

Arquitectura de Computadores (AC)

Cuaderno de prácticas.

Bloque Práctico 0. Entorno de programación

Estudiante (nombre y apellidos): Juan Manuel Rubio Rodríguez

Grupo de prácticas: D1

Fecha de entrega:

Fecha evaluación en clase:

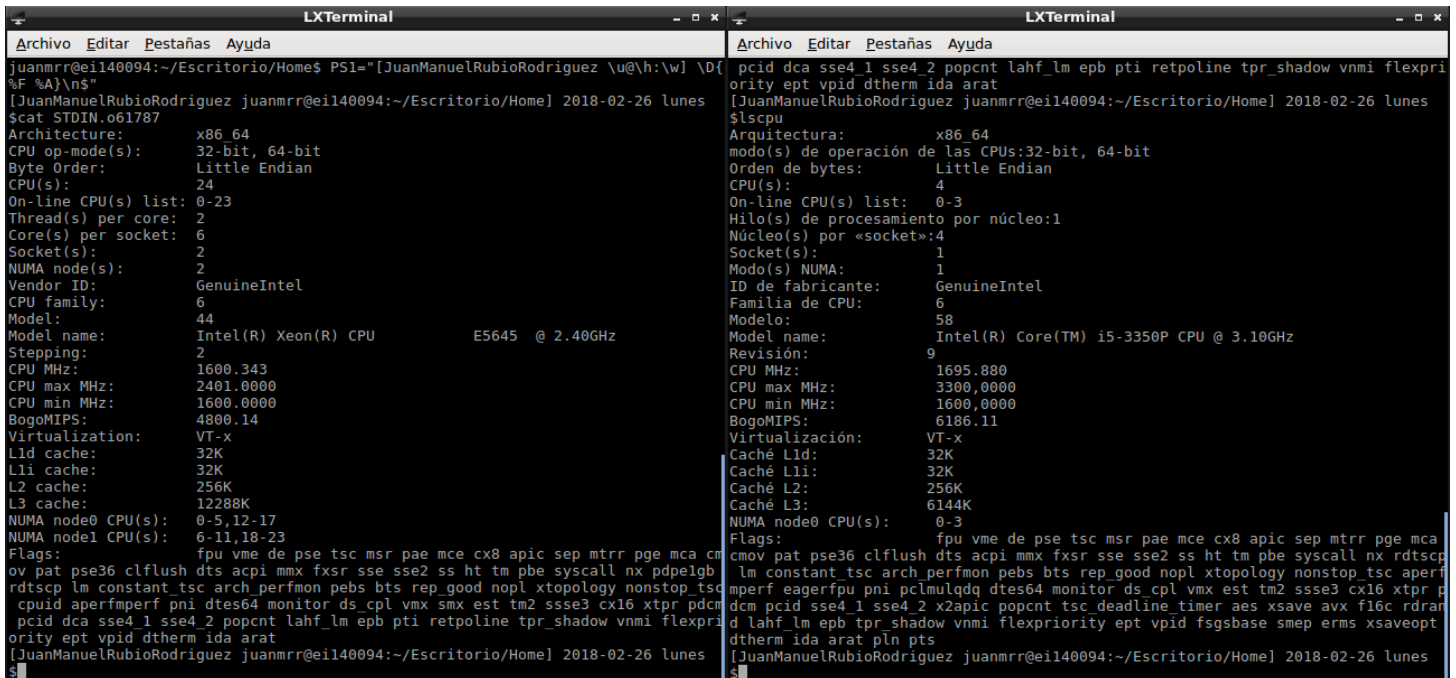
2º curso / 2º cuatr.

Grado Ing. Inform.

Doble Grado Ing.
Inform. y Mat.

