**Arquitecturas Virtuales**

**Práctica 7.- Prestaciones de la virtualización**

Noviembre, 2022

**RESULTADOS DE LOS EXPERIMENTOS (Ver el guion de la práctica)**

**Máquina Host en las que se realiza los experimentos:**

|  |  |
| --- | --- |
| Procesador y número de procesadores (/proc/cpuinfo) | Intel Core i5-7300HQ – 4 cores |
| RAM disponible (free) | 14679 MB |
| Kernel (uname -a) | Linux 6.6.1-arch1-1 |
| Distribución Linux (cat /etc/\*release\*) | ArcoLinux |
| Versión QEMU (qemu-system-x86\_64 --version) | 8.1.2 |

**Configuración de la(s) máquina(s) virtual(es):**

|  |  |
| --- | --- |
| Procesador y número de procesadores (/proc/cpuinfo) | Intel Core i5-7300H – 1 core |
| RAM disponible (free) | 824 MB |
| Kernel (uname -a) | Linux 3.2.0-126-generic |
| Distribución Linux (cat /etc/\*release\*) | Ubuntu 12.04 |

Completa la Tabla 1, con los resultados observados. Para una mayor precisión, repite los experimentos varias veces, y promedia los resultados. Descarta los datos de la primera ejecución de cada prueba y quédate con aquéllos que se hayan realizado con una frecuencia de CPU similar.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Microbenchmark | Máquina HOST | | Máquina virtual 1  *KVM (Trap & Emulate)* | | Máquina virtual 2  *TCG (Traducción)* | |
| elapsed time | # instr. | elapsed time | # instr. | elapsed time | # instr. |
| bc π | 5,14 | 35.219.233.350 | 7,81 | 40.564.463.158 | 38,839 | 387.548.938.832 |
| forkwait | 7,22 | 56.344.859 | 3,44 | 5.845.775.360 | 72,499 | 439.865.620.525 |
| ddcopy | 4,86 | 15.283.974 | 3,16 | 1.414.279.387 | 1,872 | 11.177.447.554 |
| Tabla 1 | | | | | | |

Podemos modelar el *overhead* de una prueba, como el trabajo extra realizado lanzan el comando a través de ssh, más el “ruido de fondo” de ejecución del hipervisor durante el intervalo de tiempo que dura la monitorización:

*Time overhead = τ ,*

donde τ representa el tiempo extra introducido por el lanzamiento del comando bajo test (principalmente debido a la ejecución remota con ssh),

*Instruction overhead =* α *+* β *t,*

donde t es la duración de la prueba, α representa el número de instrucciones extra introducidas por el lanzamiento del comando y β representa el número de instrucciones por unidad de tiempo que el hipervisor ejecuta en reposo (“ruido de fondo”).

Estima los parámetros de overhead de las ejecuciones[[1]](#footnote-1), mediante diferentes ejecuciones del comando sleep t , con diferentes valores de t. Completa previamente para ello la Tabla 2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | HOST | | Máquina virtual 1  *KVM (Trap & Emulate)* | | Máquina virtual 2  *TCG (Traducción)* | |
| elapsed time | # instr. | extra elapsed time (diferencia con host) | # instr. extra (diferencia con host) | extra elapsed time (diferencia con host) | # instr. extra (diferencia con host) |
| sleep 1 | 1 | 6.034.133 | 0,07 | 106.416.984 | 0,36 | 2.231.874.802 |
| sleep 5 | 5 | 6.049.477 | 0,07 | 279.843.871 | 0,36 | 2.402.543.102 |
| sleep 10 | 10 | 6.049.930 | 0,07 | 493.315.007 | 0,36 | 2.767.222.662 |
| Tabla 2 | | | | | | |

Anota la estimación de los parámetros de *overhead*, para cada máquina virtual

|  |  |  |  |
| --- | --- | --- | --- |
| Máquina virtual 1 (KVM) | **τ = 0,07** | **α = 63983400** | **β = 42976600** |
| Máquina virtual 2 (TCG) | **τ = 0,36** | **α = 2147030000** | **β = 60034400** |

