

Dynamic Acceleration Kernel



Iurii Kondrakov @ OSTEP
iurii.kondrakov@senecacollege.ca

Linux Kernel Overview

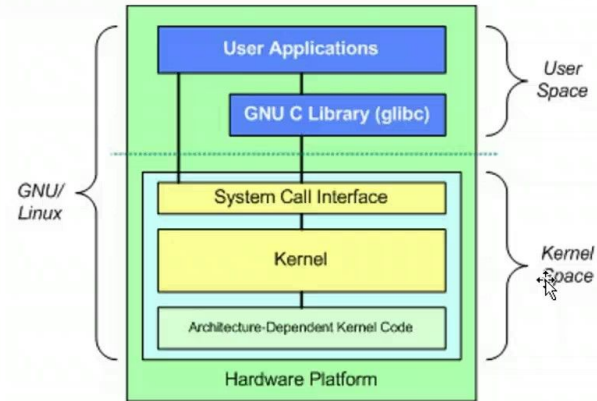
The Linux kernel is the core interface between system processes and the computer's hardware in Linux operating system (OS).

It exists within the OS, managing hardware functions and communication between layers as fast and efficiently as possible.

A Kernel is responsible for:

- Memory management
- Process management
- Device drivers
- System calls and security

Architecture :



https://www.youtube.com/watch?v=OOqgM6ycEgs&ab_channel=AkankshaThorat

Problem Statement

Some tests require a lot of time to complete and find all the edges, bugs and incompleteness in a software.

These tests cannot be skipped or reduced as it directly impacts the quality of a given piece of computer software.

A solution to the problem would be an **ACCELERATION.**



<https://www.sensoft.com/software-testing/software-testing-trends>

Our* Solution

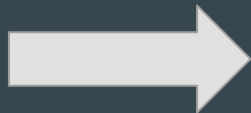
We cannot effortlessly accelerate the hardware such as CPU, Drives and Network devices, but we can speed up the Kernel's clock programmatically.

Definition of acceleration:

Acceleration = $\text{jiffies}/\text{timeout} * (\text{speedup})$,

Where speedup is a ratio of 1, 2, 3, etc

And jiffies is the duration of one tick of the system timer interrupt.



The Dynaccel Linux Kernel that can speed up the long-term testing process through accelerated kernel's flow of time.

* The idea originated from the Toshiba's work on Kernel clock acceleration

Our* Solution cont'd

```
@@ -3587,9 +3588,9 @@ int ata_wait_ready(struct ata_link *link, unsigned long deadline,

    /* choose which 0xff timeout to use, read comment in libata.h */
    if (link->ap->host->flags & ATA_HOST_PARALLEL_SCAN)
-       nodev_deadline = ata_deadline(start, ATA_TMOUT_FF_WAIT_LONG);
+       nodev_deadline = ata_deadline(start, ATA_TMOUT_FF_WAIT_LONG * speedup_ratio);
    else
-       nodev_deadline = ata_deadline(start, ATA_TMOUT_FF_WAIT);
+       nodev_deadline = ata_deadline(start, ATA_TMOUT_FF_WAIT * speedup_ratio);

@@ -599,7 +600,7 @@ static void atkbd_event_work(struct work_struct *work)
    * rescheduling till reconnect completes.
    */
    schedule_delayed_work(&atkbd->event_work,
-       msecs_to_jiffies(100));
+       msecs_to_jiffies(100 * speedup_ratio));
} else {
    if (test_and_clear_bit(ATKBD_LED_EVENT_BIT, &atkbd->event_mask))
        atkbd_set_leds(atkbd);
```

```
[deezir@dynaccel ~]$ sudo sysctl --write kernel.accel=5
kernel.accel = 5
[deezir@dynaccel ~]$ cat /proc/sys/kernel/accel
5
```

The Dynaccel Kernel is based on the upstream CentOS 8s kernel version 4.18.0.

The speedup_ratio (acceleration) can be changed with sysctl.

Example: `sysctl --write kernel.accel=50`.

The current acceleration ratio can be accessed through `cat /proc/sys/kernel/accel`.

