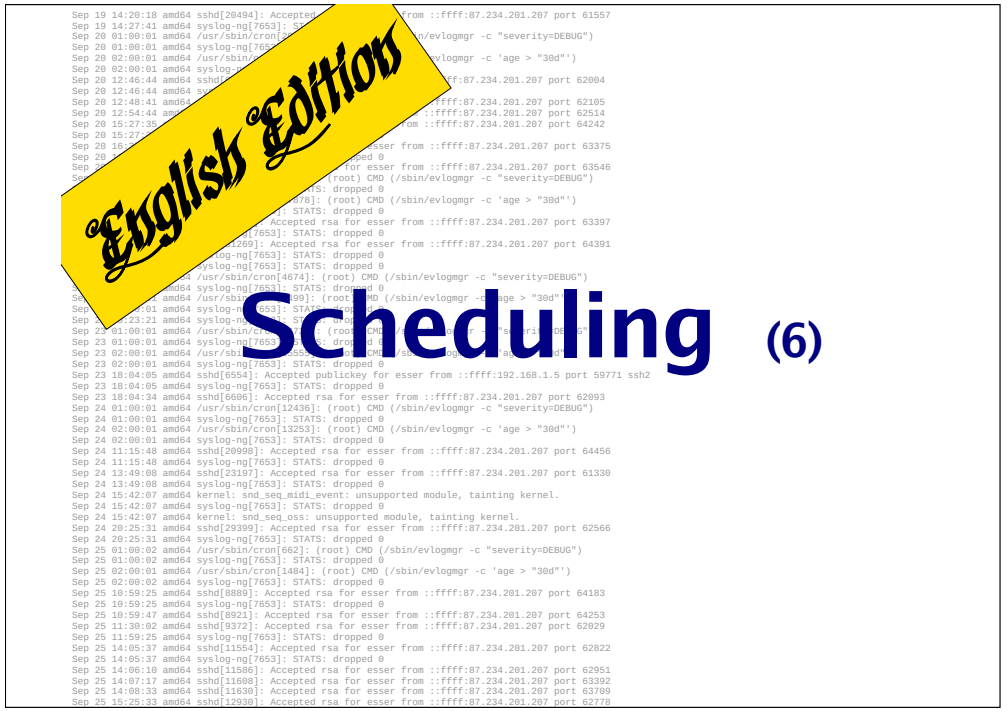
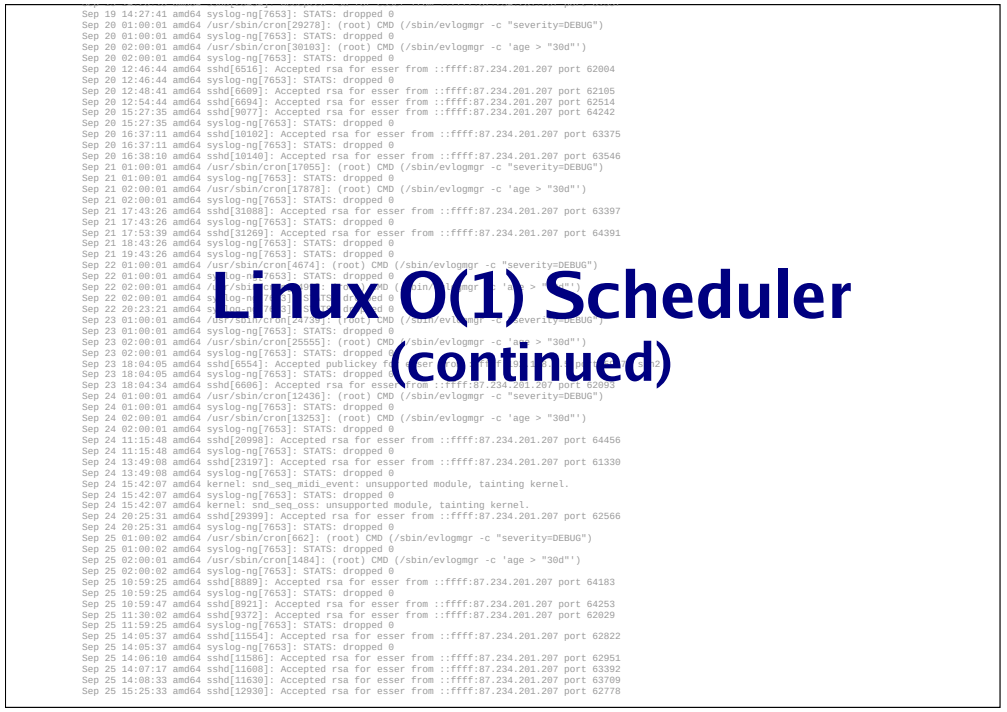


Scheduling (6)

Linux O(1) Scheduler (5)

