INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TEGNOLOGÍAS AVANZADAS

SISTEMAS OPERATIVOS EN TIEMPO REAL

“**Implementación de RT a una Raspberry 3b”**

Presentan los alumnos

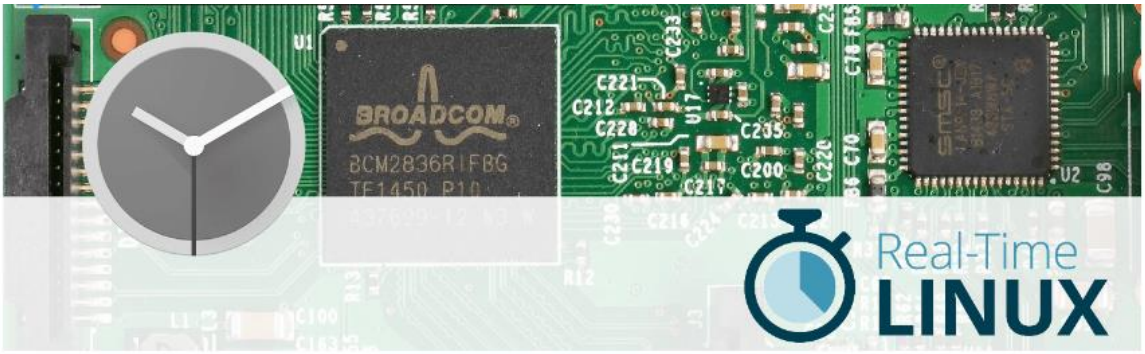
Figueroa Sánchez Eliot Noe

Guerrero Olvera Benjamín

Profesor

Maza Casas Lamberto

En la presente practica se verá paso a paso el proceso de implementación del parche Preempt-RT a una raspberry pi 3 modelo b basada en raspbian 4.14.21 así como la comparación de la latencia entre ambas configuraciones.



Hardware y software

- Raspberry pi 3 b
- Raspbian 3
- RT-Pruebas

  El conjunto de RT-Tests básicamente mide el tiempo necesario para responder a una interrupción, y en el programa de ciclo del conjunto, esta interrupción es generada por un temporizador. En un sistema en tiempo real, la latencia de la respuesta de interrupción debe ser predecible, y la distribución y la latencia máxima son muy importantes.
  rt-tests es un conjunto de pruebas que contiene programas para probar varias características de Linux en tiempo real. Lo mantienen Clark Williams y John Kacur. El código fuente está alojado en kernel.org , las versiones están disponibles aquí . Para la comunicación, se utiliza la lista de correo rt-users .

  ```
  sudo apt-get install build-essential libnuma-dev
  ```

  ```
  git clone git://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git
  cd rt-tests
  git checkout stable/v1.0
  make all
  make install
  ```

  https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/rt-tests
- Preempt-RT RPi

- Cyclictest

  Cyclictest mide de forma precisa y repetida la diferencia entre el tiempo de activación previsto de un subproceso y el momento en que realmente se activa para proporcionar estadísticas sobre la latencia del sistema. La prueba original fue escrita por Thomas Gleixner (tglx), pero varias personas han contribuido modificaciones posteriormente. Actualmente, Clark Williams y John Kacur mantienen a cyclictest como parte de las pruebas rt de la serie de pruebas .

Explicación

Cyclictest ejecuta un hilo maestro no en tiempo real (clase de planificación SCHED_OTHER) que inicia un número definido de hilos de medición con una prioridad definida en tiempo real (clase de programación SCHED_FIFO). Los hilos de medición se activan periódicamente con un intervalo definido por un temporizador que expira (alarma cíclica). Posteriormente, la diferencia entre el tiempo de activación programado y el efectivo se calcula y se transfiere al hilo maestro a través de la memoria compartida. El hilo maestro rastrea los valores de latencia e imprime el mínimo, el máximo y el promedio de la latencia una vez que se completa el número de iteraciones especificadas

Instalando Raspbian

Este recurso explica cómo instalar una imagen del sistema operativo Raspberry Pi en una tarjeta SD. Necesitará otra computadora con un lector de tarjetas SD para instalar la imagen.

Recomendamos a la mayoría de los usuarios que descarguen NOOBS , que está diseñado para ser muy fácil de usar. Sin embargo, los usuarios más avanzados que deseen instalar una imagen particular deberían usar esta guía.

https://www.raspberrypi.org/documentation/installation/installing-images/

Parchar el kernel

Preempt-RT es un parche popular para el kernel de Linux para transformar Linux en un sistema operativo en tiempo real. Integré el kernel Raspian estándar con el parche Preempt-RT y lo compilé en forma cruzada en mi computadora host, que ejecuta Ubuntu 16.04 LTS.

https://lemariva.com/blog/2018/02/raspberry-pi-rt-preempt-tutorial-for-kernel-4-14-y

Suite RT-Tests

Compilar herramientas

Comencemos con la suite RT-Tests. Para incluir el programa de prueba de ciclo en el conjunto de RT-Tests, debe clonar y compilar el repositorio de pruebas rt . El paquete necesita el paquete build-essential para compilarse en Raspbian / Debian.

```
sudo apt-get install -y build-essential # if you are using the Raspbian lite, # you need to install git: sudo apt-get install -y git git clone git://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git cd rt-tests git checkout stable/v1.0 make all -j4 sudo make install
```

Debe verificar la rama stable/v1.0 para construir el proyecto. Si solo desea compilar la prueba cíclica, simplemente reemplace make all -j4 con make cyclictest .

