**ASIGNATURA Computación de altas prestaciones**

**Práctica 1 Parte0-S1 : Instalar cluster Rocks utilizando Maquina Virtuales**

# EJERCICIOS de la práctica1-parte0-sección1 (15%)

**Ejercicio 1:**  Pare y reinicie el cluster de manera ordenada. Enumere los pasos que ha realizado para conseguirlo.

**Ejercicio 2:** Suponga que uno de los nodos ha fallado. Elimine el nodo de las bases de datos de rocks y reinstale uno nuevo en su lugar manteniendo el nombre del antiguo, aunque no se mantenga necesariamente su IP. Enumere los pasos que ha realizado para conseguirlo.

**Ejercicio 3:** Realice un script que compruebe el estado de los nodos de computo y sea capaz de devolver el porcentaje de nodos que están activos en el cluster. Adicionalmente la salida debe indicar para cada nodo, su nombre, su dirección IP y su estado. Considere que un nodo no esta activo si la columna states indica  au.

**Ejercicio 4:** Realice un script que añada usuarios al cluster rocks y devuelva como salida el nombre y el password de cada usuario. El nombre debe ser usuBDxx, donde xx se incrementa de 01 a 20 y utilice una política de passwords razonable.

¿Cómo puede verificar cuantos usuarios existen en el cluster y que nombre tienen?

**Ejercicio 5:** Realice las siguientes pruebas de conexión entre el las MV.

5.1.- Conexión desde el anfitrión al cluster ( sería el equivalente a un acceso en remoto);

$ ssh [bigdata@192.168.182.100](mailto:bigdata@192.168.182.100)

5.2.- Desde la sesión anterior conectarse a un nodo de computo:

$ ssh compute-0-0

5.3.- ¿Qué sudera si intentamos la conexión desde el anfitrión a los nodos de computo?

$ ssh [bigdata@10.11.12.](mailto:bigdata@10.11.12.)252

**Práctica 1 – parte0-S2 : Ejecución y Planificación de tareas en cluster Rocks**

# EJERCICIOS: Práctica 1 – Parte0 – Sección2 (15%)

**Ejercicio 6:** Con un editor cree un fichero con el siguiente contenido

compute-0-0

compute-0-1

compute-0-2

y denomínelo **maquinasMPI.txt**

Pruebe el siguiente comando:

$ /opt/openmpi/bin/mpirun -np 10 –machinefile maquinasMPI.txt hostname

y explique como varía el resultado respecto al comando:

/opt/openmpi/bin/mpirun -np 10 hostname

Responda a las siguientes preguntas:

* ¿Se ha ejecutado algún proceso en el frontend?
* ¿Cómo se puede ejecutar parte de los procesos en el frontend?
* Varíe el número de procesos e intente deducir el reparto de tareas que se utiliza.

**Ejercicio 7:** Compile los ejemplos de mpi disponibles en /opt/mpi-tests/e indique como funcionan.

**Ejercicio 8:** Crear una cola denominada colapares.q basándose en la cola all.q que tenga solo los nodos con nombres compute-0-x, siendo x par. Compruebe su funcionamiento.

**Indique los comandos utilizados para su creación y el proceso para comprobar su funcionamiento.**

**Ejercicio 9:** Cree nuevas colas de acuerdo a criterios que le parezcan significativos, por ejemplo, cola1core para máquinas con un solo procesador y cola2core para máquinas con dos procesadores. Para ello reinstale el nodo 1 con 2 cores y cree una nuevo nodo compute-0-3 con 2 cores. Realice pruebas para comprobar el funcionamiento de las colas creadas.

**Indique los comandos utilizados para su creación y el proceso para comprobar su funcionamiento.**

**Práctica1 Parte 1 (35%)**

**Vector processing and SIMD**

You must write a report answering the questions proposed in each exercise, plus the requested files. Submit a zip file through Moodle. Check submission date in Moodle (deadline is until 11:59 pm of that date).

* Exercise 1:
  + Identify your CPU model and list the supported SIMD instructions.

CPU Model:

Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz

SIMD ISA:

mmx sse sse2 ssse3 sse4\_1 sse4\_4 avx avx2 fma adx aes bmi1 bmi2 clfushopt fsgsbase fxsr invpcid monitor movbe mpx pclmulqdq popcnt rdrand rdseed rdtscp sxavec xsaveopt

* + Explain the main differences between both assembly codes (vectorized and non-vectorized) focused on the SIMD instructions generated by the compiler.

