Work Update

CHALOYARD Lucas

April 29, 2022

1 Linux Kernel 5.16

1.1 Performances overview, 5.15 & 5.16

Test	Better	CFS 5.15 sched neowise	CFS 5.15 perf neowise	Nest 5.15 sched neowise	Nest 5.15 perf neowise	CFS 5.16 sched neowise	CFS 5.16 perf neowise	Nest 5.16 sched neowise	Nest 5.16 perf neowise
ffmpeg_configure	L	5.36	1.00	1.22	1.20	0.96	0.95	1.11	1.12
gcc_configure	L	1.39	1.00	1.26	1.23	0.96	0.96	1.05	1.06
gdb_configure	L	1.24	0.99	1.22	1.22	0.95	0.95	1.10	1.10
imagemagick_configure	L	14.92	1.00	1.17	1.14	0.95	0.95	1.05	1.07
linux_configure	L	2.44	1.00	1.06	1.06	0.98	0.98	1.00	1.00
llvm_ninja_configure	L	10.53	1.00	1.07	1.06	0.97	0.97	1.02	1.02
mplayer_configure	L	9.56	1.00	1.22	1.21	0.94	0.94	1.11	1.11
php_configure	L	12.65	1.00	1.14	1.14	0.95	0.95	1.05	1.06

Figure 1: Configure benchmark runned on a neowise machine

Test	Better	CFS 5.9 sched dahu	CFS 5.9 perf dahu	Nest 5.9 sched dahu	Nest 5.9 perf dahu	CFS 5.15 sched dahu	CFS 5.15 perf dahu	Nest 5.15 sched dahu	Nest 5.15 perf dahu	CFS 5.16 sched dahu	CFS 5.16 perf dahu	Nest 5.16 sched dahu	Nest 5.16 perf dahu
erlang_configure	L	16.88	1.20	1.25	1.25	1.01	1.20	1.23	1.26	0.97	1.16	1.16	1.19
ffmpeg_configure	L	7.04	1.27	1.38	1.39	1.01	1.26	1.37	1.37	0.96	1.21	1.25	1.31
gcc_configure	L	1.66	1.18	1.30	1.29	0.92	1.21	1.23	1.31	0.87	1.16	1.15	1.26
gdb_configure	L	1.47	1.18	1.29	1.29	0.92	1.20	1.20	1.31	0.89	1.15	1.09	1.20
imagemagick_configure	L	19.51	1.23	1.34	1.33	0.99	1.22	1.35	1.33	0.93	1.18	1.24	1.27
linux_configure	L	2.89	1.14	1.19	1.20	0.99	1.14	1.18	1.19	0.95	1.11	1.09	1.14
llvm_ninja_configure	L	12.89	1.20	1.25	1.26	0.92	1.20	1.26	1.25	0.90	1.17	1.20	1.22
llvm_unix_configure	L	16.43	1.24	1.31	1.32	0.96	1.24	1.30	1.29	0.94	1.22	1.24	1.27
mplayer_configure	L	12.89	—	1.42	1.44	1.00	1.24	1.44	1.44	0.94	1.18	1.32	1.35
nodejs_configure	L	1.65	1.02	1.02	1.03	1.00	1.02	1.02	1.02	0.99	1.02	0.95	0.99
php_configure	L	17.47	1.26	1.37	1.36	0.98	1.24	1.36	1.36	0.94	1.20	1.26	1.31

Figure 2: Configure benchmark runned on a dahu machine

As we can see on both figures, there is a decrease in performances in almost all benchmarks when we use the Linux Kernel 5.16 (compared to Linux Kernel 5.15 or Linux Kernel 5.9).

When we're using NEST we're reaching a decrease of 21% in the case of a neowise machine, and 12% on a dahu machine.
When we're using regular CFS we're reaching a decrease of 6% in the case of a neowise machine, and 6% too on a dahu machine.

Not only we observe high decreases on some benchmarks but we can also see that in almost all of them, a decrease is present but not always of the same amplitude.

As for the dacapo benchmarks, I have results for a few of them, it seems that there is still a decrease but less

important.

