

Федеральное государственное автономное образовательное учреждение высшего образования  
"Национальный Исследовательский Университет ИТМО"  
Мегафакультет Компьютерных Технологий и Управления  
Факультет Программной Инженерии и Компьютерной Техники



**Лабораторная работа №1**  
по дисциплине  
**'Операционные системы'**

Выполнил Студент группы Р32101  
**Лапин Алексей Александрович**  
Преподаватель:  
**Осипов Святослав Владимирович**

г. Санкт-Петербург  
2023г.

# Содержание

<b>1 CPU Testing</b>	<b>3</b>
1.1 fft . . . . .	3
1.1.1 Команда lscpu . . . . .	3
1.1.2 Команда lstopo . . . . .	5
1.1.3 Команда pidstat . . . . .	5
1.1.4 Команда mpstat . . . . .	8
1.1.5 Команда atop . . . . .	9
1.1.6 Команда stress-ng . . . . .	10
1.1.7 Flame Graph . . . . .	12
1.2 Переходим к активным действиям: nice . . . . .	12
1.2.1 Команда chrt . . . . .	12
1.2.2 Финал . . . . .	14
1.3 cdouble . . . . .	14

# 1 CPU Testing

Выданные параметры: [fft,cdouble]

Параметры процессора используемого в тесте:

Intel Core i5-8259U

Ядер: 4

Потоков: 8

Базовая частота: 2.30 ГГц

Кэш 1-го уровня: 64К (на ядро)

Кэш 2-го уровня: 256К (на ядро)

Кэш 3-го уровня: 6 Мб (всего)

Входе теста используется ОС Linux установленная на виртуальную машину через Parallels.

