

# Arquitectura de Computadores (AC)

## Cuaderno de prácticas.

### Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos):

Grupo de prácticas y profesor de prácticas:

## Parte I. Ejercicios basados en los ejemplos del seminario práctico

Crear el directorio con nombre bp0 en atcgrid y en el PC (PC = PC del aula de prácticas o su computador personal).

**NOTA:** En las prácticas se usa slurm como gestor de colas. Consideraciones a tener en cuenta:

- Slurm está configurado para asignar recursos a los procesos (llamados *tasks* en slurm) a nivel de core físico. Esto significa que por defecto slurm asigna un core a un proceso, para asignar x se debe usar con sbatch/srun la opción `--cpus-per-task=x` (`-cx`).
- En slurm, por defecto, `cpu` se refiere a cores lógicos (ej. en la opción `-c`), si no se quieren usar cores lógicos hay que añadir la opción `--hint=nomultithread` a sbatch/srun. Para que con sbatch se tenga en cuenta `---hint=nomultithread` se debe usar srun dentro del script delante del ejecutable.
- Para asegurar que solo se crea un proceso hay que incluir `--ntasks=1` (`-n1`) en sbatch/srun.
- Para que no se ejecute más de un proceso en un nodo de cómputo de atcgrid hay que usar `--exclusive` con sbatch/srun (se recomienda no utilizarlo en los srun dentro de un script).
- Los srun dentro de un *script* heredan las opciones fijadas en el sbatch que se usa para enviar el script a la cola (partición slurm).
- Las opciones de sbatch se pueden especificar también dentro del *script* (usando `#SBATCH`, ver ejemplos en el script del seminario)
- Se recomienda escribir las órdenes directamente en la ventana de comandos (*shell*) en lugar de usar copy/paste.

- Ejecutar `lscpu` en el PC, en atcgrid4 (usar en este caso `-p ac4`) y en uno de los restantes nodos de cómputo (atcgrid1, atcgrid2 o atcgrid3, usar en este caso `-p ac`).

(a) Mostrar con capturas de pantalla el resultado de estas ejecuciones.

MI PC

```
Archivo Acciones Editar Vista Ayuda
[icaro@kali:~]
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Address sizes:          48 bits physical, 48 bits virtual
Byte Order:             Little Endian
CPU(s):                 8
On-line CPU(s) list:    0-7
Vendor ID:              AuthenticAMD
Model name:             AMD Ryzen 7 4700U with Radeon Graphics
CPU family:             23
Model:                  96
Thread(s) per core:     1
Core(s) per socket:     8
Socket(s):              1
Stepping:               1
Frequency boost:        enabled
CPU max MHz:            2800.0000
CPU min MHz:            1400.0000
BogoMIPS:               3992.42
Flags:                  fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl nonstop_tsc cpuid extd_apicid aperfmperf
f rapl mri pcmulopq monitor ssse3 fma c16 sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand lahf_lm cmp_legacy svm extapic cr3_legacy abm sse4a misalignsse 3dnowprefetch osvw ibs skinit wdt tce topoext perfctr_core perfctr_nb bpext perfctr_llc mwaitx cpb cat_l3 cdp_l3 hwpstate ssbd mba lbrs ibpb stibp vmmcall fsgsbase bmi1 avx2 smep bmi2 cqm rdt_a rdseed adx smap clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 xsave
s cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local clzero irperf xsaveerptr rdpru wbnoinvd arat npt lbrv svm_lock nrip_save tsc_scale vmcb_clean flushbyasid decodeassists pausefilter pfthreshold avic v_vmsave_vm
load vgif v_spec_ctrl umip rdpid overflow_recov succor smca

Virtualization features:
Virtualization:         AMD-V
Caches (sum of all):
L1d:                    256 KiB (8 instances)
L1i:                    256 KiB (8 instances)
L2:                     4 MiB (8 instances)
L3:                     8 MiB (2 instances)
NUMA:
NUMA node(s):           1
NUMA node0 CPU(s):      0-7
Vulnerabilities:
Itlb multihit:          Not affected
L1tf:                   Not affected
Mds:                    Not affected
Meltdown:              Not affected
Spec store bypass:      Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Spectre v1:             Mitigation: usercopy/swapgs barriers and __user pointer sanitization
Spectre v2:             Mitigation: Full AMD retpoline, IBPB conditional, IBRS_FW, STIBP disabled, RSB filling
Srbds:                  Not affected
Tsx async abort:        Not affected
```

## ATCGRID 4

```

ac274@atcgrid:~
Archivo Acciones Editar Vista Ayuda
[ac274@atcgrid ~]$ srun -p ac4 lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 64
On-line CPU(s) list:    0-63
Thread(s) per core:     2
Core(s) per socket:     16
Socket(s):              2
NUMA node(s):           2
Vendor ID:              GenuineIntel
CPU family:             6
Model:                  85
Model name:             Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
Stepping:               7
CPU MHz:                1082.812
CPU max MHz:            3200,0000
CPU min MHz:            800,0000
BogoMIPS:               4200.00
Virtualization:         VT-x
L1d cache:              32K
L1i cache:              32K
L2 cache:               1024K
L3 cache:               22528K
NUMA node0 CPU(s):      0-15,32-47
NUMA node1 CPU(s):      16-31,48-63
Flags:                  fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx
                        fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopo
                        logy nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr p
                        dcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefet
                        ch epb cat_l3 cdp_l3 invpcid_single intel_ppin intel_pt ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flex
                        priority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdsee
                        d adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_mbm_total c
                        qm_mbm_local dtherm ida arat pln pts pku ospke avx512_vnni md_clear spec_ctrl intel_stibp flush_l1d arch_capabiliti
                        es

