre del cluster seguido por chpc.utah.edu.

Por ejemplo, si deseo conectarme al cluster “kingspeak”:

ssh [u6059911@kingspeak.chpc.utah.edu](ssh://u6059911@kingspeak.chpc.utah.edu)

Luego, ingresarás tu contraseña y se te pedirá si deseas guardarla en el dispositivo.

Si necesitas usar aplicaciones gráficas en Linux o Mac, cuando redactes en SSH, necesitas hacer una cosa adicional, el cual es habilitar el reenvío X11. Esto se hace así:

ssh -X [u6059911@kingspeak.chpc.utah.edu](ssh://u6059911@kingspeak.chpc.utah.edu)

Esto habilita el reenvío X11. En un escritorio Linux, ya se debería tener X11, y una aplicación fácil que puedes usar para probar el reenvío X11 es llamada "xeyes". Así:

xeyes

Si deseas salir, el comando es simplemente "exit"

## Secure Shell

ssh es el camino usual para ganar acceso al shell para sistemas basados en Linux de CHPC, como los clusters HPC. Se requiere ssh para conectar ya que encripta la conexión y es mucho más seguro que otros métodos de inicio de sesión por línea de comandos.

## FastX

Se usará FastX para conectar a los recursos CHPC.

- FastX es usado para conectar recursos Linux gráficamente, si deseas usar un ambiente de escritorio o solo conectarse a un terminal mediante la interfaz gráfica.
- Hace multitareas más simples, así como puedes tener un ambiente de escritorio y tener muchas ventanas abiertas al mismo tiempo, en lugar de conectarse a múltiples sesiones ssh.
- Se puede usar para sesiones persistentes (comenzar trabajando en una computadora, y terminar en otra), lo cual es genial si trabajas desde casa.

### ¿Dónde puedo utilizar esto?

CHPC tiene una licencia de sitio para FastX2, lo cual significa que está instalado en muchos de los nodos interactivos. Sin embargo, debes respetar las políticas, y si tienes algo que es computacionalmente intensivo, necesitarás usar un nodo computacional.

- FastX2 server está instalado en muchos nodos.
- Los nodos frisco son recomendados (usado para trabajo gráfico).

Si estás haciendo trabajo gráfico, los nodos frisco son recomendados en lugar de un nodo interactivo normal como los nodos interactivos kingspeak, porque todos los nodos friscos tienen tarjetas gráficas. Se tiene frisco1 hasta frisco8, todos de ellos tienen tarjeta gráfica, por lo que puedes usar aceleración de hardware y cualquier programa que necesite librerías gráficas más complicadas.

### Actualizar a las instrucciones de instalación

Ha habido un ligero cambio en la forma de instalar el cliente de escritorio de FastX en todos los sistemas operativos.

Vamos a ir a un nodo interactivo kingspeak, para eso, ponemos en el navegador:

`kingspeak.chpc.utah.edu:3000`

Y en lugar de tener un enlace para descargar el cliente aquí, tendrás que loguearte con tu uNID y contraseña. Ahora, deberá aparecer un enlace en la esquina inferior derecha, que te lleva a una página donde puedes descargar FastX Client, el software para tu sistema operativo. Nuevamente, hay solo un paso adicional donde necesitarás iniciar sesión cuando alcanzas la interfaz web de FastX antes de descargar el software.

### Introduction to FastX (Updated 45:50)

#### Instalando FastX en Linux

Adicionalmente:

- [Información sobre GPUs en CHPC](#)
  - Información de placa Nvidia en la subsección ***GPU Programming Environment***
- [Documentación de Software en CHPC](#)
  - Información sobre versiones de Python
  - Información sobre creación de un entorno de programación Python
- [Información sobre Sistemas Operativos en CHPC](#)
- [Módulo de Deep Learning de CHPC \(Información sobre CUDA\)](#)

## GPUs en CHPC

CHPC tiene un número limitado de nodos computacionales cluster con GPU.

### Descripción general del hardware GPU

- Los dispositivos GPU se encuentran en los clústeres Kingspeak, Notchpeak y Redwood.
- El uso de los nodos GPU en notchpeak y redwood no afecta su allocation (su uso no cuenta en contra de una asignación que pueda tener el grupo).
- El acceso a los nodos GPU están restringidos a usuarios con cuenta especial para GPU, el cual debe ser solicitada a [helpdesk@chpc.utah.edu](mailto:helpdesk@chpc.utah.edu)

Hay dos categorías de nodos GPU:

- Nodos generales: Comprados por CHPC, y puede ser usado por cualquier usuario con una cuenta CHPC. Se puede ingresar a través de la partición `$(clustername)-gpu`, donde `$(clustername)` puede ser notchpeak, kingspeak o lonepeak en el entorno general, o redwood en el entorno protegido.
- Nodos de propietario: Comprados por el grupo de investigación, identificables con `$(owner)-gpu-$(cluster)`, donde `$(cluster)` puede ser identificado por np, kg, lp o rw.

