TOSHIBA

Leading Innovation >>>

Linux Kernel Acceleration for Long-term Testing

Yoshitake Kobayashi
Advanced Software Technology Group
Corporate Software Engineering Center
TOSHIBA CORPORATION

28 Oct, 2010

Outline

- Overview
- How to accelerate Linux kernel
- Development and problem
- Evaluation
- Conclusion

Overview

Today, I'll talking about…

- "What I did" ☺

What I did is…

- Linux kernel acceleration for long-term testing
- …. But this technique may not always right.

Problem and solution

Problem

- Long-term testing takes really long time

  → We want results as fast as possible

→ **Acceleration**

Limitations

A lot of things that cannot be accelerated.

- CPU frequency
- HDD or SSD access speed
- Network link speed
- etc…

Hardware devices are not…

What can be accelerate?

- Clock!

Timer related variables

Timer related variables in Linux kernel

- jiffies
  - A jiffy is the duration of one tick of the system timer interrupt
- xtime
  - Current time and date

Definition of acceleration

- Acceleration = jiffy * (speedup ratio)

note: Speedup ratio = 1,2,3,…

Implementation

- Environment: kernel-2.6.18 (Debian/GNU Linux 4.0).. pretty old

1. Add a parameter to Kconfig
   - Set SPEEDUP_RATIO (range: 1～1000)
2. modified do_timer() a little bit

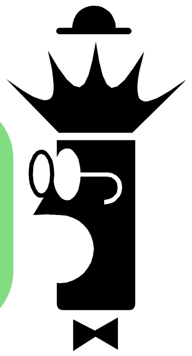
```
void do_timer(....)
{
     jiffies_64 = jiffies_64 + speedup_ratio;
    .....
}
```

   - just add speedup ratio to jiffies
3. Speedup ratio can be controlled via procfs

   ex: echo 100 > /proc/accel

Expected to boot 1000 times faster

Not Work!!



Why not working

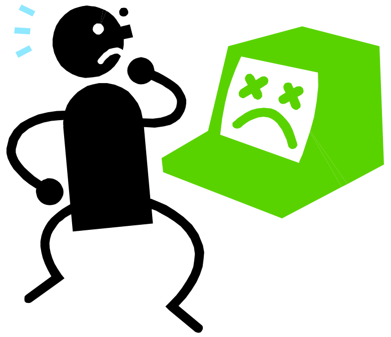
1. Issue
   - Unable to mount the root file system
   - Unable to use physical devices

     (ex. HDD, PS2 mouse)
2. Why?
   - Timeout in linux kernel

     - Device drivers

     - File systems
   - Timeout in user level processes

     (ex. udev)



Issue - Example

PS2 mouse

- Frequently print the following messages
  - Mar 4 00:18:13 accel kernel: psmouse.c: Wheel Mouse at isa0060/serio1/input0 lost synchronization, throwing 1 bytes away.
  - The reported “psmouse.c” is actually ”psmouse-base.c”

Keyboard

- keyboard input speed： really easy to input same characters
- Serial consoles works fine

Screensaver

- Blackout in a blink of an eye

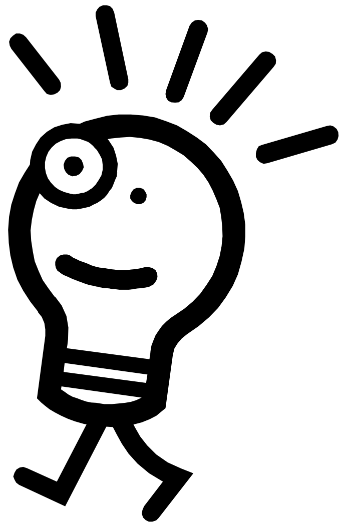
Counterapproach for timeout

1. What?
   - all timeout values in kernel code

     (most of them can be find “grep jiffies”)
2. How?
   - adjust the timeout value by speedup ratio

     ex. timeout * speedup_ratio

Gnome desktop environment works!



Evaluation

- Test cases
  - use gettimeofday() to check time passing
  - use times() to check time passing
  - check system state from syslog, messages and vmstat
  - all test cases runs more than 10 years
    (

- Results
  - no problems with gettimeofday()
  - times() get overflowed
    (same as explained in manual pages)
  - Cannot find any error from syslog, messages and vmstat
  - Correctly power off via ACPI after 10 years

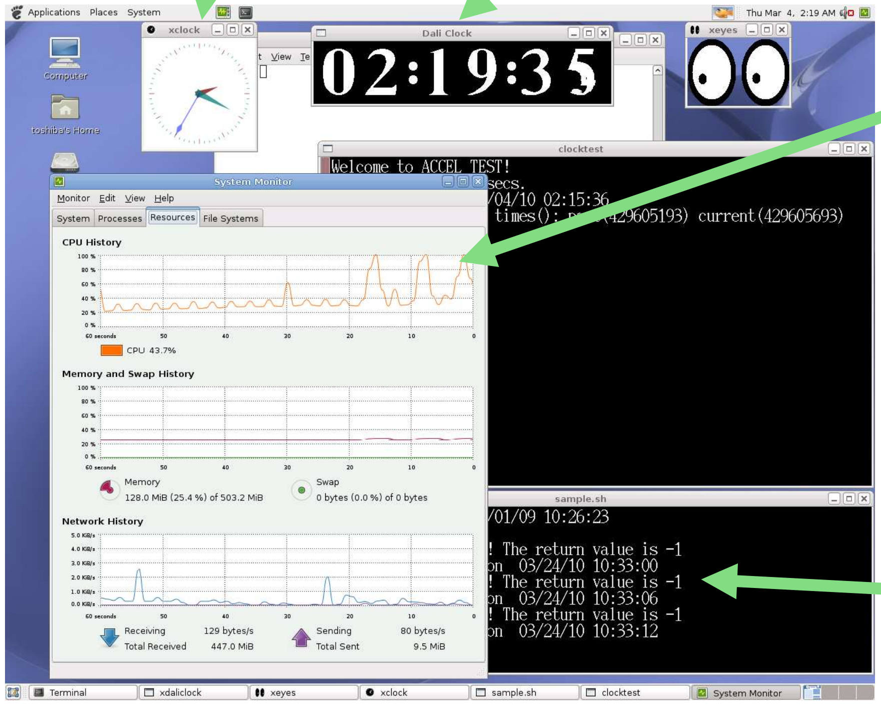
Screenshot

The second hand of xclock skipping a lot

Xdaliclock works as a stopwatch

About 40 times faster get 100% CPU usage

returned incorrect value after about 450 days



clocktest

```
Welcome to ACCEL TEST!
secs.
/04/10 02:15:36
 times(): p[illegible](429605193) current(429605693)
```

sample.sh

```
/01/09 10:26:23

! The return value is -1
on  03/24/10 10:33:00
! The return value is -1
on  03/24/10 10:33:06
! The return value is -1
on  03/24/10 10:33:12
```

TOSHIBA
Leading Innovation >>>

Conclusion

Linux kernel Acceleration

- Not exactly same as hardware acceleration
  - there are several limitations because it isn’t physical acceleration
  - time acceleration only (but it works!)
- Some software appear to work faster
- Can check clock_t overflow in about a day
- Linux kernel may work fine after 10 years
- Easy to test for long-term running test case
- Need more ideas

TOSHIBA

Leading Innovation >>>