```

## ATCGRID [1-3]

```

ac274@atcgrid:~
Archivo Acciones Editar Vista Ayuda
[ac274@atcgrid ~]$ srun -p ac lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 24
On-line CPU(s) list:    0-23
Thread(s) per core:     2
Core(s) per socket:     6
Socket(s):              2
NUMA node(s):           2
Vendor ID:              GenuineIntel
CPU family:             6
Model:                  44
Model name:             Intel(R) Xeon(R) CPU E5645 @ 2.40GHz
Stepping:               2
CPU MHz:                1600.000
CPU max MHz:            2401,0000
CPU min MHz:            1600,0000
BogoMIPS:               4799.64
Virtualization:         VT-x
L1d cache:              32K
L1i cache:              32K
L2 cache:               256K
L3 cache:               12288K
NUMA node0 CPU(s):      0-5,12-17
NUMA node1 CPU(s):      6-11,18-23
Flags:                  fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx
                        fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology
                        nonstop_tsc aperfmperf eagerfpu pni dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse
                        4_2 popcnt lahf_lm epb ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid dtherm ida arat spec_ctrl intel_s
                        tibp flush_l1d

```

**(b)** ¿Cuántos cores físicos y cuántos cores lógicos tiene atcgrid4?, ¿cuántos tienen atcgrid1, atcgrid2 y atcgrid3? y ¿cuántos tiene el PC? Razonar las respuestas

**RESPUESTA:**

MI PC:

Segun lcpu, tendo 1 socket, un slot de cpu , por lo que tengo 1 cpu/procesador

Veo que core per socket = 8, es decir tengo 8 core por procesador

$1 \text{ socket} * 8 \text{ core/socket} = 8 \text{ core fisico}$

Para los lógicos les multiplico el fisico thread per core = 1

$8 \text{ core fisicos} * 1 \text{ thread per core} = 8 \text{ cores lógicos}$

ATCGRID [4]

2 sockets

16 cores/sockets

2 threads/core

core fisico  $\rightarrow 2 \text{ sockets} * 16 \text{ cores/sockets} = 32 \text{ cores fisicos}$

core lógicos  $\rightarrow 32 \text{ cores fisicos} * 2 \text{ threads/core} = 64 \text{ cores lógicos (threads)}$

ATCGRID[1-3]

2 sockets

6 cores/socket

2 threads/core

core fisico  $\rightarrow 2 \text{ sockets} * 6 \text{ cores/sockets} = 12 \text{ cores lógicos}$

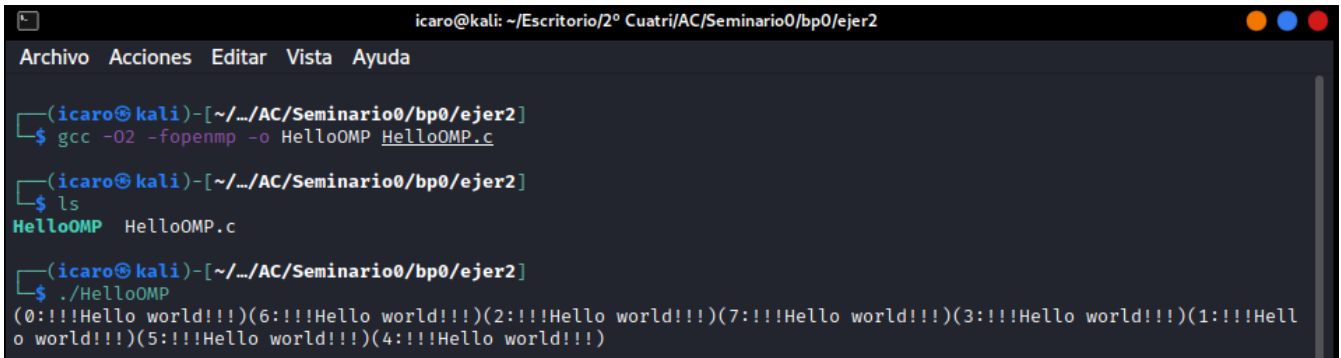
core lógico  $\rightarrow 12 \text{ cores lógicos} * 2 \text{ thread/core} = 24 \text{ cores lógicos (threads)}$

2. Compilar y ejecutar en el PC el código HelloOMP.c del seminario.

(a) Adjuntar capturas de pantalla que muestren la compilación y ejecución en el PC.

Primero creo dentro de BP0 → mkdir ejer2

**RESPUESTA:**



```
icaro@kali: ~/Escritorio/2º Cuatri/AC/Seminario0/bp0/ejer2
Archivo Acciones Editar Vista Ayuda

(icaro@kali)-[~/AC/Seminario0/bp0/ejer2]
$ gcc -O2 -fopenmp -o HelloOMP HelloOMP.c

(icaro@kali)-[~/AC/Seminario0/bp0/ejer2]
$ ls
HelloOMP  HelloOMP.c

(icaro@kali)-[~/AC/Seminario0/bp0/ejer2]
$ ./HelloOMP
(0:!!!Hello world!!!)(6:!!!Hello world!!!)(2:!!!Hello world!!!)(7:!!!Hello world!!!)(3:!!!Hello world!!!)(1:!!!Hello world!!!)(5:!!!Hello world!!!)(4:!!!Hello world!!!)
```

(b) Justificar el número de “Hello world” que se imprimen en pantalla teniendo en cuenta la salida que devuelve ls -l en el PC.

**RESPUESTA:**

Devuelve 8 Hello world

Como mi PC cuenta con 8 cores lógicos, la sentencia que imprime “Hello World !!!” ha sido ejecutada por 8 threads, de manera que cada hebra imprimira por pantalla “Hello world !!!” y su numero de hebra asociada, por tanto habra 8 hello wolrds del 0 al 7.