1. Incorpore volcados de pantalla que muestren lo que devuelve `lscpu` en atcgrid y en su PC.

CAPTURAS:



```
Archivo Editar Pestañas Ayuda
juanmrr@eil140094:~/Escritorio/Home$ PS1="[JuanManuelRubioRodriguez \u@\h:\w] \D{\
%F %A}\n$"
[JuanManuelRubioRodriguez juanmrr@eil140094:~/Escritorio/Home] 2018-02-26 lunes
$cat STDIN.o61787
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:                1600.343
CPU max MHz:            2401.0000
CPU min MHz:            1600.0000
BogoMIPS:               4800.14
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
ov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb
rdtsmp lm constant tsc arch.perfmon pebs bts rep good nopl xtopology nonstop tsc
aperf cpuid aperfperf pni dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm
pcid dca sse4_1 sse4_2 popcnt lahf_lm epb pti retpoline tpr_shadow vnmi flexpri
ority ept vpid dtherm ida arat
[JuanManuelRubioRodriguez juanmrr@eil140094:~/Escritorio/Home] 2018-02-26 lunes
$

Archivo Editar Pestañas Ayuda
pcid dca sse4_1 sse4_2 popcnt lahf_lm epb pti retpoline tpr_shadow vnmi flexpri
ority ept vpid dtherm ida arat
[JuanManuelRubioRodriguez juanmrr@eil140094:~/Escritorio/Home] 2018-02-26 lunes
$lscpu
Arquitectura:          x86_64
modo(s) de operación de las CPUs:32-bit, 64-bit
Orden de bytes:       Little Endian
CPU(s):               4
On-line CPU(s) list:  0-3
Hilo(s) de procesamiento por núcleo:1
Núcleo(s) por «socket»:4
Socket(s):             1
Modo(s) NUMA:          1
ID de fabricante:      GenuineIntel
Familia de CPU:        6
Modelo:                58
Modelo name:           Intel(R) Core(TM) i5-3350P CPU @ 3.10GHz
Revisión:              9
CPU MHz:               1695.880
CPU max MHz:           3300.0000
CPU min MHz:           1600.0000
BogoMIPS:              6186.11
Virtualización:        VT-x
Caché L1d:             32K
Caché L1i:             32K
Caché L2:              256K
Caché L3:              6144K
NUMA node0 CPU(s):     0-3
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 rdtscp
lm constant tsc arch.perfmon pebs bts rep good nopl xtopology nonstop tsc aperf
mperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr p
dcm pcid sse4_1 sse4_2 x2apic popcnt tsc deadline timer aes xsave avx f16c rdran
d lahf_lm epb tpr_shadow vnmi flexpriority ept vpid fsgsbase smep erms xsaveopt
dtherm ida arat pln pts
[JuanManuelRubioRodriguez juanmrr@eil140094:~/Escritorio/Home] 2018-02-26 lunes
$
```

Conteste a las siguientes preguntas:

- a. ¿Cuántos cores físicos y cuántos cores lógicos tiene atcgrid de prácticas o su PC?

RESPUESTA: Ejecutada la orden `lscpu` sobre el pc del aula, obtengo 4 cores físicos y un único hilo, con lo cual también tengo 4 cores lógicos.

* Este ejercicio está realizado sobre los ordenadores del aula, el resto están ejecutados sobre mi pc personal.

- b. ¿Cuántos cores físicos y cuántos cores lógicos tiene un nodo de atcgrid?

RESPUESTA: Cada nodo tiene dos sockets, cada socket una cpu con 6 cores físicos y dos hilos de ejecución, con lo cual obtengo 12 cores físicos y 24 lógicos.

2. En el Listado 1 se puede ver un código fuente C que calcula la suma de dos vectores y en el Listado 2 una versión con C++:

$v3 = v1 + v2; \quad v3(i) = v1(i) + v2(i), \quad i=0, \dots, N-1$

Los códigos utilizan directivas del compilador para fijar el tipo de variable de los vectores ($v1$, $v2$ y $v3$). En los comentarios que hay al principio de los códigos se indica cómo hay que compilarlos.

Los vectores pueden ser:

- Variables locales: descomentando en el código `#define VECTOR_LOCAL` y comentando `#define VECTOR_GLOBAL` y `#define VECTOR_DYNAMIC`
- Variables globales: descomentando `#define VECTOR_GLOBAL` y comentando `#define VECTOR_LOCAL` y `#define VECTOR_DYNAMIC`
- Variables dinámicas: descomentando `#define VECTOR_DYNAMIC` y comentando `#define VECTOR_LOCAL` y `#define VECTOR_GLOBAL`. Si se usan los códigos tal y como están en Listado 1 y Listado 2, sin hacer ningún cambio, los vectores (v1, v2 y v3) serán variables dinámicas.