Para ver las especificaciones de la GPU en los nodos, puedes usar la [versión si del comando sinfo](#) junto con el nombre de la partición, así:

```
si -p notchpeak-gpu
```

Al correr, AVAIL\_FEATURES incluye información del tipo de GPU (v100, 3090, 2080ti, etc). Para información adicional sobre las especificaciones y rendimiento por cada uno de los diferentes GPUs, la mejor fuentes es hacer una búsqueda en la GPU. AVAIL\_FEAUTRES también incluye información sobre el resto del nodo como la generación de CPU (skl, mil,

[etc](#)), el recuento de núcleos físicos en la CPU (cxx, con xx es el recuento de núcleos), y la cantidad de memoria de la CPU (mxxx), donde mxx es la cantidad de memoria en GB.

Para averiguar cuántos gpus hay por nodo, se puede usar el comando `scontrol show node (nodename)`, por ejemplo, `scontrol show node notch293`

### Tipos de GPU en CHPC

- P40: Gen Pascal, 24 GB GDDR5X, 12 TFlops SP, 0.35 TFlops DP, 346 GB/s, 3584 núcleos CUDA.
- Titan V: Gen Volta, 12 GB HBM2, 13.8 TFlops SP, 6.9 TFlops DP, 652.8 GB/s, 5120 núcleos CUDA.
- GTX 1080 Ti: Gen Pascal, 11 GB GDDR5X, 10.6 TFlops SP, 0.33 TFlops DP, 484 GB/s, 3584 núcleos CUDA.
- RTX 2080 Ti: Gen Turing, 11 GB GDDR6, 11.75 TFlops SP, 0.37 TFlops DP, 616 GB/s, 4352 núcleos CUDA.
- GeForce TitanX: Gen Maxwell, 12 GB, ~7 TFlops SP, <200 GFlops DP, núcleos CUDA no especificados.
- Tesla P100: Gen Pascal, 16 GB, 9.3 TFlops SP, 4.7 TFlops DP, 18.7 TFlops HP, PCIe Gen3, 3584 núcleos CUDA.
- Tesla V100: Gen Volta, 16 GB, 14 TFlops SP, 7 TFlops DP, 900 GB/s, 5120 núcleos CUDA.
- Tesla A100: Gen Ampere, 40/80 GB, 9.7 TFlops DP, 1555 GB/s, 6912 núcleos CUDA.
- RTX 3090: Gen Ampere, 24 GB, 35.7 TFlops SP, 936 GB/s, 10,496 núcleos CUDA, 328 núcleos Tensor.
- Tesla T4: Gen Turing, 16 GB, 8.1 TFlops SP, 65 TFlops MP, 320 GB/s, 2560 núcleos CUDA, 320 núcleos Tensor.
- Tesla A40: Gen Ampere, 48 GB, 696 GB/s, 10,752 núcleos CUDA, 84 núcleos RT, 336 núcleos Tensor.
- Tesla A30: Gen Ampere, 24 GB, 5.2 TFlops DP, 933 GB/s, núcleos CUDA no especificados.
- RTX A6000: Gen Ampere, 48 GB, 768 GB/s, 10,752 núcleos CUDA.

### Entorno de programación GPU

- Todas las GPU de CHPC son de Nvidia. Se ofrecen herramientas de programación Nvidia (CUDA, PGI CUDA Fortran y compiladores OpenACC)
- Las últimas herramientas de programación están incluidas en el Nvidia HPC SDK, disponible por `module load nvhpc`, que incluye compiladores de Nvidia CUDA, como los compiladores GPI, bibliotecas, debuggers y profilers de Nvidia GPU.
- Se puede cargar explícitamente una versión CUDA con `module load cuda/version`. Se pueden ver las versiones de CUDA disponibles usando `module spider cuda`, y así usar la que esté disponible. Los compiladores PGI obsoletos que vienen con su propio CUDA, pueden ser configurados usando el módulo PGI, `module load pgi`.
- Para compilar código CUDA en los cuatro tipos de GPU, deben usarse las [banderas de compilador](#) `-gencode arch=compute_20,code=sm_20` `-gencode arch=compute_37,code=sm_37`

```

arch=compute_52,code=sm_52 -gencode
arch=compute_60,code=sm_60 -gencode
arch=compute_70,code=sm_70