3. Copiar el ejecutable de `HelloOMP.c` que ha generado anteriormente y que se encuentra en el directorio `ejer2` del PC al directorio `ejer2` de su home en el *front-end* de `atcgrid`. Ejecutar este código en un nodo de cómputo de `atcgrid` (de 1 a 3) a través de cola ac del gestor de colas utilizando directamente en línea de comandos (no use ningún *script*):

(a) `srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithread HelloOMP`

(Alternativa: `srun -pac -Aac -n1 -c12 --hint=nomultithread HelloOMP`)

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

#### RESPUESTA:

```
ac274@atcgrid:~/bp0
Archivo Acciones Editar Vista Ayuda
[ac274@atcgrid bp0]$ ls
HelloOMP
[ac274@atcgrid bp0]$ srun --partition=ac --account=ac --ntasks=1 --cpus-per-task=12 --hint=nomultithread HelloOMP
(0:!!!Hello world!!!)(5:!!!Hello world!!!)(7:!!!Hello world!!!)(1:!!!Hello world!!!)(3:!!!Hello world!!!)(9:!!!Hello world!!!)(4:!!!Hello world!!!)(6:!!!Hello world!!!)(8:!!!Hello world!!!)(11:!!!Hello world!!!)(10:!!!Hello world!!!)(2:!!!Hello world!!!)[ac274@atcgrid bp0]$
[ac274@atcgrid bp0]$ srun -pac -Aac -n1 -c12 --hint=nomultithread HelloOMP
(9:!!!Hello world!!!)(11:!!!Hello world!!!)(1:!!!Hello world!!!)(0:!!!Hello world!!!)(4:!!!Hello world!!!)(3:!!!Hello world!!!)(10:!!!Hello world!!!)(8:!!!Hello world!!!)(6:!!!Hello world!!!)(2:!!!Hello world!!!)(5:!!!Hello world!!!)(7:!!!Hello world!!!)[ac274@atcgrid bp0]$
```

(b) `srun -pac -Aac -n1 -c24 HelloOMP`

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

En comparacion al a, ahora con `-c24` ejecuto 24 procesos

```
ac274@atcgrid:~/bp0
Archivo Acciones Editar Vista Ayuda
[ac274@atcgrid bp0]$ srun -pac -Aac -n1 -c24 HelloOMP
(12:!!!Hello world!!!)(17:!!!Hello world!!!)(6:!!!Hello world!!!)(2:!!!Hello world!!!)(19:!!!Hello world!!!)(4:!!!Hello world!!!)(0:!!!Hello world!!!)(7:!!!Hello world!!!)(13:!!!Hello world!!!)(10:!!!Hello world!!!)(16:!!!Hello world!!!)(23:!!!Hello world!!!)(22:!!!Hello world!!!)(9:!!!Hello world!!!)(15:!!!Hello world!!!)(1:!!!Hello world!!!)(3:!!!Hello world!!!)(21:!!!Hello world!!!)(5:!!!Hello world!!!)(14:!!!Hello world!!!)(8:!!!Hello world!!!)(20:!!!Hello world!!!)(11:!!!Hello world!!!)(18:!!!Hello world!!!)[ac274@atcgrid bp0]$
```

#### RESPUESTA:

(c) `srun -n1 HelloOMP`

Adjuntar capturas de pantalla que muestren el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas. ¿Qué partición (cola) se está usando?

Por defecto se ejecutan 4 procesos y se esta utilizando la particion de colas `atcgrid`

con `sinfo` veo que `*` es el predeterminado , es decir, cuando no se una `-p`.

```
ac274@atcgrid:~/bp0
Archivo Acciones Editar Vista Ayuda
[ac274@atcgrid bp0]$ srun -n1 HelloOMP
(0:!!!Hello world!!!)(1:!!!Hello world!!!)[ac274@atcgrid bp0]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE MODELIST
ac*        up      1:00      3    idle atcgrid[1-3]
ac4        up      1:00      1    idle atcgrid4
[ac274@atcgrid bp0]$
```

**RESPUESTA:**

**(d)** ¿Qué orden `srun` usaría para que HelloOMP utilice todos los cores físicos de `atcgrid4` (se debe imprimir un único mensaje desde cada uno de ellos)?

En `atcgrid4` hay un total de 32 cores físicos

`srun -pac4 --cpus-per-task=32 --hint=nomultithread HelloOMP`

Se debería imprimir un total de 32 hellos → 32 procesos

```
ac274@atcgrid:~/bp0
Archivo Acciones Editar Vista Ayuda
[ac274@atcgrid bp0]$ srun -pac4 --hint=nomultithread HelloOMP
(0:!!!Hello world!!!)[ac274@atcgrid bp0]$ srun -pac4 --cpus-per-task=32 --hint=nomultithread HelloOMP
(25:!!!Hello world!!!)(20:!!!Hello world!!!)(16:!!!Hello world!!!)(8:!!!Hello world!!!)(14:!!!Hello world!!!)(26:!!!Hello world!!!)(22:!!!Hello world!!!)(2:!!!Hello world!!!)(31:!!!Hello world!!!)(6:!!!Hello world!!!)(9:!!!Hello world!!!)(27:!!!Hello world!!!)(19:!!!Hello world!!!)(17:!!!Hello world!!!)(0:!!!Hello world!!!)(11:!!!Hello world!!!)(10:!!!Hello world!!!)(3:!!!Hello world!!!)(30:!!!Hello world!!!)(18:!!!Hello world!!!)(29:!!!Hello world!!!)(23:!!!Hello world!!!)(12:!!!Hello world!!!)(7:!!!Hello world!!!)(1:!!!Hello world!!!)(5:!!!Hello world!!!)(21:!!!Hello world!!!)(24:!!!Hello world!!!)(4:!!!Hello world!!!)(28:!!!Hello world!!!)(15:!!!Hello world!!!)(13:!!!Hello world!!!)(26:!!!Hello world!!!)(10:!!!Hello world!!!)(25:!!!Hello world!!!)(20:!!!Hello world!!!)(9:!!!Hello world!!!)(31:!!!Hello world!!!)(0:!!!Hello world!!!)(15:!!!Hello world!!!)(8:!!!Hello world!!!)(22:!!!Hello world!!!)(17:!!!Hello world!!!)(27:!!!Hello world!!!)(12:!!!Hello world!!!)(29:!!!Hello world!!!)(7:!!!Hello world!!!)(28:!!!Hello world!!!)(1:!!!Hello world!!!)(19:!!!Hello world!!!)(24:!!!Hello world!!!)(14:!!!Hello world!!!)(6:!!!Hello world!!!)(18:!!!Hello world!!!)(2:!!!Hello world!!!)(13:!!!Hello world!!!)(3:!!!Hello world!!!)(5:!!!Hello world!!!)(4:!!!Hello world!!!)(16:!!!Hello world!!!)(21:!!!Hello world!!!)(23:!!!Hello world!!!)(11:!!!Hello world!!!)(30:!!!Hello world!!!)[ac274@atcgrid bp0]$
```