Ejecutar pruebas

Mucha gente está usando el mklatencyplot script bash de OSADL para medir la latencia y generar algunas parcelas. El script necesita la biblioteca gnuplot , que necesita alrededor de 139 bibliotecas y 331 MB de espacio adicional en el disco. Estoy usando Raspbian-Lite porque realmente quiero una imagen pequeña. Luego, tomé solo las primeras líneas del código para ejecutar la prueba y para las tramas, utilicé algo de Python. La prueba cíclica se ejecuta con los siguientes parámetros:

```
sudo cyclictest -l50000000 -m -S -p90 -i200 -h400 -q > output.txt
```

- `-l50000000` : iteraciones de 50M (¡aproximadamente 2,5 horas!);
- `-m` : bloquear las asignaciones de memoria actuales y futuras (¿evitar que se pagine?)
- `-S` : Prueba SMP estándar: opciones -a -t -n y la misma prioridad de todos los hilos (Raspberry Pi tiene 4 núcleos y luego 4 hilos)
- `-p90` : prioridad del hilo de prio más alto establecido en 90 (para los 4 hilos, luego: 90 89 88 87)
- `-i200` : intervalo para el primer hilo (en nosotros).
- `-h1000` : histograma de volcado para la latencia máxima (hasta 1000us).
- `-q` : imprime un resumen solo al salir.

Una vez que finaliza la prueba, obtienes el archivo output.txt con los resultados. Escribiendo lo siguiente:

```
grep -v -e "^#" -e "^$" output.txt | tr " " "," | tr "\t" "," >histogram.csv sed -i '1s/^/time,core1,core2,core3,core4\n /' histogram.csv
```

usted agarra las líneas de datos, elimina las líneas vacías y crea un archivo de valores separados por comas (.csv).

Esta información se puede usar con Python para hacer algunos análisis y gráficos.

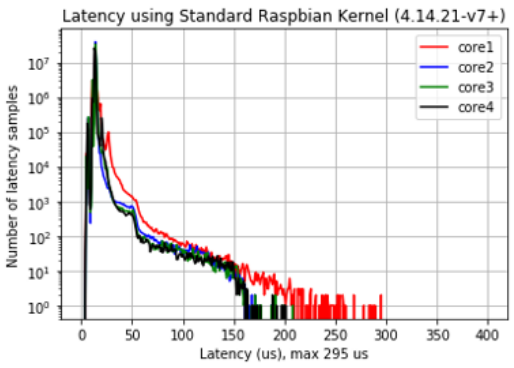
Temperatura

Con una temperatura ambiente cercana a los 20 ° C y utilizando el kernel Standard Raspbian, las temperaturas de la GPU y la CPU se mantuvieron en el rango de 53.0 ° C ± 2.5 ° C. Las temperaturas del kernel Preempt-RT fueron ligeramente superiores en el rango de 55 ° C ± 5.5 ° C. Sin embargo, esto no debería influir en el rendimiento general del sistema.

Estado latente

La figura a muestra la latencia usando el kernel Standard Raspbian, la latencia máxima alcanza 295 us. La figura b muestra la latencia usando el kernel Preempt-RT Raspbian, la máxima latencia se reduce a 147 us (factor ~ 0.5). El núcleo 1 definió este máximo para ambas configuraciones. El máximo es para Std-Raspbian kernel sobre 14 us, mientras que para Preempt-RT es 16 us. Comparando los resultados con

los obtenidos usando kernel 4.9 , el kernel estándar y el kernel Preempt-RT 4.14.21 funcionan mejor.



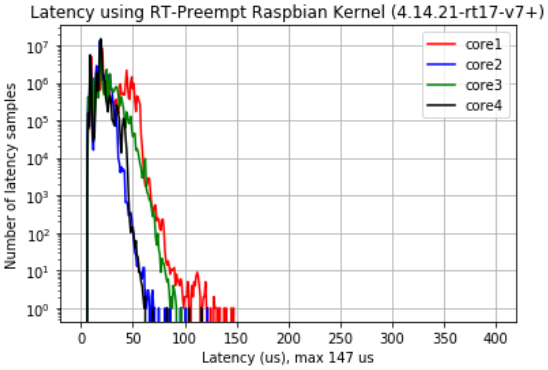


Fig. 2 (a): Latencias usando Std Raspbian Kernel (4.14.21-v7 +). (b): Latencias que utilizan Preempt-RT Raspbian Kernel (4.14.21-rt17-v7 +).

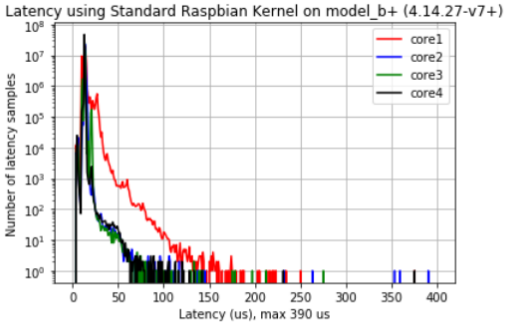


Fig. 1 (a): Latency Using Standard Raspbian Kernel (4.14.27-v7+) on Raspberry Pi 3 B+

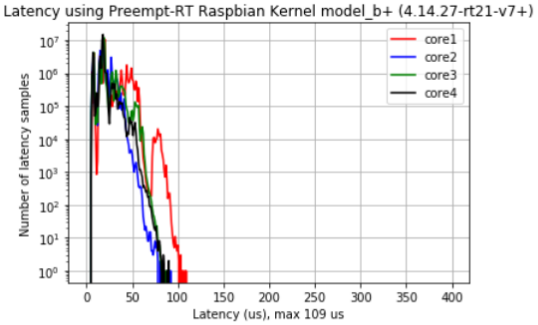


Fig. 1 (b): Latency Using Preempt-RT Patched Raspbian Kernel (4.14.27-rt21-v7+) on Raspberry Pi 3 B+