Por tanto, se debe definir sólo una de las siguientes constantes: `VECTOR_LOCAL`, `VECTOR_GLOBAL` o `VECTOR_DYNAMIC`.

- a. En los dos códigos (Listado 1 y Listado 2) se utiliza la función `clock_gettime()` para obtener el tiempo de ejecución del trozo de código que calcula la suma de vectores. En el código se imprime la variable `ncgt`, ¿qué contiene esta variable? ¿qué información devuelve exactamente la función `clock_gettime()`? ¿en qué estructura de datos devuelve `clock_gettime()` la información (indicar el tipo de estructura de datos y describir la estructura de datos)?

RESPUESTA: La variable `ncgt` contiene la suma total de tiempo que ha tardado en sumar las componentes del vector. La función `clock_gettime` almacena el tiempo especificado en `clk_id` y devuelve un 0 si ha tenido éxito y un -1 en caso de error. `Clock_gettime` devuelve un struct de tipo `timespec`, con dos tipos de datos, un `time_t` y un `long`, que almacenan el tiempo en segundos y en nanosegundos respectivamente.

```
struct timespec {
    time_t    tv_sec;          /* seconds */
    long      tv_nsec;        /* nanoseconds */
};
```

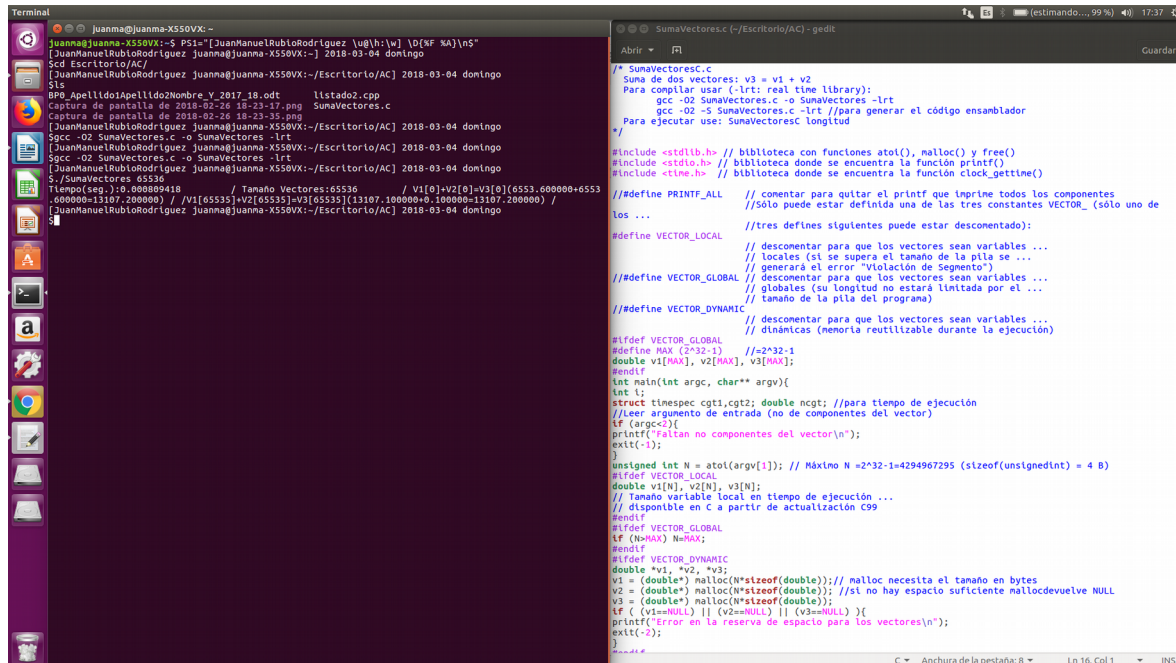
- b. Escribir en el cuaderno de prácticas las diferencias que hay entre el código fuente C y el código fuente C++ para la suma de vectores.