4. Modificar en su PC `HelloOMP.c` para que se imprima “world” en un `printf` distinto al usado para “Hello”. En ambos `printf` se debe imprimir el identificador del thread que escribe en pantalla. Nombrar al código resultante `HelloOMP2.c`. Compilar este nuevo código en el PC y ejecutarlo. Copiar el fichero ejecutable resultante al front-end de `atcgrid` (directorio `ejer4`). Ejecutar el código en un nodo de cómputo de `atcgrid` usando el `script` `script_helloomp.sh` del seminario (el nombre del ejecutable en el script debe ser `HelloOMP2`).

**(a)** Utilizar: `sbatch script_helloomp.sh`. Adjuntar capturas de pantalla que muestren el nuevo código, la compilación, el envío a la cola de la ejecución y el resultado de esta ejecución tal y como la devuelve el gestor de colas.

```
icaro@kali: ~/Escritorio/2º Cuatri/AC/Seminario0/bp0/ejer2
Archivo Acciones Editar Vista Ayuda
GNU nano 6.2 HelloOMP2.c
/**
 *
 * Compilar con gcc -O2 -fopenmp HelloOMP.c -o HelloOMP
 */
#include <stdio.h>
#include <omp.h>

int main(void) {
#pragma omp parallel
    printf("(0:!!!Hello)", omp_get_thread_num());
    printf("(0:world!!!)", omp_get_thread_num());
return(0);
}
```

```

ac274@atcgrid:~/bp0
Archivo Acciones Editar Vista Ayuda
[ac274@atcgrid bp0]$ ls
HelloOMP HelloOMP2 script_helloomp.sh
[ac274@atcgrid bp0]$ sbatch script_helloomp.sh
Submitted batch job 121293
[ac274@atcgrid bp0]$ ls
HelloOMP HelloOMP2 script_helloomp.sh slurm-121293.out
[ac274@atcgrid bp0]$ cat slurm-121293.out
Id. usuario del trabajo: ac274
Id. del trabajo: 121293
Nombre del trabajo especificado por usuario: helloOMP2
Directorio de trabajo (en el que se ejecuta el script): /home/ac274/bp0
Cola: ac
Modo que ejecuta este trabajo:atcgrid.ugr.es
Nº de nodos asignados al trabajo: 1
Nodos asignados al trabajo: atcgrid1
CPUs por nodo: 24

1. Ejecución helloOMP2 una vez sin cambiar nº de threads (valor por defecto)
(5:!!!Hello)(11:!!!Hello)(2:!!!Hello)(1:!!!Hello)(3:!!!Hello)(7:!!!Hello)(9:!!!Hello)(0:!!!Hello)(8:!!!Hello)(4:!!!
Hello)(6:!!!Hello)(10:!!!Hello)(0:world!!!)

2. Ejecución helloOMP2 varias veces con distinto nº de threads:
- Para 12 threads:
(11:!!!Hello)(1:!!!Hello)(7:!!!Hello)(3:!!!Hello)(2:!!!Hello)(4:!!!Hello)(0:!!!Hello)(6:!!!Hello)(10:!!!Hello)(9:!!
!Hello)(5:!!!Hello)(8:!!!Hello)(0:world!!!) - Para 6 threads:
(0:!!!Hello)(4:!!!Hello)(1:!!!Hello)(2:!!!Hello)(3:!!!Hello)(5:!!!Hello)(0:world!!!) - Para 3 threads:
(0:!!!Hello)(2:!!!Hello)(1:!!!Hello)(0:world!!!) - Para 1 threads:
(0:!!!Hello)(0:world!!!)[ac274@atcgrid bp0]$

```

**RESPUESTA:**

(b) ¿Qué nodo de cómputo de atcgrid ha ejecutado el *script*? Explicar cómo ha obtenido esta información.

Esta info se obtiene con \$SLURM\_JOB\_NODELIST que es una variable de entorno del sistema de colas que muestra los nodos asignados al trabajo, por lo que corresponde al nodo de computoatcgrid 1

**RESPUESTA:**

(c) ¿Qué órdenes para el gestor de colas slurm incluye el *script*? Explicar cómo ha obtenido esta información.

Las ordenes del gestor de colas que se han usado ha sido:

--job-name=helloOMP2

--partition=ac

--account=ac

--exclusive

--ntask 1

--cpus-per-task 12 // este luego cambia su valor en el script

Esta informacion se obtiene al principio del script en # donde pone ordenes para el Gestor de carga de trabajo

**RESPUESTA:**

(d) Haga los cambios necesarios en el *script* para que se utilice atcgrid4. Comentar los cambios realizados y los motivos por los que se han hecho.