Ограничение по процессорам: 8

Ограничение по памяти: 2048 MB

## 1.1 fft

### 1.1.1 Команда lscpu

```
Architecture:           x86_64
CPU op-mode(s):        32-bit, 64-bit
Address sizes:         36 bits physical, 48 bits virtual
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:  0-7
Vendor ID:             GenuineIntel
Model name:            Intel(R) Core(TM) i5-8259U CPU @ 2.30GHz
CPU family:            6
Model:                 142
Thread(s) per core:   1
Core(s) per socket:   8
Socket(s):            1
Stepping:              10
BogoMIPS:              4608.00
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mc
                      a cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscal
                      l nx rdtscp lm constant_tsc nopl xtopology nonstop_tsc
                      cpuid tsc_known_freq pni pclmulqdq ssse3 fma cx16 pcid
                      sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer ae
                      s xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowpre
                      fetch invpcid_single pti fsgsbase tsc_adjust bmi1 avx2
                      smep bmi2 invpcid rdseed adx smap clflushopt xsaveopt x
                      savec dtherm arat pln pts
Virtualization features:
Hypervisor vendor:    KVM
Virtualization type:   full
Caches (sum of all):
L1d:                  256 KiB (8 instances)
```

```
L1i:          256 KiB (8 instances)
L2:          2 MiB (8 instances)
L3:          6 MiB (1 instance)

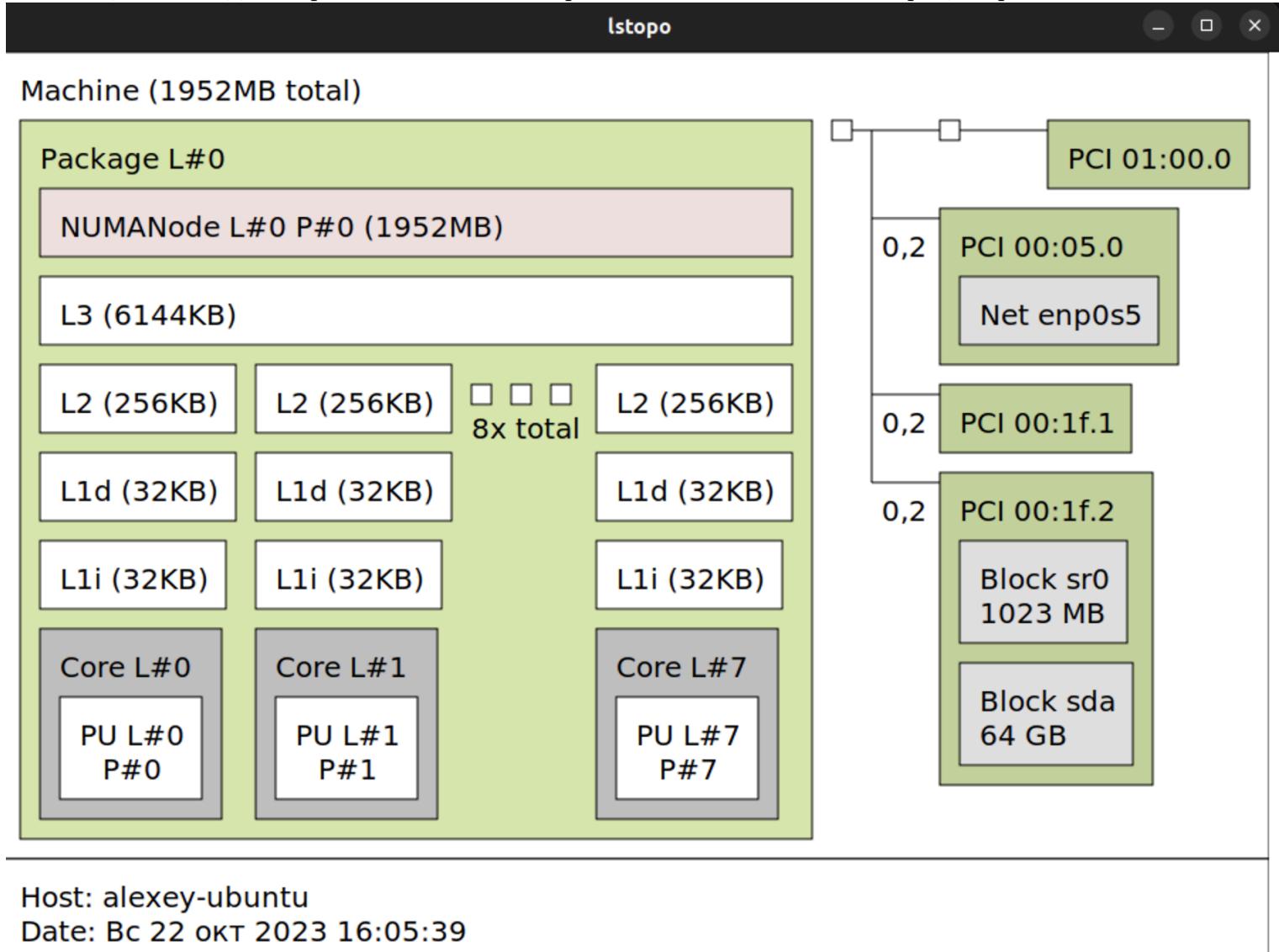
NUMA:
NUMA node(s):      1
NUMA node0 CPU(s): 0-7

Vulnerabilities:
Gather data sampling: Unknown: Dependent on hypervisor status
Itlb multihit:       KVM: Mitigation: VMX unsupported
L1tf:                Mitigation; PTE Inversion
Mds:                 Vulnerable: Clear CPU buffers attempted, no microcode;
                      SMT Host state unknown
Meltdown:            Mitigation; PTI
Mmio stale data:    Vulnerable: Clear CPU buffers attempted, no microcode;
                      SMT Host state unknown
Retbleed:            Vulnerable
Spec rstack overflow: Not affected
Spec store bypass:   Vulnerable
Spectre v1:          Mitigation; usercopy/swapgs barriers and __user pointer
                      sanitization
Spectre v2:          Mitigation; Retpolines, STIBP disabled, RSB filling, PB
                      RSB-eIBRS Not affected
Srbds:               Unknown: Dependent on hypervisor status
Tsx async abort:    Not affected
```

Видим, что Linux распознает наши 4 ядра по 2 потока, как 8 ядер по 1 потоку.