Test	Better	CFS 5.9 sched neowise	CFS 5.9 perf neowise	Nest 5.9 sched neowise	Nest 5.9 perf neowise	CFS 5.15 sched neowise	CFS 5.15 perf neowise	Nest 5.15 sched neowise	Nest 5.15 perf neowise	CFS 5.16 sched neowise	CFS 5.16 perf neowise	Nest 5.16 sched neowise	Nest 5.16 perf neowise
avrora	L	25.42	1.00	1.15	1.16	1.00	1.01	1.15	1.14	1.00	1.00	1.08	1.10
graphchi-eval	L	6.74	1.00	1.09	1.10	1.00	1.01	1.11	1.11	0.99	0.99	1.04	1.05
h2	L	39.59	1.00	1.11	1.09	1.01	1.01	1.10	1.10	1.00	1.01	1.07	1.08
lusearch	L	2.86	1.02	1.19	1.14	1.44	1.44	1.46	1.44	1.45	1.44	1.39	1.38
lusearch-fix	L	2.81	0.99	1.14	1.17	1.42	1.43	1.43	1.42	1.41	1.41	1.37	1.38
pmd	L	8.11	1.00	1.05	1.05	1.05	1.05	1.05	1.05	1.03	1.03	1.02	1.02
sunflow	L	5.73	0.98	1.04	1.05	1.08	1.11	1.08	1.07	1.05	1.09	1.07	1.01
xalan	L	4.20	1.01	1.07	1.05	1.13	1.11	1.11	1.12	1.13	1.08	1.07	1.07
zxing-eval	L	8.12	1.00	1.05	1.05	1.07	1.08	1.08	1.17	1.14	1.13	1.11	1.06

Figure 3: Dacapo benchmarks results on neowise machine

As we can see, when we're using the NEST version there is still a decrease on almost all benchmark, and this decrease can reach 11%.
As for the Linux Kernel using CFS, we cannot conclude anything about decreases, it seems like there is no differences big or frequent enough to say anything about it.

1.2 Trace analysis

1.2.1 CFS call

I've tried to run several times the mplayer benchmark on the neowise machines using a modified kernel printing some information about cpu selection.

```
configure-56800 [014]    52.376734: bprint:               find_idlest_group: [13] Selected group : [1835008;28;0]
configure-56800 [014]    52.376735: bprint:               select_task_rq_fair: [6] Selected CPU from group : 67
configure-56800 [014]    52.376736: bprint:               select_task_rq_fair: [2] Previous CPU : 14 | Selected CPU : 67
configure-57076 [016]    52.523821: bprint:               find_idlest_group: [13] Selected group : [14680064;224;0]
configure-57076 [016]    52.523822: bprint:               select_task_rq_fair: [6] Selected CPU from group : 71
configure-57076 [016]    52.523824: bprint:               select_task_rq_fair: [2] Previous CPU : 16 | Selected CPU : 71
configure-57081 [063]    52.548053: bprint:               find_idlest_group: [13] Selected group : [117440512;1792;0]
configure-57081 [063]    52.548054: bprint:               select_task_rq_fair: [6] Selected CPU from group : 74
configure-57081 [063]    52.548056: bprint:               select_task_rq_fair: [2] Previous CPU : 63 | Selected CPU : 74
configure-57093 [063]    52.574620: bprint:               find_idlest_group: [13] Selected group : [939524096;14336;0]
configure-57093 [063]    52.574620: bprint:               select_task_rq_fair: [6] Selected CPU from group : 77
configure-57093 [063]    52.574622: bprint:               select_task_rq_fair: [2] Previous CPU : 63 | Selected CPU : 77
```

Figure 4: Print displayed when NEST is using CFS, Linux Kernel 5.15

```
configure-56801 [050]    56.103145: bprint:               find_idlest_group: [13] Selected group : [1261007895663743360;0]
configure-56801 [050]    56.103146: bprint:               select_task_rq_fair: [6] Selected CPU from group : 56
configure-56801 [050]    56.103147: bprint:               select_task_rq_fair: [2] Previous CPU : 50 | Selected CPU : 56
configure-57077 [051]    56.259167: bprint:               find_idlest_group: [13] Selected group : [1008806316530994688;0;0]
configure-57077 [051]    56.259168: bprint:               select_task_rq_fair: [6] Selected CPU from group : 58
configure-57077 [051]    56.259171: bprint:               select_task_rq_fair: [2] Previous CPU : 51 | Selected CPU : 58
configure-57094 [058]    56.307310: bprint:               find_idlest_group: [13] Selected group : [8070450532247957504;0;0]
configure-57094 [058]    56.307311: bprint:               select_task_rq_fair: [6] Selected CPU from group : 13
configure-57094 [058]    56.307313: bprint:               select_task_rq_fair: [2] Previous CPU : 58 | Selected CPU : 13
```

Figure 5: Print displayed when NEST is using CFS, Linux Kernel 5.16

Unfortunately, nothing come out of this kind of analysis, there is not one version of the Kernel that is using more CFS than the others. Sometimes the 5.15 will call CFS one or two more times and at other times it will be the 5.16 versions which will do more calls to CFS.