**RESPUESTA:**

añado --hint=nomultithread y además cambio → --partition=ac4



**NOTA:** Utilizar siempre con `sbatch` las opciones `-n1` y `-c`, `--exclusive` y, para usar cores físicos y no lógicos, no olvidar incluir `--hint=nomultithread`. Utilizar siempre con `srun`, si lo usa fuera de un script, las opciones `-n1` y `-c` y, para usar cores físicos y no lógicos, no olvide incluir `--hint=nomultithread`. Recordar que los `srun` dentro de un *script* heredan las opciones incluidas en el `sbatch` que se usa para enviar el *script* a la cola slurm. Se recomienda usar `sbatch` en lugar de `srun` para enviar trabajos a ejecutar a través slurm porque éste último deja bloqueada la ventana hasta que termina la ejecución, mientras que usando `sbatch` la ejecución se realiza en segundo plano.

## Parte II. Resto de ejercicios

5. Generar en el PC el ejecutable del código fuente C `SumaVectores.c` para vectores locales (para ello antes de compilar debe descomentar la definición de `VECTOR_LOCAL` y comentar las definiciones de `VECTOR_GLOBAL` y `VECTOR_DYNAMIC`). El comentario inicial del código muestra la orden para compilar (siempre hay que usar `-O2`). Incorporar volcados de pantalla que demuestren la compilación y la ejecución correcta del código en el PC (leer lo indicado al respecto en las normas de prácticas).

### RESPUESTA:

compilas com gcc -O2 SumaVectores.c -o SumaVectores

```

icaro@kali: ~/Escritorio/2º Cuatri/AC/Seminario0/bp0
Archivo Acciones Editar Vista Ayuda
GNU nano 6.2 SumaVectores.c *
/**
 *
 * Código para suma secuencial de 2 vectores SumaVectores.c
 *
 */

#include <stdlib.h> // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h> // biblioteca donde se encuentra la función printf()
#include <time.h> // biblioteca donde se encuentra la función clock_gettime()

//Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los ...
//tres defines siguientes puede estar descomentado):
#define VECTOR_LOCAL // descomentar para que los vectores sean variables ...
// locales (si se supera el tamaño de la pila se ...
// generará el error "Violación de Segmento")
// #define VECTOR_GLOBAL // descomentar para que los vectores sean variables ...
// globales (su longitud no estará limitada por el ...
// tamaño de la pila del programa)
// #define VECTOR_DYNAMIC // descomentar para que los vectores sean variables ...
// dinámicas (memoria reutilizable durante la ejecución)

(icaro@kali)~[~/2º Cuatri/AC/Seminario0/bp0]
$ ls
bp0.zip ejer2 script_helloomp.sh SumaVectores.c

(icaro@kali)~[~/2º Cuatri/AC/Seminario0/bp0]
$ gcc -O2 SumaVectores.c -o SumaVectores

(icaro@kali)~[~/2º Cuatri/AC/Seminario0/bp0]
$ ./SumaVectores 12
Tamaño Vectores:12 (4 B)
Tiempo:0.000000230 / Tamaño Vectores:12 / V1[0]+V2[0]=V3[0](1.200000+1.200000=2.400000) / / V1[11]+V2[11]=V
3[11](2.300000+0.100000=2.400000) /

(icaro@kali)~[~/2º Cuatri/AC/Seminario0/bp0]
$ ./SumaVectores 8
Tamaño Vectores:8 (4 B)
Tiempo:0.000000210 / Tamaño Vectores:8
/ V1[0]+V2[0]=V3[0](0.800000+0.800000=1.600000) /
/ V1[1]+V2[1]=V3[1](0.900000+0.700000=1.600000) /
/ V1[2]+V2[2]=V3[2](1.000000+0.600000=1.600000) /
/ V1[3]+V2[3]=V3[3](1.100000+0.500000=1.600000) /
/ V1[4]+V2[4]=V3[4](1.200000+0.400000=1.600000) /
/ V1[5]+V2[5]=V3[5](1.300000+0.300000=1.600000) /
/ V1[6]+V2[6]=V3[6](1.400000+0.200000=1.600000) /
/ V1[7]+V2[7]=V3[7](1.500000+0.100000=1.600000) /

```



6. En el código `SumaVectores.c` se utiliza la función `clock_gettime()` para obtener el tiempo de ejecución del trozo de código que calcula la suma de vectores. El código se imprime la variable `ncgt`,

(a) ¿Qué contiene esta variable?

**RESPUESTA:**

Esta variable contiene el tiempo en segundos que tarda en sumarse 2 vectores, viene dado por la expresion:

```
ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+  
    (double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));
```

consiste en la definicion de tiempo real:

Tiempo = tv\_sec + (tv\_nsec/10<sup>9</sup>)

tv\_sec() → devuelve el valor de tiempo como un entero equivalente al numero de segundos desde the Epoch( Epoca, una epoca, para fines de cronologia y periodizacion, es un instante en el tiempo elegido como el origen de una era de calendario particular, punto de inicio a calcular el tiempo)

tv\_nsec() → devuelve el tiempo en nanosegundos

(b) ¿En qué estructura de datos devuelve `clock_gettime()` la información de tiempo (indicar el tipo de estructura de datos, describir la estructura de datos, e indicar los tipos de datos que usa)?

**RESPUESTA:**

La info del tiempo devuelta por `clock_gettime()` es una estructura de datos **struct timespec**

Es una estructura para especificar el tiempo con alta resolucion

Dato miembro →

time\_t , tv\_sec → numero de segundos

long tv\_nsec → nanosegundos

time\_t es un tipo de dato definido para el almacenamiento de valores de tiempo del sistema

(c) ¿Qué información devuelve exactamente la función `clock_gettime()` en la estructura de datos descrita en el apartado (b)? ¿qué representan los valores numéricos que devuelve?