- separate queues for each CPU
- 140 priority levels
  small value = high priority:
  - 1-100: realtime processes (MAX_RT_PRIO=100)
  - 101-140: normal processes (MAX_PRIO=140)
- normal tasks
  - have nice value n ($-19 \le n \le 20$),
  - Prio = MAX_RT_PRIO + n + 20,
  - receive time quantum



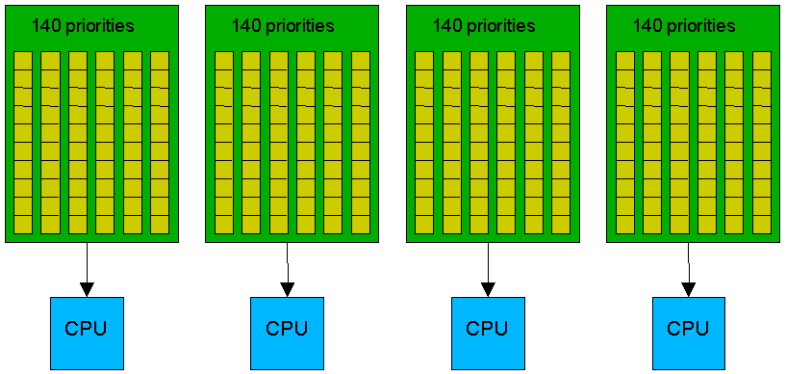
Linux O(1) Scheduler (continued)

Linux O(1) Scheduler (6)

- realtime tasks
  - static priority
  - two classes:
    - FIFO (without preemption) and
    - Round Robin (with time quanta)
- Interactivity estimator:
  checks whether a process is interactive – if so, then it will run with higher priority
  (only normal processes, not realtime ones)

Linux O(1) Scheduler (7)

for each CPU and each priority one queue (that is: 140 lists per CPU)!



Picture: Linux Journal, http://www.linuxjournal.com/node/7178/

Linux O(1) Scheduler (8)

Finding the next process is very easy:

- each CPU only has to search its private process list
- Bitmap stores information which (of the 140) queues are empty – a search of the kind „1st bitmap field with value 1“ is quick
- within the found list pick the first process
- search time depends „on 140“, but not on the number of processes -> O(1)

Linux O(1) Scheduler (9)

in addition to the Runqueue there is an extra „Expired Runqueue“

- active process whose time quantum runs out will be preempted and moved to the Expired Queue
- while moving, the scheduler recalculates quantum and priority for this process (i.e. possibly sorts it into a different priority level).
- when the Runqueue is completely emptied, swap Runqueue and Expired Runqueue

Linux O(1) Scheduler (10)

Interactivity estimator

- scheduler tries to find out whether processes are I/O bound or CPU bound
- metric: proportion of compute time and (I/O) waiting time
- scheduler
  - awards I/O bound processes
  - punishes CPU bound processes

up to +/- 5 points for priority calculation

Linux O(1) Scheduler (11)

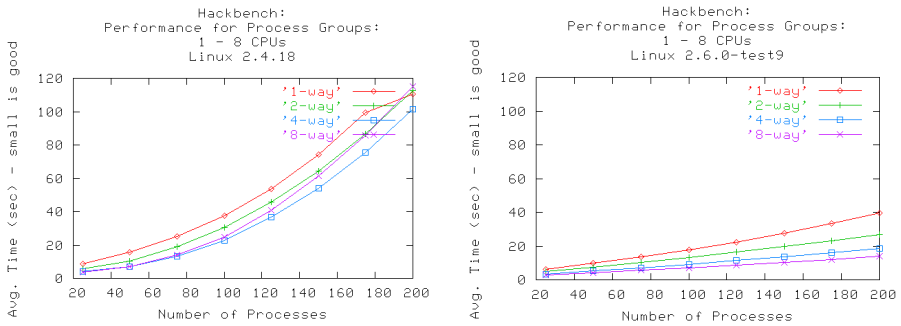
Load Balancer

- actually: try to avoid CPU migration since CPU cache becomes unusable
- on the other hand: CPUs with long idle times are even worse
- every 200 ms some CPU checks whether the load distribution is unbalanced;
  if so, processes are redistributed
- problem: treatment of HyperThreading CPUs with virtual CPUs