1.2.2 NEST analysis

I also tried to analyze which part of the NEST code is called by each version of the Kernel in order to know if we could a difference in the behaviour of each versions.
In order to do so, I placed one "trace_printk" before each modification of the expand or reserve mask done in the "kernel/sched/fair.c" file.
Each of this printk has been given a number **"[NX]"** with X a number going from 1 to 5, which simply corresponds to the order of apparition in the code.

```
report-mplayer_configure_neowise-3_5.16.0Nest_schedutil_1.txt
[N1] 469
[N2] 15067
[N3] 2016
[N4] 1
[N5] 5
```

```
report-mplayer_configure_neowise-3_5.15.0Nest_schedutil_1.txt
[N1] 417
[N2] 12425
[N3] 4702
[N4] 1
[N5] 5
```

Figure 6: Summary of NEST call done in the 1st run

```
report-mplayer_configure_neowise-5_5.16.0Nest_schedutil_1.txt
[N1] 289
[N2] 9910
[N3] 7329
[N4] 0
[N5] 6
```

```
report-mplayer_configure_neowise-5_5.15.0Nest_schedutil_1.txt
[N1] 467
[N2] 15222
[N3] 1854
[N4] 1
[N5] 4
```

Figure 7: Summary of NEST call done in the 2nd run

```
report-mplayer_configure_neowise-5_5.16.0Nest_schedutil_1.dat.txt
[N1] 337
[N2] 9958
[N3] 7262
[N4] 0
[N5] 7
```

```
report-mplayer_configure_neowise-5_5.15.0Nest_schedutil_1.dat.txt
[N1] 320
[N2] 9903
[N3] 7302
[N4] 0
[N5] 6
```

Figure 8: Summary of NEST call done in the 3rd run

And as we can, there is not particular repetition of behaviour on any of this Linux kernel versions, so it is not really possible to use this simple measurements to detect any modification in the Linux kernel 5.16.

1.2.3 CPU Selection

I tried to look at which CPUs are selected when a call to CFS is made.
I ran the mplayer benchmark 3 times with each Linux Kernel versions, and I ended with the following sequences of selected CPU by CFS when NEST is making a call to CFS (the arrow only indicated that in which order each cpu has been chosen).

Linux 5.15 :

- 1st run : $67 \rightarrow 71 \rightarrow 74 \rightarrow 77$
- 2nd run : $57 \rightarrow 60$
- 3rd run : $34 \rightarrow 84 \rightarrow 40 \rightarrow 44 \rightarrow 47$

Linux 5.16 :

- 1st run : $27 \rightarrow 79 \rightarrow 83 \rightarrow 37 \rightarrow 87$
- 2nd run : $56 \rightarrow 58 \rightarrow 13$
- 3rd run : $27 \rightarrow 31 \rightarrow 81 \rightarrow 36 \rightarrow 40$

Let's take the third run and highlight on the figure below, where each of this cpu is chosen.
Red : 5.15, Blue: 5.16, Red & Blue : Both.
Here is a representation of cluster topology of neowise machine.

0	48	24	72
1	49	25	73
2	50	26	74
3	51	27	75
4	52	28	76
5	53	29	77
6	54	30	78
7	55	31	79
8	56	32	80
9	57	33	81
10	58	34	82
11	59	35	83
12	60	36	84
13	61	37	85
14	62	38	86
15	63	39	87
16	64	40	88
17	65	41	89
18	66	42	90
19	67	43	91
20	68	44	92
21	69	45	93
22	70	46	94
23	71	47	95

As we can see, each version of the kernel will select one cpu per cluster, and if we take a closer look to the cpu sequence, we can see that it is done following something similar to a round-robin order.

Once again, it doesn't seem that a difference exists in this behaviour. Execpt that, on all the runs done, it seems for now that the Linux Kernel 5.16 switch more oftenly between both side of the hyperthreads (so on our figure, it will more oftenly switch the left side and the right side of a square).

2 Best task placement for AMD machines

I've designed artificial benchmark (using the spinning program that I wrote) and also used hackbench benchmark in order to do some experiments about where should the task be placed in order to have the best performances possibles.

Explanation about placement abbrevation (use of 6 cores in total each time).

- split : One core per cluster maximum (6 clusters used).
- mix : Two cores (sharing the same physical core so two hyperthreads) per cluster maximum (3 clusters used).
- same_ccd : One cluster maximum (One cluster has 6 cores).
- same_ccx : Two clusters maximum, 3 cores per cluster (CCX means two cluster that are sharing some ressources, bus, and other things)

Taskset used :

placement name	taskset
split	taskset -c 0,6,12,18,24,32
mix	taskset -c 0,48,6,54,12,60
same_ccd	taskset -c 0,1,2,3,4,5
same_ccx	taskset -c 0,1,2,48,49,50

So for a benchmark only doing computation like the spinning programm, we would have this kind of results.

placement	runtime
split	13.1688
mix	25.5557
same_ccd	13.1532
same_ccx	25.7430

And for the hackbench benchmark, we would have this kind of results.

placement	runtime
split	3.2751
mix	4.0243
same_ccd	2.4443
same_ccx	2.7927

As we can see, depending on the benchmark, each placement is not necessarily good each time. But one placement is the best in both of these cases, the same cdd placement.
Runned using the following command : hackbench -s 512 -l 200 -g 15 -f 50 -P

It could be interesting to try other composition of placements and on other types of workloads.