Introduce en la Tabla 3 la corrección de los parámetros de *overhead* estimados en las medidas de las máquinas virtuales de la Tabla 1:

*t’ = t - τ*

*#instr.’ = #inst - (*α + β *t)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Microbenchmark | Máquina virtual 1  *KVM (Trap & Emulate)* | | Máquina virtual 2  *TCG (Traducción)* | |
| elapsed time  (SIN overhead) | # instr. (SIN overhead) | elapsed time  (SIN overhead) | # instr. (SIN overhead) |
| bc π | 7,74 | 40.164.832.512 | 38,479 | 383.070.232.770 |
| forkwait | 3,37 | 5.633.952.456 | 72,139 | 433.366.156.559 |
| ddcopy | 3,09 | 1.214.489.931 | 1,512 | 8.918.033.157 |
| Tabla 3 | | | | |

A partir de las observaciones determina ahora los parámetros de la ecuación

:

Resultados para KVM:

|  |  |  |  |
| --- | --- | --- | --- |
| Benchmark | Slowdown | *fp* | *Ne* |
| bc π | 1,506 | 0,7827 | 1,6463 |
| forkwait | 0,4668 | -0,0054 | 99,4573 |
| ddcopy | 0.6358 | -0,0047 | 79,0975 |

Resultados para TCG:

|  |  |  |  |
| --- | --- | --- | --- |
| Benchmark | Slowdown | *fp* | *Ne* |
| bc π | 7,4862 | 17,3629 | 0,3964 |
| forkwait | 9,9916 | 0,0012 | 7700,3 |
| ddcopy | 0,3111 | -0,0012 | 582,8003 |

**Interpretación de los resultados y Observaciones:**

Explica cualquier suposición, hipótesis o metodología que hayas considerado aparte de lo comentado en el guión.

Interpreta los resultados y anota cualquier particularidad que observes en los mismos, resultados paradójicos, explicaciones, etc. **Puedes añadir páginas adicionales para ello.**

* En la ejecución de ddcopy con KVM se obtienen resultados muy dispersos, desde 0.7-0.8 segundos a 45 segundos. Para obtener el resultado, se ha realizado una media de 7 ejecuciones diferentes.
* Podemos ver algunos resultados curiosos, como lo son las ejecuciones de ddcopy y forkwait con KVM, que se obtienen mejores tiempos en comparación con la ejecución en la máquina host.
* A la hora de ejecutar ddcopy con TCG se puede observar el mismo caso que con KVM, pero, en este caso, se puede decir que si se realizan ejecuciones muy seguidas, el tiempo de ejecución aumenta notablemente, pero si se deja un espacio de unos segundos, el tiempo de ejecución se mantiene en el establecido en la tabla.
* Otra cosa curiosa que se observa es el tiempo de ejecución de ddcopy en TCG, ya que es el que mejor resultado obtiene (teniendo en cuenta lo anterior). Es interesante también ya que para el resto de tests arroja resultados mucho peores que el resto.
* En cuanto a los resultados, podemos ver un par de casos curiosos, utilizando KVM obtenemos peor resultado que en la máquina host para el cálculo de pi, el cual es un programa de cálculo con pocas instrucciones privilegiadas, pero en el caso de forkwait, que emplea más instrucciones privilegiadas, obtenemos mejor resultado que en la máquina host.
* Para TCG sí que observamos el overhead que conlleva la traducción binaria ya que obtenemos resultados mucho peores que en los casos anteriores, exceptuando la ejecución de ddcopy.

1. Para realizar un ajuste lineal puedes utilizar WolframAlfpha (http://www.wolframalpha.com/), introduciendo linear fit {{x1,y1}, {x2,y2}, {x3,y3}, ...}}. [↑](#footnote-ref-1)