RESPUESTA:

Descripción diferencia	En C	En C++
Mostrar por pantalla	<code>printf</code>	<code>cout</code>
Reserva de memoria	<code>malloc</code>	<code>new</code>
Liberar memoria	<code>free</code>	<code>delete</code>
Declarar variables dentro de un bucle for	No se puede, se debe declarar fuera del bucle	Se puede declarar en la cabecera del for

3. Generar el ejecutable del código fuente C del Listado 1 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). Incorporar volcados de pantalla que demuestren la ejecución correcta en atcgrid o en su PC.

RESPUESTA:



```

Terminal
juanma@juanma-X550VX:~$
juanma@juanma-X550VX:~$ PS1="[\u@h:\w] \D[\XF MA]\n$"
[\u@h:\w] \D[\XF MA]\n$
Sed Escritorio/AC/
[\u@h:\w] \D[\XF MA]\n$
BPO_ApellidoApellidoNombre_Y_2017_18.odt Listado2.cpp
Captura de pantalla de 2018-02-26 18-23-17.png SumaVectores.c
Captura de pantalla de 2018-02-26 18-23-35.png
[\u@h:\w] \D[\XF MA]\n$
Sgcc -O2 SumaVectores.c -o SumaVectores -lrt
[\u@h:\w] \D[\XF MA]\n$
Sgcc -O2 SumaVectores.c -o SumaVectores -lrt
[\u@h:\w] \D[\XF MA]\n$
S./SumaVectores 65536
Tiempo(seg.):0.00000048 / Tamaño Vectores:65536 / V1[0]+V2[0]+V3[0](6553.600000+6553.600000+13107.200000) / V1[65535]+V2[65535]+V3[65535](13107.100000+0.100000+13107.200000)
[\u@h:\w] \D[\XF MA]\n$

SumaVectores.c (-/Escritorio/AC) - gedit
Abrir Guardar
/* SumaVectores.c
 * Suma de dos vectores: v1 + v2
 * Para compilar usar (-lrt: real time library):
 * gcc -O2 SumaVectores.c -o SumaVectores -lrt
 * gcc -O2 -S SumaVectores.c -lrt //para generar el código ensamblador
 * Para ejecutar usar: SumaVectores Longitud
 */
#include <stdio.h> // biblioteca con funciones atoi(), malloc() y free()
#include <stdlib.h> // biblioteca donde se encuentra la función printf()
#include <time.h> // biblioteca donde se encuentra la función clock_gettime()

//define PRINTF_ALL // comentar para quitar el printf que imprime todos los componentes
//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)

#ifdef VECTOR_GLOBAL
#define MAX (2*32-1) //2*32-1
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(unsignedint) = 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) NoMax;
#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 mallocdevuelve 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);
    }
}

```

4. Ejecutar en atcgrid el código generado en el apartado anterior usando el script del Listado 3. Generar el ejecutable usando la opción de optimización -O2 tal y como se indica en el comentario que hay al principio del programa. Ejecutar el código también en su PC para los mismos tamaños. ¿Se obtiene error para alguno de los tamaños? En caso afirmativo, ¿a qué se debe este error? (Incorporar volcados de pantalla)

RESPUESTA:

En la captura aparecen los resultados de los archivos producto de ejecutar el script en atcgrid, tanto los resultados como los errores; y el resultado de ejecutar en el pc local el mismo script. Se ve que para ambos, y a partir del mismo tamaño (el último calculado correctamente es de tamaño 262144), a partir del cual comienza a presentar segmentation fault por intentar acceder a memoria que no le ha sido asignada a la ejecución. El error se produce porque se le asigna menos memoria de la necesaria para almacenar los vectores de mayor tamaño. Abajo incorporo la captura de un terminal local mostrando la salida de la ejecución en atcgrid y la ejecución local.

```

juanma@juanma-X550VX: ~
[JuanManuelRubioRodriguez juanma@juanma-X550VX:~/Escritorio/AC] 2018-03-04 domingo
$cat SumaVectoresC_vdinamicas.o64378
Tiempo(seg.):0.000417734 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](6553.600000+6553.60
0000=13107.200000) / /V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.000856327 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](13107.200000+13107.
200000=26214.400000) / /V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.001641770 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](26214.400000+26214.
400000=52428.800000) / /V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
[JuanManuelRubioRodriguez juanma@juanma-X550VX:~/Escritorio/AC] 2018-03-04 domingo
$cat SumaVectoresC_vdinamicas.e64378
/var/lib/torque/mom_priv/jobs/64378.atcgrid.SC: line 23: 9860 Segmentation fault (core dumped) ./
SumaVectores $N
/var/lib/torque/mom_priv/jobs/64378.atcgrid.SC: line 23: 9863 Segmentation fault (core dumped) ./
SumaVectores $N
/var/lib/torque/mom_priv/jobs/64378.atcgrid.SC: line 23: 9866 Segmentation fault (core dumped) ./
SumaVectores $N
/var/lib/torque/mom_priv/jobs/64378.atcgrid.SC: line 23: 9870 Segmentation fault (core dumped) ./
SumaVectores $N
/var/lib/torque/mom_priv/jobs/64378.atcgrid.SC: line 23: 9878 Segmentation fault (core dumped) ./
SumaVectores $N
/var/lib/torque/mom_priv/jobs/64378.atcgrid.SC: line 23: 9883 Segmentation fault (core dumped) ./
SumaVectores $N
/var/lib/torque/mom_priv/jobs/64378.atcgrid.SC: line 23: 9887 Segmentation fault (core dumped) ./
SumaVectores $N
/var/lib/torque/mom_priv/jobs/64378.atcgrid.SC: line 23: 9890 Segmentation fault (core dumped) ./
SumaVectores $N
[JuanManuelRubioRodriguez juanma@juanma-X550VX:~/Escritorio/AC] 2018-03-04 domingo
$./SumaVectores.sh
Tiempo(seg.):0.000923207 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](6553.600000+6553.60
0000=13107.200000) / /V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.000847834 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](13107.200000+13107.
200000=26214.400000) / /V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.001332388 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](26214.400000+26214.
400000=52428.800000) / /V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
./SumaVectores.sh: línea 23: 13547 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 23: 13549 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 23: 13551 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 23: 13553 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 23: 13555 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 23: 13557 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 23: 13559 Violación de segmento ('core' generado) ./SumaVectores $N
./SumaVectores.sh: línea 23: 13561 Violación de segmento ('core' generado) ./SumaVectores $N
[JuanManuelRubioRodriguez juanma@juanma-X550VX:~/Escritorio/AC] 2018-03-04 domingo
$

```

5. Generar los ejecutables del código fuente C para vectores globales y para dinámicos. Genere el ejecutable usando -O2. Ejecutar los dos códigos en atcgrid usando un script como el del Listado 3 (hay que poner en el script el nombre de los ficheros ejecutables generados en este ejercicio) para el mismo rango de tamaños utilizado en el ejercicio anterior. Ejecutar también los códigos en su PC. ¿Se obtiene error usando vectores globales o dinámicos? ¿A qué cree que es debido? (Incorporar volcados de pantalla)

RESPUESTA:

El resultado de ejecutar el script habiendo descomentado en cada caso en el código la definición para crear vectores globales y dinámicos, es el esperado en principio. No produce ningún error, pues en el primer caso, vector global (captura de la izquierda), la longitud no está limitada por el tamaño de la pila; y en el segundo caso, el dinámico (captura de la derecha), la memoria se reutiliza y tampoco se produce error, en contraposición al ejercicio anterior en el que se superaba el tamaño de la pila al ser local.


```

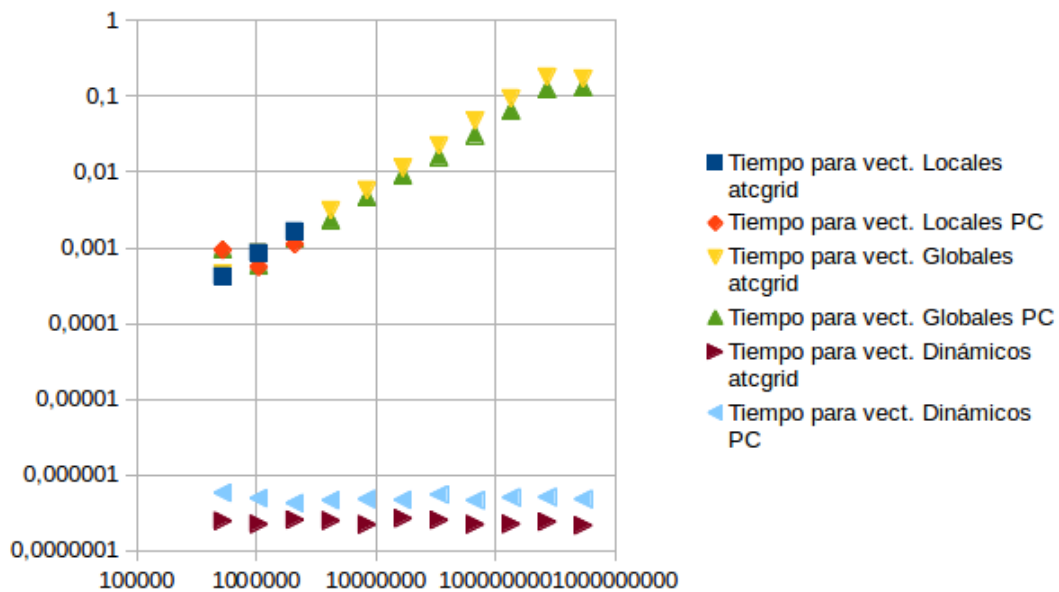
j Juanma@Juanma-X550VX: ~
[JuanManuelRubioRodriguez Juanma@Juanma-X550VX:~/Escritorio/AC] 2018-03-04 domingo
Scat SumaVectoresC_vdinamicas.o64554
Tiempo(seg.):0.000447753 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](6553.600000+6553.60
0000=13107.200000) / /V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.000859327 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](13107.200000+13107.
200000=26214.400000) / /V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.001459693 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](26214.400000+26214.
400000=52428.800000) / /V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tiempo(seg.):0.003133631 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](52428.800000+52428.
800000=104857.600000) / /V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tiempo(seg.):0.005792233 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](104857.600000+10485
7.600000=209715.200000) / /V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo(seg.):0.011364951 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](209715.200000+20971
5.200000=419430.400000) / /V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo(seg.):0.022526055 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](419430.400000+41943
0.400000=838860.800000) / /V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.047782419 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](838860.800000+83886
0.800000=1677721.600000) / /V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000)
/
Tiempo(seg.):0.093911318 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1677721.600000+1677
721.600000=3355443.200000) / /V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.20
0000) /
Tiempo(seg.):0.181188818 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355443.200000+3355
443.200000=6710886.400000) / /V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.40
0000) /
Tiempo(seg.):0.171046827 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355443.200000+3355
443.200000=6710886.400000) / /V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.40
0000) /
[JuanManuelRubioRodriguez Juanma@Juanma-X550VX:~/Escritorio/AC] 2018-03-04 domingo
$

j Juanma@Juanma-X550VX: ~
[JuanManuelRubioRodriguez Juanma@Juanma-X550VX:~/Escritorio/AC] 2018-03-04 domingo
Scat SumaVectoresC_vdinamicas.o64476
Tiempo(seg.):0.000418963 / Tamaño Vectores:65536 / V1[0]+V2[0]=V3[0](6553.600000+6553.60
0000=13107.200000) / /V1[65535]+V2[65535]=V3[65535](13107.100000+0.100000=13107.200000) /
Tiempo(seg.):0.000563636 / Tamaño Vectores:131072 / V1[0]+V2[0]=V3[0](13107.200000+13107.
200000=26214.400000) / /V1[131071]+V2[131071]=V3[131071](26214.300000+0.100000=26214.400000) /
Tiempo(seg.):0.001718040 / Tamaño Vectores:262144 / V1[0]+V2[0]=V3[0](26214.400000+26214.
400000=52428.800000) / /V1[262143]+V2[262143]=V3[262143](52428.700000+0.100000=52428.800000) /
Tiempo(seg.):0.002610571 / Tamaño Vectores:524288 / V1[0]+V2[0]=V3[0](52428.800000+52428.
800000=104857.600000) / /V1[524287]+V2[524287]=V3[524287](104857.500000+0.100000=104857.600000) /
Tiempo(seg.):0.006074798 / Tamaño Vectores:1048576 / V1[0]+V2[0]=V3[0](104857.600000+10485
7.600000=209715.200000) / /V1[1048575]+V2[1048575]=V3[1048575](209715.100000+0.100000=209715.200000) /
Tiempo(seg.):0.012573697 / Tamaño Vectores:2097152 / V1[0]+V2[0]=V3[0](209715.200000+20971
5.200000=419430.400000) / /V1[2097151]+V2[2097151]=V3[2097151](419430.300000+0.100000=419430.400000) /
Tiempo(seg.):0.023907014 / Tamaño Vectores:4194304 / V1[0]+V2[0]=V3[0](419430.400000+41943
0.400000=838860.800000) / /V1[4194303]+V2[4194303]=V3[4194303](838860.700000+0.100000=838860.800000) /
Tiempo(seg.):0.045317516 / Tamaño Vectores:8388608 / V1[0]+V2[0]=V3[0](838860.800000+83886
0.800000=1677721.600000) / /V1[8388607]+V2[8388607]=V3[8388607](1677721.500000+0.100000=1677721.600000)
/
Tiempo(seg.):0.094472452 / Tamaño Vectores:16777216 / V1[0]+V2[0]=V3[0](1677721.600000+1677
721.600000=3355443.200000) / /V1[16777215]+V2[16777215]=V3[16777215](3355443.100000+0.100000=3355443.20
0000) /
Tiempo(seg.):0.176863702 / Tamaño Vectores:33554432 / V1[0]+V2[0]=V3[0](3355443.200000+3355
443.200000=6710886.400000) / /V1[33554431]+V2[33554431]=V3[33554431](6710886.300000+0.100000=6710886.40
0000) /
Tiempo(seg.):0.356072631 / Tamaño Vectores:67108864 / V1[0]+V2[0]=V3[0](6710886.400000+6710
886.400000=13421772.800000) / /V1[67108863]+V2[67108863]=V3[67108863](13421772.700000+0.100000=13421772
.800000) /
[JuanManuelRubioRodriguez Juanma@Juanma-X550VX:~/Escritorio/AC] 2018-03-04 domingo
$