* The idea originated from the Toshiba's work on Kernel clock acceleration

Acceleration works

Even inner VMs inherit the accelerated time flow

```
top - 21:10:44 up 15 days, 21:47, 3 users, load average: 5.08, 5.64, 5.57
Tasks: 212 total, 1 running, 211 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.2 sy, 0.0 ni, 27.4 id, 72.1 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7767.2 total, 2003.3 free, 2153.1 used, 3610.8 buff/cache
MiB Swap: 3584.0 total, 3581.1 free, 2.9 used, 5320.8 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3310	qemu	20	0	5970652	1.5g	17744	S	0.4	19.2	340:34.47	qemu-kvm
232044	deezir	20	0	264096	4268	3628	R	0.2	0.1	0:04.22	top
1	root	20	0	241224	14116	9088	S	0.0	0.2	0:10.42	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.31	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-events_highpri
9	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_rude_
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tasks_trace
12	root	20	0	0	0	0	S	0.0	0.0	0:00.18	ksoftirqd/0
13	root	20	0	0	0	0	I	0.0	0.0	0:05.99	rcu_sched
14	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
15	root	rt	0	0	0	0	S	0.0	0.0	0:00.56	watchdog/0
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
17	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
18	root	rt	0	0	0	0	S	0.0	0.0	0:01.68	watchdog/1
19	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/1
20	root	20	0	0	0	0	S	0.0	0.0	0:00.12	ksoftirqd/1
22	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H-events_highpri
23	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2
24	root	rt	0	0	0	0	S	0.0	0.0	0:01.74	watchdog/2
25	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/2
26	root	20	0	0	0	0	S	0.0	0.0	0:00.11	ksoftirqd/2
28	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/2:0H-events_highpri
29	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/3
30	root	rt	0	0	0	0	S	0.0	0.0	0:02.45	watchdog/3

GIF of [top](#) command

Dynacel Kernel SRC.RPMs

The Dynacel Kernel can be wrapped into an RPM package and hosted as a repository for easy distribution and installation.

The `kernel.spec` specifies EPOCH 1 for Dynacel to take precedence over the latest kernel present in DNF.

4.18.0 version overtakes the most recent 5.18.11

```
[deezir@greenland ~]$ sudo dnf update
[sudo] password for deezir:
Last metadata expiration check: 0:08:06 ago on Wed 10 Aug 2022 05:17:00 PM EDT.
Dependencies resolved.

=====
Package                                Architecture      Version                                Repository      Size
=====
Installing:
kernel                                 x86_64            1:4.18.0-394.el8.dynacel              dynacel         8.3 M
kernel-core                           x86_64            1:4.18.0-394.el8.dynacel              dynacel         40 M
kernel-modules                         x86_64            1:4.18.0-394.el8.dynacel              dynacel         32 M
kernel-modules-extra                   x86_64            1:4.18.0-394.el8.dynacel              dynacel         8.9 M
Upgrading:
code                                  x86_64            1.70.1-1660113182.el7                 code            115 M
kernel-headers                         x86_64            1:4.18.0-394.el8.dynacel              dynacel         9.6 M
Removing:
kernel                                 x86_64            5.18.11-100.fc35                       @updates        0
kernel-core                           x86_64            5.18.11-100.fc35                       @updates        92 M
kernel-modules                         x86_64            5.18.11-100.fc35                       @updates        57 M
kernel-modules-extra                   x86_64            5.18.11-100.fc35                       @updates        3.3 M

Transaction Summary
=====
Install 4 Packages
Upgrade 2 Packages
Remove  4 Packages

Total size: 213 M
Total download size: 115 M
Is this ok [y/N]:
```

The Dynacel RPM shows during `dnf update`

Steps to Build the Dynacel RPM package

1. Install RPM Developer tools.
2. Set up the RPM build tree.
3. Copy the *src.rpm to the SRPMS folder and install the RPM.
4. Build the RPM
 - a. The output will be a list of kernel RPMs
5. Create and sign an RPM repository.

```
[deezzir@fedora rpmbuild]$ tree .
```

```
├── BUILD
├── RPMS
├── SOURCES
├── SPECS
└── SRPMS
```

```
5 directories, 0 files
```

RPM build tree

```
[deezzir@dynacel ~]$ cat /etc/yum.repos.d/dynacel.repo
[dynacel]
name=dynacel
baseurl=file:///var/www/html/dynacel
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-dynacel
enabled=1
```

RPM Repository definition

Conclusion

The Dynacel Kernel makes long-term testing easier by accelerating the flow of time.

A programmer can handily deploy the Dynacel Kernel RPM packages within an environment to speed up the testing process.

The project can be maintained and improved with the help of an open-source community.

The sources must be compiled under gcc-8 and g++-8, otherwise compilation will fail.



The Dynacel patches were only applied and tested on kernel version 4.18.0.



It is not a hardware acceleration, so there isn't any performance boost, just accelerated time.

Sources

- A slides by the original authors (Toshiba)
https://elinux.org/images/6/6d/Linux_Kernel_Acceleration_for_Long-term_Testing.pdf
- What is the Linux kernel? Redhat
<https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel>
- Dynaccel-kernel-4.18.0 Documentation
<https://github.com/Seneca-CDOT/dynaccel-kernel-4.18.0/blob/main/README.md>
- CDOT Wiki page on how to create and sign RPM repository
https://wiki.cdot.senecacollege.ca/wiki/Signing_and_Creating_a_Repository_for_RPM_Packages
- Fedora Wiki page on how to create a custom Kernel RPM package
https://www.fedoraproject.org/wiki/Building_a_custom_kernel/Source_RPM