Desktop Scheduling

Performance Linux 2.4 / 2.6

Hackbench: up to 200 client / server processes



Picture: http://developer.osdl.org/craiger/hackbench/

„Desktop Scheduling“

Multimedia Scheduler

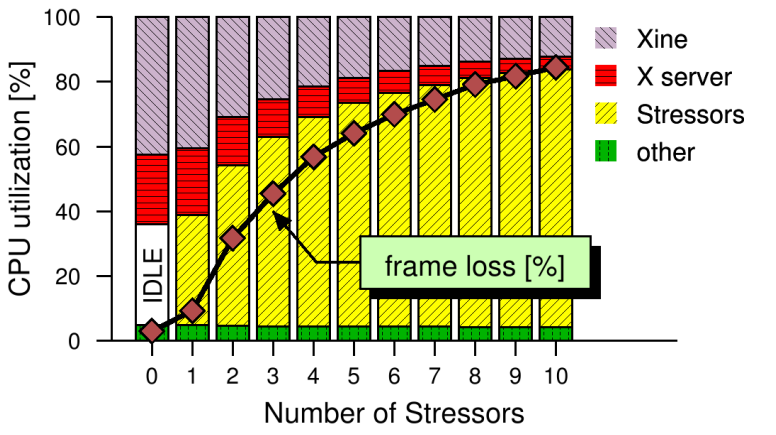
- most schedulers differentiate classically:
  - I/O bound (interactive) vs.
  - CPU bound (non-interactive)
- low CPU utilization -> high priority
- problem: multimedia applications (video, games) require a lot of CPU time, i.e., they are indistinguishable from classical background jobs -> they will not perform well

„Desktop Scheduling“

Performance Test:

- „Stressor applications“: very CPU bound, e.g. Kernel compilation
- how do multimedia applications behave when the number of stressor processes grows?

„Desktop Scheduling“



classical scheduling (picture: [1])

[1] Etsion, Tsafrir, Feitelson, „Desktop Scheduling: How can we know what the user wants?“, 14th ACM Intl. Workshop on Network & Operating Systems Support for Digital Audio & Video (NOSSDAV), pp. 110-115, Jun 2004, www.cse.huji.ac.il/~dants/papers/HuCpri04NOSSDAV.pdf

„Desktop Scheduling“

Old approaches for recognizing interactivity:

- programs make statements about themselves („I am interactive/not interactive“)
- user decides (via start parameters or with a process monitor) which programs should run with a high priority
- program window with focus (active) receives more compute time (MS Windows)

„Desktop Scheduling“

New approach for recognizing interactivity:

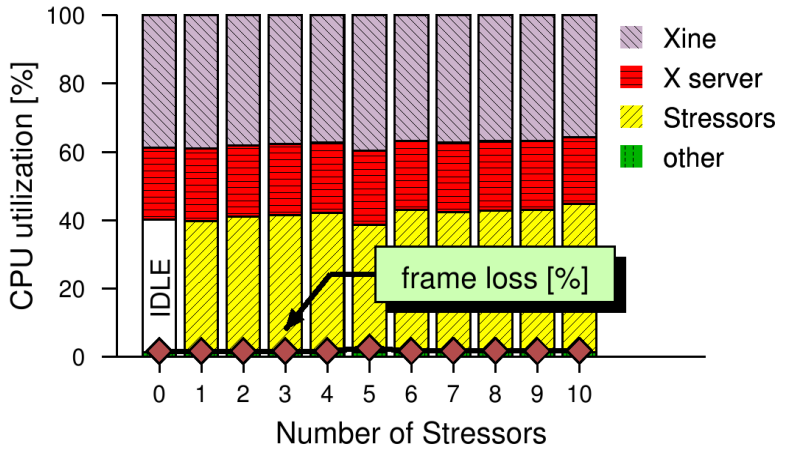
- some I/O devices are very recently used by interactive programs, e.g.
  - keyboard: obviously interactive
  - video card: when a process permanently updates a large screen area, then that is also assumed to be interactive
- statistical data of the I/O devices tell the scheduler which processes are „relevant“ for the user

„Desktop Scheduling“

HuC (human centered) Devices

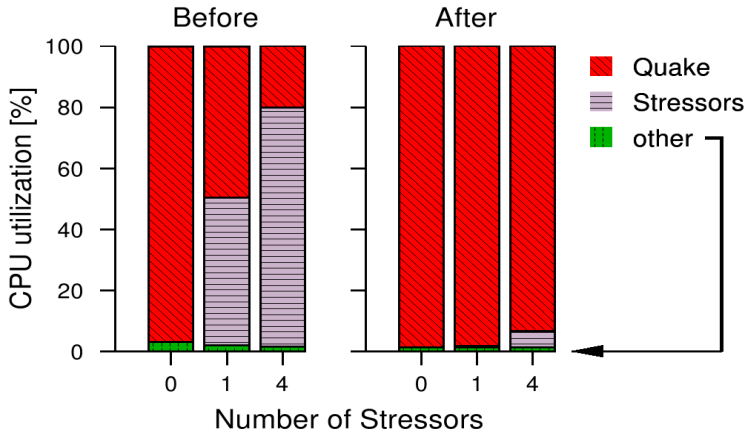
- only selected I/O devices are interesting, e.g. keyboard, mouse, display, joystick, sound card -> **HuC Devices**
- for the screen: patch the X server
  - collect client-specific I/O traffic data
  - forward them 1x/sec to the scheduler
  - measure: what fraction of the screen size has been changed?

„Desktop Scheduling“



HuC scheduler leads to better results Picture: [1]

„Desktop Scheduling“



how much compute time does Quake receive? Bild: [1]
left: normal Linux scheduler, right: HuC scheduler