```

### **Sobre compiladores HPC de Nvidia...**

- Los compiladores HPC de Nvidia (antes compiladores PGI) especifican la arquitectura de la GPU con la bandera `-tp=tesla`. Si no se especifica ninguna otra opción, la bandera generará código para todas las capacidades de cómputo disponibles (en el momento de escribir cc20, cc30, cc35, cc50 y cc60). Para ser específico para cada GPU, se puede usar `-tp=tesla:cc20` para la M2090, `-tp=tesla:cc50` para la TitanX y `-tp=tesla:cc60` para la P100.
- En 2020, Nvidia lanzó el HPC Software Development Kit (SDK) que combina antiguos compiladores de Portland Group con el compilador Nvidia CUDA. Los compiladores Nvidia HPC SDK reemplazaron entonces a los compiladores PGI y CUDA.
- Los compiladores Nvidia están disponibles como módulos `nvhpc`. Para usar el compilador, los usuarios deben cargar el módulo que define las rutas y algunas otras variables de entorno. El valor predeterminado suele apuntar a la última versión.
- `module load nvhpc`, los compiladores se invocan como `nvc`, `nvc++`, `nvfortran` para C, C++ y Fortran, o nombres PGI antiguos como `pgcc`, `pgc++` y `pgfortran`. El nombre de compilador de CUDA sigue siendo el mismo que `nvcc`. Para una lista de flags disponibles, usar el man page (`man nvc`).
- Para encontrar la versión del compilador, se usa flag `-version`, es decir, `nvc -version`. La versión más reciente de NVHPC que admite todos los clústeres HPC es 21.5, la predeterminada. Versiones a partir de la 22.1 solo funcionan en notchpeak
- Más información en [Documentación de compiladores HPC](#)

Tutoriales de programación GPU [aquí](#).

Al ejecutar código GPU, vale la pena verificar los recursos que está utilizando el programa para asegurarse de que la GPU se utilice bien, a partir del comando `nvidia-smi`, donde observarás la utilización de la memoria y CPU, y configurar varias opciones de la GPU. `nvidia-smi -help` para conocer todas las funciones del comand; por ejemplo, `nvidia-smi -L` enlista las propiedades de tarjeta GPU.

Puedes verificar la utilización de las GPU asignadas a un trabajo por lotes (batch job) desde un nodo interactivo con el comando y número de job: `srun -pty -jobid XXXX nvidia-smi`. Lo cual devolverá los resultados de `nvidia-smi` para las GPU asignadas al job.

## **Módulos en CHPC**

Verificamos que los módulos están implementados correctamente:

```
Sebastian — u6059911@kingspeak2:~ — ssh u6059911@kingspeak.chpc...
[Sebastian@Marielas-MacBook-Air ~ % ssh u6059911@kingspeak.chpc.utah.edu ]
[(u6059911@kingspeak.chpc.utah.edu) Password: ]
      Center for High Performance Computing
      -----
Help: Please send issues or questions to helpdesk@chpc.utah.edu

News: https://www.chpc.utah.edu/news/latest_news/index.php

Tip: Be sure you are subscribed to CHPC mailing list
      (chpc-hpc-users@lists.utah.edu) and checking the associated
      email to be notified of upcoming downtimes, presentations,
      and changes.
-----

               kingspeak
               -----
The general nodes of this cluster are run unallocated; jobs on
general nodes are not subject to preemption.
-----

Last login: Thu May 23 13:25:14 2024 from 187.191.38.52
[(u6059911@kingspeak2:~)]$ module list
```

Podemos implementar módulos a partir de `module load`

```
[(u6059911@kingspeak2:~)]$ module list

Currently Loaded Modules:
  1) chpc/1.0 (S)

Where:
  S: Module is Sticky, requires --force to unload or purge

[(u6059911@kingspeak2:~)]$ module load matlab
[(u6059911@kingspeak2:~)]$ module list

Currently Loaded Modules:
  1) chpc/1.0 (S)  2) matlab/R2023b

Where:
  S: Module is Sticky, requires --force to unload or purge

[(u6059911@kingspeak2:~)]$ module swap matlab/R2023b matlab/R2023a

The following have been reloaded with a version change:
  1) matlab/R2023b => matlab/R2023a

[(u6059911@kingspeak2:~)]$ module list

Currently Loaded Modules:
  1) chpc/1.0 (S)  2) matlab/R2023a

Where:
  S: Module is Sticky, requires --force to unload or purge
```

## Sistema Operativo en CHPC



- CHPC está actualizando sus sistemas Linux de CentOS7 a RockyLinux8. La mayoría de instalaciones de CentOS7 corren en RockyLinux8, por lo que se ha conservado la mayoría de aplicaciones.
- FastX: En RockyLinux3, [FastX](#) 3 está instalado.
- Módulos: Eliminación de módulos para instalaciones que no se ejecutan y limpieza de módulos antiguos.
- Python: RockyLinux8 no proporciona el comando Python, se deberá usar solo Python2 o Python3. A veces el comando Python es necesario; por ello, se proporcionan dos módulos (python/2.7.18 y python/3.6.8) que configuran el comando python en python2 o python3. CHPC recientemente instaló python 3.10.3 en RockyLinux8, que es el python3 recomendado. Las versiones anteriores de Python, como la 3.6.8, no funcionan.