### 1.1.2 Команда lstopo

С помощью команды lstopo мы можем посмотреть на топологию нашего процессора:



### 1.1.3 Команда pidstat

Посмотрим статистику по процессу

Для этого запустим команду: `stress-ng -cpu 1 -cpu-method fft -metrics -timeout 120`

И посмотрим статистику запустив команду: `pidstat -p 58452 1`, где pid мы узнали спомощью команды `top`.

20:45:33	UID	PID	%usr	%system	%guest	%wait	%CPU	CPU	Command
20:45:34	1000	58452	100,00	0,00	0,00	0,00	100,00	3	stress-ng
20:45:35	1000	58452	100,00	0,00	0,00	0,00	100,00	3	stress-ng
20:45:36	1000	58452	100,00	0,00	0,00	0,00	100,00	3	stress-ng
20:45:37	1000	58452	100,00	0,00	0,00	0,00	100,00	3	stress-ng
20:45:38	1000	58452	100,00	0,00	0,00	0,00	100,00	3	stress-ng
20:45:39	1000	58452	99,01	0,00	0,00	0,00	99,01	3	stress-ng
20:45:40	1000	58452	100,00	0,00	0,00	0,00	100,00	3	stress-ng
20:45:41	1000	58452	101,00	0,00	0,00	0,00	101,00	3	stress-ng
20:45:42	1000	58452	100,00	0,00	0,00	0,00	100,00	3	stress-ng

```

20:45:43    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:44    1000    58452  99,01   0,00   0,00   0,00   99,01   3 stress-ng
20:45:45    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:46    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:47    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:48    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:49    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:50    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:51    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:52    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:53    1000    58452  101,00   0,00   0,00   0,00  101,00   3 stress-ng
20:45:54    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:55    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:56    1000    58452  99,01   0,00   0,00   0,00   99,01   3 stress-ng
20:45:57    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:58    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:45:59    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:46:00    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:46:01    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:46:02    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
20:46:03    1000    58452  100,00   0,00   0,00   0,00  100,00   3 stress-ng
^C
Average:   1000    58452  99,97   0,00   0,00   0,00   99,97   - stress-ng

```

Мы видим, что процесс выполнялся на процессоре номер 3 и занимал почти все его время(100%). При этом процесс выполнялся практически все время в режиме *пользователя*, что объяснимо так как fft - вычислительная задача.

Результат *stress-ng*:

```

1 alexey@alexey-ubuntu:~$ stress-ng --cpu 1 --cpu-method fft --metrics --timeout 120
3 stress-ng: info: [58451] setting to a 120 second (2 mins, 0.00 secs) run per stressor
5 stress-ng: info: [58451] dispatching hogs: 1 cpu
7 stress-ng: info: [58451] successful run completed in 120.00s (2 mins, 0.00 secs)
9 stress-ng: info: [58451] stressor bogo ops real time  usr time  sys time  bogo ops/s      bogo ops/s CPU used
     per
11 stress-ng: info: [58451]                               (secs)      (secs)      (secs)      (real time) (usr+sys time) instance
     (%)
13 stress-ng: info: [58451] cpu          102216    120.00    119.97      0.00      851.79      852.01
     99.97

```

Также посмотрим на приоритет и политику планирования процесса:

```

2 alexey@alexey-ubuntu:~$ pidstat -R -p 76043 1
4
Linux 6.2.0-35-generic (alexey-ubuntu) 22.10.2023 _x86_64_ (8 CPU)
6
22:28:20      UID      PID      prio policy Command

```

8	22:28:21	1000	76043	0	NORMAL	stress-ng
10	22:28:22	1000	76043	0	NORMAL	stress-ng
12	22:28:23	1000	76043	0	NORMAL	stress-ng
14	22:28:24	1000	76043	0	NORMAL	stress-ng
16	22:28:25	1000	76043	0	NORMAL	stress-ng
18	22:28:26	1000	76043	0	NORMAL	stress-ng
20	22:28:27	1000	76043	0	NORMAL	stress-ng
22	22:28:28	1000	76043	0	NORMAL	stress-ng
24	22:28:29	1000	76043	0	NORMAL	stress-ng
26	^C					
28	Average:	1000	76043	0	NORMAL	stress-ng

**NORMAL:** *SCHED\_NORMAL* (ранее известный как *SCHED\_OTHER*) — это алгоритм планирования с разделением времени. Используется по умолчанию для пользовательских процессов. Планировщик динамически корректирует приоритет в зависимости от класса планирования. Для  $O(1)$  длительность кванта времени устанавливается на основе статического приоритета: чем выше приоритет, тем большие кванты времени. Для класса *CFS* размер кванта выбирается динамически. Использует класс планирования *CFS*.