**RESPUESTA:**

```
int clock_gettime(clockid_t clock_id, struct timespec *tp);
```

La funcion `clock_gettime()` toma el tiempo actual del reloj especificado por `clock_id`, y lo guarda en el buffer apuntado por `tp`.

Devuelve exactamente un dato de tipo entero (int)

Los valores numericos que devuelve son o bien 0 (EXITO) o bien -1 (ERROR)

7. Rellenar una tabla como la Tabla 1 en una hoja de cálculo con los tiempos de ejecución del código SumaVectores.c para vectores locales, globales y dinámicos (se pueden obtener errores en tiempo de ejecución o de compilación, ver ejercicio 9). Obtener estos resultados usando *scripts* (partir del *script* que hay en el seminario). Debe haber una tabla para un nodo de cómputo de atcgrid con procesador Intel Xeon E5645 y otra para su PC en la hoja de cálculo. En la columna “Bytes de un vector” hay que poner el total de bytes reservado para un vector. (NOTA: Se recomienda usar en la hoja de cálculo el mismo separador para decimales que usan los códigos al imprimir “.”. Este separador se puede modificar en la hoja de cálculo.)

**RESPUESTA:****Tabla 1 .** Copiar la tabla de la hoja de cálculo utilizada. El número de componentes se va duplicando.

## PC

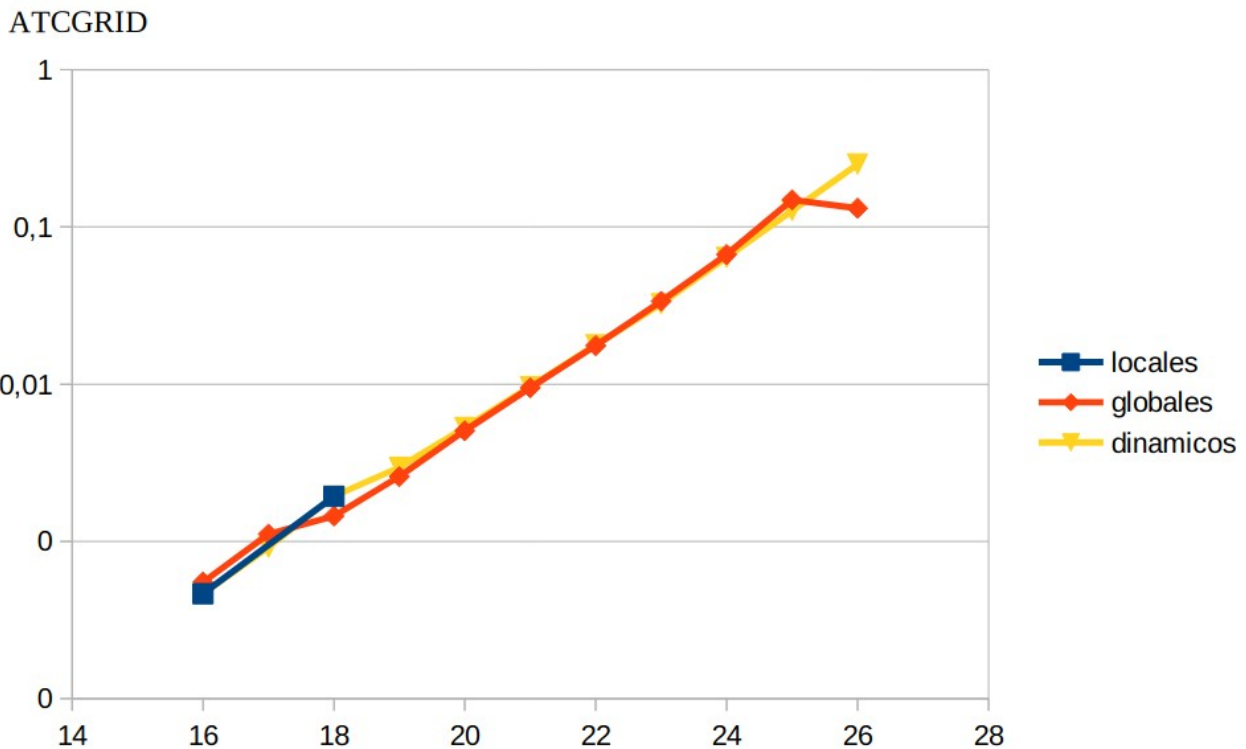
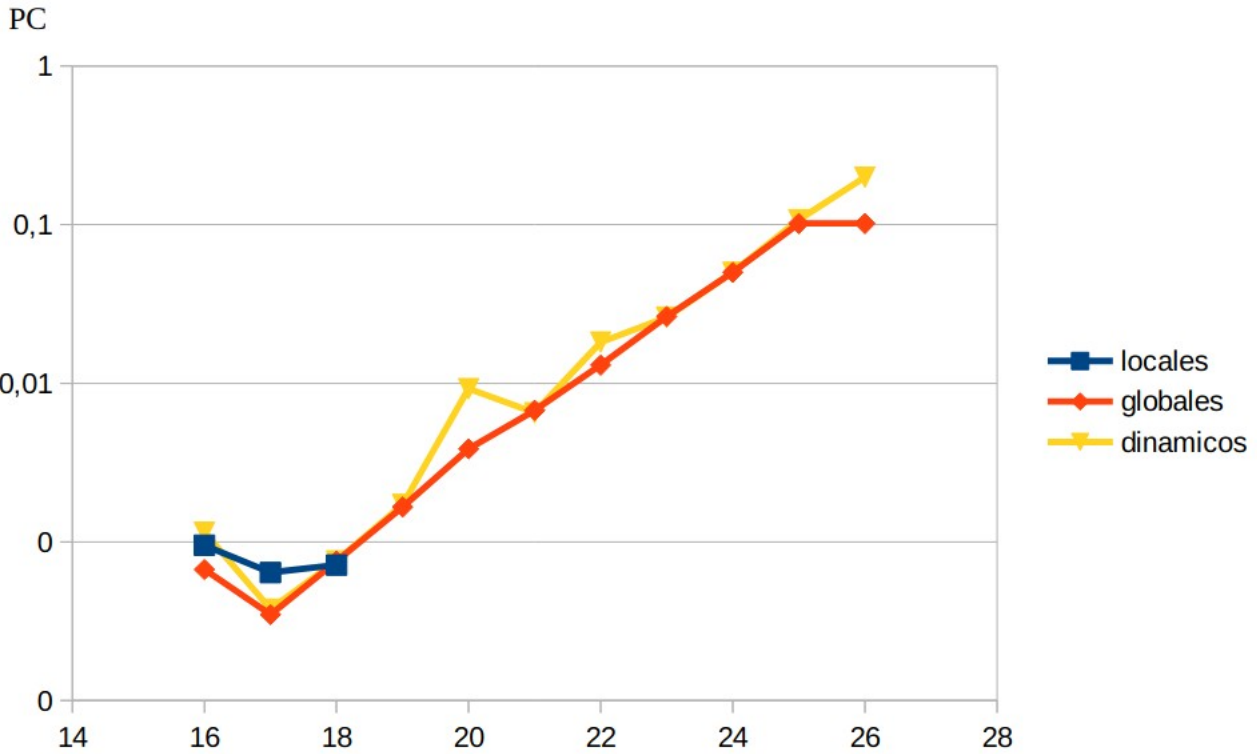
Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	16	0.000355440	0.000668289	0.000302901
131072	17	0.00065911	0.001191366	0.000670724
262144	18	0.001829719	0.002249380	0.001760098
524288	19	Seg fault	0.002705519	0.003304519
1048576	20	“”	0.006456080	0.004758519
2097152	21	“”	0.012447192	0.010030798
4194304	22	“”	0.020919842	0.015956147
8388608	23	“”	0.033071328	0.028677196
16777216	24	“”	0.069346425	0.058110836
33554432	25	“”	0.140520474	0.106981936
67108864	26	“”	0.133804696	0.221259364