## Documentación de Software CHPC (Python)

La versión predeterminada de Python en los clústeres Rocky 8 es `/usr/bin/python3`. Para obtener una versión de Python más reciente, hay dos opciones.

- [Instalar un propio Anaconda o Miniconda](#). Las últimas versiones de Anaconda2 & Anaconda3 están disponibles dentro del árbol /uufs de CHPC.
- Usar uno de los stacks de Python instaladas por CHPC. Esto conviene si nuestro código de Python no requiere muchos módulos externos o si no están proporcionados por Anaconda.

CHPC admite, dentro de OS RockyLinux8, la versión de Python 3.10.3 (python3), el cual contiene bastantes módulos externos para cálculos científicos. Si deseas usar Python 3.10.3, se debe ejecutar el comando `module load python/3.10.3`

Estos stacks de Python admiten el subprocesamiento de OpenMP, desactivada de forma predeterminada. Para activarla, se debe configurar la variable de entorno `OMP_NUM_THREADS` en `x` (con `x` menor o igual al número de núcleos en la máquina).

```
export OMP_NUM_THREADS=x # Bash
setenv OMP_NUM_THREADS x # Tcsh
```

### Entorno de desarrollo integrado

CHPC recomienda usar PyCharm Community como Python IDE (PyCharm debe ser descargado en el directorio personal). Como el IDE requiere conexión gráfica a los servidores CHPC, es necesario ejecutar FastX para usar PyCharm. Otro buen IDE es [Spyder](#), instalable fácilmente por Anaconda, por lo que se recomienda la [guía de Python instalado por el usuario](#).

Regresando a PyCharm, luego de instalarlo, se puede agregar el directorio bin de PyCharm al PATH:

```
setenv PATH "$HOME/pycharm/bin:$PATH" - for tcsh
export PATH=$HOME/pycharm/bin:$PATH" - for bash
```

O, mejor aún, crear un módulo personalizado, como un archivo `$HOME/mymodules/pycharm/version.lua`

```
-- -*- lua -*-
```

```

help(
[[
This module loads the env. variables
for PyCharm (v. 2016.1.4)
]])
whatis("Name: PyCharm ")
whatis("Version: 2016.1.4 ")
whatis("Category: IDE ")
whatis("Keywords: IDE, Python ")
whatis("Installed on 06/03/2016")
prepend_path("PATH", "/uufs/chpc.utah.edu/common/home/UNID/pycharm/2016.1.4/bin")

```

Y luego agregar el directorio de módulos como un [módulo personalizado](#) y cargar el módulo y ejecutar PyCharm

```

module use $HOME/mymodules
module load pycharm
pycharm.sh

```

## Deep Learning en CHPC

CHPC se ha comprometido a mantener un módulo (llamado deeplearning) con versiones recientes de Python, TensorFlow, Keras y PyTorch para que los usuarios puedan realizar sus proyectos en IA, a partir de paquetes de DL actualizados en Python.

El módulo deeplearning se carga con el módulo CUDA de CHPC, compatible con todos los GPUs de CHPC, a excepción de las GPU Nvidia Tesla K80 del cluster kingspeak.

Module/version	Build Date	Python	Tensorflow	Keras	PyTorch	CUDA
deeplearning/2023.3	03/13/2023	3.10.9	2.11.0	2.11.0	1.13.1+cu117	11.8.0
deeplearning/2023.1	01/05/2023	3.10.8	2.10.0	2.10.0	1.13.1+cu117	11.2
deeplearning/2022.1	01/31/2022	3.9.7	2.6.2	2.6.0	1.10.2+cu102	11.2

Donde las librerías de Python incluidas en este módulo son:

- TensorFlow
- keras
- torch
- torchvision
- torchaudio
- ipykernel

- numpy
- scipy
- sklearn
- skimage
- seaborn
- pandas
- PIL

El módulo deeplearning está implementado en un contenedor Singularity basado en [la imagen-docker jupyter/tensorflow-notebook](#) con módulos torch, torchvision y torchaudio añadidos, y con soporte tanto para jupyter lab como jupyter notebook; lo que lo hace compatible con el [portal web Open OnDemand](#).

Para usar el módulo deeplearning en scripts o en la línea de comandos, cargamos el módulo y ejecutamos python:

```
module load deeplearning  
python
```

Donde esa versión de Python es ejecutada dentro del contenedor singularity e incluye las librerías mencionadas anteriormente, así como otras.