Switching IO Schedulers on runtime

This manual explains how to change the IO-Scheduler used by your linux kernel without recompiling the kernel and without restart.

First, find out which schedulers are compiled and loaded into the kernel

```
bart:/sys/block/sda/queue # cat scheduler
anticipatory deadline [cfq]
```

cfq is selected here. I built noop as a module here, if I load it:

```
 bart:/sys/block/sda/queue # modprobe noop-iosched
bart:/sys/block/sda/queue # cat scheduler
anticipatory deadline [cfq] noop
```

it shows up as well. Switching is just as simple, you simply echo the

desired scheduler into that file:

```
bart:/sys/block/sda/queue # echo anticipatory > scheduler
bart:/sys/block/sda/queue # cat scheduler
[anticipatory] deadline cfq noop
```

OK, to know which scheduler does what, here an explaination of those schedulers

This is just an executive summary. I don't know the details.

Normally a disk scheduler tries to balance between these goals:

* fairness (let every process have its share of the access to disk)

* performance (try to serve requests close to current disk head position first, because seeking there is fastest)

* realtime (guarantee that a request is serviced in a given time)

A scheduler gets as input the list of sectors that processes have recently requested and which haven't yet given. It knows where the disk head currently is, and whatever other data it "remembers" from the past requests.

A typical scheduler is called cyclic elevator, where disk heads move from beginning of disk to the end of disk in one direction. Assume that you get some requests in a sorted queue, and you're going to satisfy them. So, you'll first filter the list of requests by deciding that you will not satisfy requests for sectors below your current head position. Then you'll simply go to the sector closest to your current position in your sorted list. So, crunch crunch crunch, you move from start of disk to the end of disk. When you reach the highest unsatisfied sector number, your cycle is over and you seek to the lowest unsatisfied sector. Rinse and repeat.

Linux 2.2 at least used the cyclic elevator if I remember right.

So, with these approximate goals in mind, we can look at the different schedulers available.

* deadline scheduler: this is a cyclic elevator but with a twist: requests are given a deadline by which they get served. When a request starts to look like it's going to expire, the kernel will skip intermediate sectors and move to that request, thus giving some realtime behaviour.

* AS scheduler: a cyclic elevator with waiting policy: after you service a request that looks like it might have future requests coming nearby, you pause even if there's more sectors in your work queue. The anticipatory scheduler literally anticipates more requests to follow on this track or very close by. How AS decides whether to anticipate is basically just lot of guesswork based on typical access patterns.

* cfq scheduler: different sort of stab at fairness. It's different from either of these two, and doesn't use cyclic elevator and has realtime guarantees and aggressively avoids starvation. It could be a good scheduler for multiuser systems.

* noop scheduler: just service next request in the queue without any algorithm to prefer this or that request.

You want to use noop scheduler on devices where there are no seeking penalty, such as flash drives. That's why USB stick wants noop. Unfortunately, harddisks are very mechanial beasts and their performance is highly controlled by their seeking abilities. All these schedulers above are really trying to figure out how to extract maximum performance off the harddisk without causing bad behaviour in other cases.

most parts from http://kerneltrap.org/node/3851

current rating:

image:rdf newsfeed / //static.linuxhowtos.org/data/rdf.png (null)
 |
image:rss newsfeed / //static.linuxhowtos.org/data/rss.png (null)
 |
image:Atom newsfeed / //static.linuxhowtos.org/data/atom.png (null)
- Powered by
image:LeopardCMS / //static.linuxhowtos.org/data/leopardcms.png (null)
 - Running on
image:Gentoo / //static.linuxhowtos.org/data/gentoo.png (null)
 -
Copyright 2004-2020 Sascha Nitsch Unternehmensberatung GmbH
image:Valid XHTML1.1 / //static.linuxhowtos.org/data/xhtml.png (null)
 :
image:Valid CSS / //static.linuxhowtos.org/data/css.png (null)
 :
image:buttonmaker / //static.linuxhowtos.org/data/buttonmaker.png (null)
- Level Triple-A Conformance to Web Content Accessibility Guidelines 1.0 -
- Copyright and legal notices -
Time to create this page: ms
<!--
image:system status display / /status/output.jpg (null)
-->