## atcgrid

Nº de Componentes	Bytes de un vector	Tiempo para vect. locales	Tiempo para vect. globales	Tiempo para vect. dinámicos
65536	16	0.000483918	0,000550994	0,000465788
131072	17	0.000926810	0,001113260	0,000918296
262144	18	0.001239772	0,001450507	0,001939926
524288	19	Seg fault	0,002584080	0,002957919
1048576	20	“”	0,005058551	0,005310951
2097152	21	“”	0,009480339	0,009653125
4194304	22	“”	0,017580072	0,017761231
8388608	23	“”	0,033724523	0,032450862
16777216	24	“”	0,066779731	0,064063861
33554432	25	“”	0,148799020	0,127272963
67108864	26	“”	0,131476807	0,251244330

1. Con ayuda de la hoja de cálculo representar **en una misma gráfica** los tiempos de ejecución obtenidos en atcgrid y en su PC para vectores locales, globales y dinámicos (eje y) en función del tamaño en bytes de un vector (por tanto, los valores de la segunda columna de la tabla, que están en escala logarítmica, deben estar en el eje x). Utilizar escala logarítmica en el eje de ordenadas (eje y). ¿Hay diferencias en los tiempos de ejecución?

**RESPUESTA:**

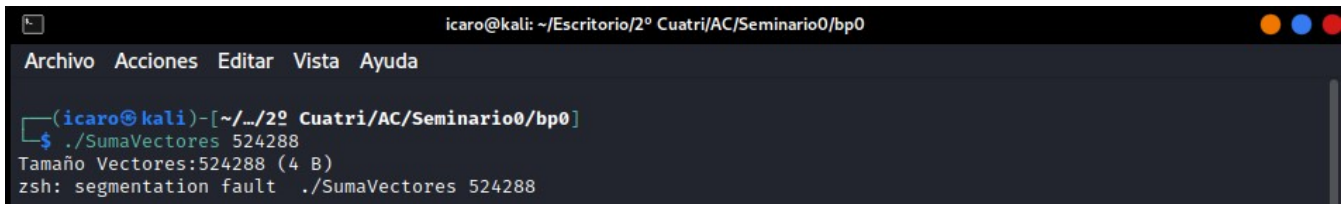


## 2. Contestar a las siguientes preguntas:

(a) Cuando se usan vectores locales, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

Se produce un segmentation fault cuando el tamaño de los vectores supera el tamaño de la pila, concretamente el primer error ocurre cuando el vector tiene  $524288 = 2^{19}$  componentes.

Ulimit -a



```
icaro@kali: ~/Escritorio/2º Cuatri/AC/Seminario0/bp0
Archivo Acciones Editar Vista Ayuda
(icaro@kali)-[~/2º Cuatri/AC/Seminario0/bp0]
$ ./SumaVectores 524288
Tamaño Vectores:524288 (4 B)
zsh: segmentation fault ./SumaVectores 524288
```

(b) Cuando se usan vectores globales, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

### RESPUESTA:

La longitud de esos vectores no está limitada por el tamaño de la pila del programa. No se produce ningún error, aunque en el caso de que se especifique que el tamaño tiene que ser de  $2^{26}$  componentes el programa sustituye dicho valor por  $2^{25}$ .

(c) Cuando se usan vectores dinámicos, ¿se obtiene error para alguno de los tamaños?, ¿a qué cree que es debido lo que ocurre? (Incorporar volcados de pantalla como se indica en las normas de prácticas)

### RESPUESTA:

No se obtiene error para ninguno de los tamaños.