```

6. Rellenar una tabla como la Tabla 1 para atcgrid y otra para su PC con los tiempos de ejecución obtenidos en los ejercicios anteriores para el trozo de código que realiza la suma de vectores. En la columna “Bytes de un vector” hay que poner el total de bytes reservado para un vector. Ayudándose de una hoja de cálculo represente 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 (los valores de la segunda columna de la tabla, que están en escala logarítmica, deben estar en el eje x). Utilice escala logarítmica en el eje de ordenadas (eje y). ¿Hay diferencias en los tiempos de ejecución?

RESPUESTA: El número de bytes por vector sale de multiplicar el número de componentes del vector por 8 bytes que ocupa cada `double` en linux de 64 bits, que es sobre el que está siendo ejecutado.



Se observan tiempos muy similares en todos los casos, excepto para los vectores dinámicos, donde en `atcgrid` hay una mejora significativa de los tiempos de ejecución.

7. Modificar el código fuente C 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 ($\text{MAX}=2^{32}-1$). Generar el ejecutable usando variables globales. ¿Qué ocurre? ¿A qué es debido? Razone además por qué el máximo número que se puede almacenar en N es $2^{32}-1$.

RESPUESTA:

La última iteración que antes la hacía con el mismo tamaño de vector, ahora que aumentamos el valor de `MAX` a 2^{32} , llegamos al último valor del vector que duplica al anterior. El vector almacena de 0 a $n-1$ posiciones.

```
// ...
#ifdef VECTOR_GLOBAL
#define MAX (2^32-1) // = 2^32 - 1
// #define MAX 33554432
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 los componentes del vector\n");
    }
}
```

Listado 1. Código C que suma dos vectores

```

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

Para compilar usar (-lrt: real time library):
    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()

// #define PRINTF_ALL // comentar para quitar el printf ...
// // que imprime todos los componentes
// // 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)

#ifdef 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
for(i=0; i<N; i++){
    v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1; //los valores dependen de N
}

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
#ifdef PRINTF_ALL
printf("Tiempo(seg.):%11.9f\t / Tamaño Vectores:%u\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]);
#endif

#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;
}