The main differences are, that we process multiple doubles at onces which prevented us from having to calculating the same operation multiple times.

Examples:

. p2align -movdqa

mulsd - mulpd

movsd - movaps

addq - paddd

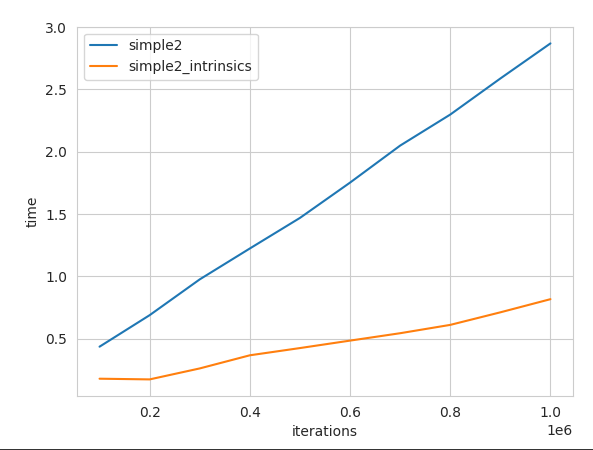
* Exercise 2:
  + Provide the source code of *simple2\_intrinsics.c* after the vectorization of the loops. Explain how you have carried out the vectorization of the code.

Instead of iterating over the array in steps of 1 when initializing and calculating the formula.

We are initializing/calculating 4 Elements at a time.

We use intrinsics like‘\_mm256\_storeu\_pd’, ‘\_mm256\_load\_pd’,

‘\_mm256\_fmadd\_pd’ and ‘\_mm256\_hadd\_pd’.

* + Compare the execution time for different values of NUMBER\_OF\_TRIALS: from 100.000 to 1.000.000 in steps of 100.000. Plot the results in a graph. Discuss the results.

We can see that we achieve a speedup of approximately 3x by using intrinsics.

* Exercise 3:
  + The program includes two loops. The first loop (indicated as Loop 0) iterates over the arguments applying the algorithm to each of them. The second loop (indicated as Loop 1) computes the grey scale algorithm. Is this loop optimal to be vectorized? Why?

Loop 0 is not optimal to be vectorized becauseit is iterating over the files,

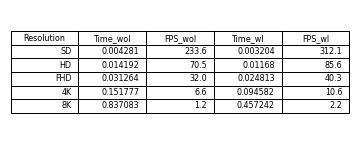
they have to be worked on serially.

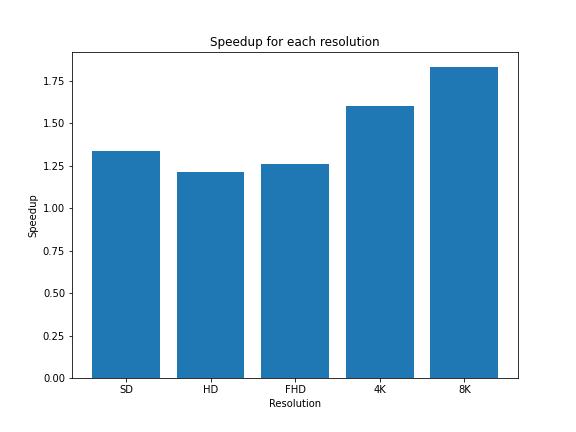
Loop 1 is good to be vectorized,

we can calculate the grey value of multiple pixles in parralel and achieve a massive speedup.

* + Provide the source code of the auto-vectorized version of the code. Explain the changes in the code to help the compiler to vectorize the loop.
  + Provide the source code after manually vectorizing the code. Explain your solution.
  + Fill in a table with time and speedup results compared to the original version and auto-vectorized version for images of different resolutions (SD, HD, FHD, UHD-4k, UHD-8k). You must include a column with the fps at which the program would process. Discuss the results.

woI – without intrinsics

wI - with Intrinsics



The achieved speedup is significant but it will not be sufficient for life video processing in FHD/60FPS or 4k/30FPS.

We can probably achieve even better results by using the full potential of intrinsics.