## 3. (a) ¿Cuál es el máximo valor que se puede almacenar en la variable N teniendo en cuenta su tipo? Razonar respuesta.

### RESPUESTA:

Teniendo en cuenta que la variable N es de tipo unsigned int, el cual tiene un tamaño máximo de 4B  $\rightarrow 4 \times 8 = 32$  bits.

el valor máximo que se puede tomar es:

$$N = 2^{32} - 1$$

(b) Modificar el código fuente C (en el PC) para que el límite de los vectores cuando se declaran como variables globales sea igual al máximo número que se puede almacenar en la variable N y generar el ejecutable. ¿Qué ocurre? ¿A qué es debido? (Incorporar volcados de pantalla que muestren lo que ocurre)

### RESPUESTA:

He modificado la línea de código  $\rightarrow \#define MAX 4294967295 // 2^{32} - 1$

### EL error lo da el enlazador, que es el que se encarga de las dependencias de las variables globales

El puntero RX86\_64\_PC32 sirve para direccionar 32B, funciones globales a algunas direcciones de memoria, y si tuviésemos que direccionar más de 32 b no podríamos.

## Entrega del trabajo

Leer lo indicado en las normas de prácticas sobre la entrega del trabajo del bloque práctico en SWAD.

**Listado 1.** Código C que suma dos vectores. Se generan aleatoriamente las componentes para vectores de tamaño mayor que 8 y se imprimen todas las componentes para vectores menores que 10.

```
/* SumaVectoresC.c
   Suma de dos vectores: v3 = v1 + v2

   Para compilar usar (-lrt: real time library, no todas las versiones de gcc necesitan que se incluya
   -lrt):
       gcc -O2 SumaVectores.c -o SumaVectores -lrt
       gcc -O2 -S SumaVectores.c -lrt //para generar el código ensamblador

   Para ejecutar use: SumaVectoresC longitud
*/

#include <stdlib.h> // biblioteca con funciones atoi(), malloc() y free()
#include <stdio.h>  // biblioteca donde se encuentra la función printf()
#include <time.h>   // biblioteca donde se encuentra la función clock_gettime()

//Sólo puede estar definida una de las tres constantes VECTOR_ (sólo uno de los ...
//tres defines siguientes puede estar descomentado):
#define VECTOR_LOCAL // descomentar para que los vectores sean variables ...
// locales (si se supera el tamaño de la pila se ...
// generará el error "Violación de Segmento")
#define VECTOR_GLOBAL // descomentar para que los vectores sean variables ...
// globales (su longitud no estará limitada por el ...
// tamaño de la pila del programa)
#define VECTOR_DYNAMIC // descomentar para que los vectores sean variables ...
// dinámicas (memoria reutilizable durante la ejecución)

#ifndef VECTOR_GLOBAL
#define MAX 33554432 //2^25
double v1[MAX], v2[MAX], v3[MAX];
#endif

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

    int i;
    struct timespec cgt1,cgt2; double ncgt; //para tiempo de ejecución

    //Leer argumento de entrada (nº de componentes del vector)
    if (argc<2){
        printf("Faltan nº componentes del vector\n");
        exit(-1);
    }

    unsigned int N = atoi(argv[1]); // Máximo N =2^32-1=4294967295 (sizeof(unsigned int) = 4 B)
    #ifdef VECTOR_LOCAL
        double v1[N], v2[N], v3[N]; // Tamaño variable local en tiempo de ejecución ...
        // disponible en C a partir de actualización C99
```

```

#endif
#ifdef VECTOR_GLOBAL
if (N>MAX) N=MAX;
#endif
#ifdef VECTOR_DYNAMIC
double *v1, *v2, *v3;
v1 = (double*) malloc(N*sizeof(double)); // malloc necesita el tamaño en bytes
v2 = (double*) malloc(N*sizeof(double)); //si no hay espacio suficiente malloc devuelve NULL
v3 = (double*) malloc(N*sizeof(double));
if ( (v1==NULL) || (v2==NULL) || (v3==NULL) ){
printf("Error en la reserva de espacio para los vectores\n");
exit(-2);
}
#endif

//Inicializar vectores
if (N < 9)
for (i = 0; i < N; i++)
{
v1[i] = N * 0.1 + i * 0.1;
v2[i] = N * 0.1 - i * 0.1;
}
else
{
srand48(time(0));
for (i = 0; i < N; i++)
{
v1[i] = drand48();
v2[i] = drand48(); //printf("%d:%f,%f/",i,v1[i],v2[i]);
}
}

clock_gettime(CLOCK_REALTIME,&cgt1);
//Calcular suma de vectores
for(i=0; i<N; i++)
v3[i] = v1[i] + v2[i];

clock_gettime(CLOCK_REALTIME,&cgt2);
ncgt=(double) (cgt2.tv_sec-cgt1.tv_sec)+
(double) ((cgt2.tv_nsec-cgt1.tv_nsec)/(1.e+9));

//Imprimir resultado de la suma y el tiempo de ejecución
if (N<10) {
printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%lu\n",ncgt,N);
for(i=0; i<N; i++)
printf("/ V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
i,i,i,v1[i],v2[i],v3[i]);
}
else
printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\t/ V1[0]+V2[0]=V3[0](%8.6f+%8.6f=%8.6f) / /
V1[%d]+V2[%d]=V3[%d](%8.6f+%8.6f=%8.6f) /\n",
ncgt,N,v1[0],v2[0],v3[0],N-1,N-1,N-1,v1[N-1],v2[N-1],v3[N-1]);

#ifdef VECTOR_DYNAMIC
free(v1); // libera el espacio reservado para v1
free(v2); // libera el espacio reservado para v2
free(v3); // libera el espacio reservado para v3
#endif
return 0;
}

```