```

Listado 2. Código C++ que suma dos vectores


```

/* SumaVectoresCpp.cpp
   Suma de dos vectores: v3 = v1 + v2

   Para compilar usar (-lrt: real time library):
       g++ -O2 SumaVectoresCpp.cpp -o SumaVectoresCpp -lrt

   Para ejecutar use: SumaVectoresCpp longitud
*/

#include <cstdlib> // biblioteca con atoi()
#include <iostream> // biblioteca donde se encuentra la función cout
using namespace std;
#include <time.h> // biblioteca donde se encuentra la función clock_gettime()

// #define COUT_ALL // comentar para quitar el cout ...
// // que imprime todos los componentes
// // 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){

    struct timespec cgt1, cgt2; // para tiempo de ejecución

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

    unsigned int N = atoi(argv[1]);
    #ifdef VECTOR_LOCAL
        double v1[N], v2[N], v3[N];
    #endif
    #ifdef VECTOR_GLOBAL
        if (N > MAX) N = MAX;
    #endif
    #ifdef VECTOR_DYNAMIC
        double *v1, *v2, *v3;
        v1 = new double [N]; // si no hay espacio suficiente new genera una excepción
        v2 = new double [N];
        v3 = new double [N];
    #endif

    // Inicializar vectores
    for(int i=0; i<N; i++){

```

```

    v1[i] = N*0.1+i*0.1; v2[i] = N*0.1-i*0.1; //los valores dependen de N
}
clock_gettime(CLOCK_REALTIME,&cgt1);
//Calcular suma de vectores
for(int i=0; i<N; i++)
    v3[i] = v1[i] + v2[i];
clock_gettime(CLOCK_REALTIME,&cgt2);
double 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
#ifdef COUT_ALL
cout << "Tiempo(seg.):" << ncgt << "\t/ Tamaño Vectores:" << N << endl;
for(int i=0; i<N; i++)
    cout << "/ v1[" << i << "]+v2[" << i << "]=v3[" << i << "]([" << v1[i] << "+"
<< v2[i] << "="
    << v3[i] << ") /\t" << endl;
cout << "\n" << endl;
#else
    cout << "Tiempo(seg.):" << ncgt << "\t/ Tamaño Vectores:" << N << "\t/
v1[0]+v2[0]=v3[0]("
    << v1[0] << "+" << v2[0] << "=" << v3[0] << ") / / v1[" << N-1 << "]+v2["
<< N-1 << "]=v3["
    << N-1 << "]([" << v1[N-1] << "+" << v2[N-1] << "=" << v3[N-1] << ")/\n" <<
endl;
#endif

#ifdef VECTOR_DYNAMIC
delete [] v1; // libera el espacio reservado para v1
delete [] v2; // libera el espacio reservado para v2
delete [] v3; // libera el espacio reservado para v3
#endif
return 0;
}

```

Listado 3. Script para la suma de vectores (SumaVectores.sh). Se supone en el script que el fichero a ejecutar se llama SumaVectorC y que se encuentra en el directorio en el que se ha ejecutado qsub.

```

#!/bin/bash
#Se asigna al trabajo el nombre SumaVectoresC_vlocales
#PBS -N SumaVectoresC_vlocales
#Se asigna al trabajo la cola ac
#PBS -q ac
#Se imprime información del trabajo usando variables de entorno de PBS
echo "Id. usuario del trabajo: $PBS_O_LOGNAME"
echo "Id. del trabajo: $PBS_JOBID"
echo "Nombre del trabajo especificado por usuario: $PBS_JOBNAME"
echo "Nodo que ejecuta qsub: $PBS_O_HOST"
echo "Directorio en el que se ha ejecutado qsub: $PBS_O_WORKDIR"
echo "Cola: $PBS_QUEUE"
echo "Nodos asignados al trabajo:"
cat $PBS_NODEFILE
#Se ejecuta SumaVectorC, que está en el directorio en el que se ha ejecutado qsub,
#para N potencia de 2 desde 2^16 a 2^26
for ((N=65536;N<67108865;N=N*2))

```

```
do
    $PBS_O_WORKDIR/SumaVectoresC $N
done
```