#### 1.1.4 Команда mpstat

Запустим команду *mpstat*, чтобы посмотреть загруженность всех процессоров.

Linux 6.2.0-35-generic (alexey-ubuntu) 22.10.2023 _x86_64_ (8 CPU)											
	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
23:00:29	all	13,50	0,00	0,25	0,00	0,00	0,00	0,00	0,00	0,00	86,25
23:00:30	0	1,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	98,00
23:00:30	1	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	99,00
23:00:30	2	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	99,00
23:00:30	3	1,98	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	98,02
23:00:30	4	100,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
23:00:30	5	1,01	0,00	1,01	0,00	0,00	0,00	0,00	0,00	0,00	97,98
23:00:30	6	2,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	98,00
23:00:30	7	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	100,00
23:00:30	all	13,22	0,00	0,50	0,00	0,00	0,12	0,00	0,00	0,00	86,16
23:00:31	0	1,96	0,00	0,98	0,00	0,00	0,98	0,00	0,00	0,00	96,08
23:00:31	1	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	99,00
23:00:31	2	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	100,00
23:00:31	3	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	99,00
23:00:31	4	100,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
23:00:31	5	1,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	98,00
23:00:31	6	0,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	99,00
23:00:31	7	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	99,00
23:00:31	all	13,17	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	86,32
23:00:32	0	0,99	0,00	1,98	0,00	0,00	0,00	0,00	0,00	0,00	97,03
23:00:32	1	0,00	0,00	1,01	0,00	0,00	0,00	0,00	0,00	0,00	98,99
23:00:32	2	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	100,00
23:00:32	3	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	99,00
23:00:32	4	100,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
23:00:32	5	1,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	98,99
23:00:32	6	1,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	98,99
23:00:32	7	1,00	0,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00	98,00

Видим, что *stress-ng* загружает только один процессор на 100% (в данном случае процессор 4). Такое поведение и ожидалось так как мы запустили *stress-ng* с параметром *-cpu 1*. Для следующих запусков определим командой *taskset* процессор на котором будет выполняться *stress-ng*.

alexey@alexey-ubuntu:~\$ taskset -c 0 stress-ng --cpu 1 --cpu-method fft --metrics --timeout 120
stress-ng: info: [83847] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [83847] dispatching hogs: 1 cpu
stress-ng: info: [83847] successful run completed in 120.00s (2 mins, 0.00 secs)
stress-ng: info: [83847] stressor bogo ops real time usr time sys time bogo ops/s bogo ops/s CPU used per
stress-ng: info: [83847] (secs) (secs) (secs) (secs) (real time) (usr+sys time) instance (%)
stress-ng: info: [83847] cpu 110118 120.00 119.96 0.00 917.65 917.96 99.97
alexey@alexey-ubuntu:~\$ taskset -c 0 stress-ng --cpu 1 --cpu-method fft --metrics --timeout 120
stress-ng: info: [84662] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [84662] dispatching hogs: 1 cpu
stress-ng: info: [84662] successful run completed in 120.00s (2 mins, 0.00 secs)
stress-ng: info: [84662] stressor bogo ops real time usr time sys time bogo ops/s bogo ops/s CPU used per
stress-ng: info: [84662] (secs) (secs) (secs) (secs) (real time) (usr+sys time) instance (%)
stress-ng: info: [84662] cpu 111239 120.00 119.96 0.00 926.99 927.30 99.97

Как мы можем заметить, производительность немного подросла, особенно это видно при повторном запуске. Это следует из того, что при использовании одного и того же процессора кэши остаются горячими.

alexey@alexey-ubuntu:~\$ mpstat -P ALL 1												
Linux 6.2.0-35-generic (alexey-ubuntu) 22.10.2023 _x86_64_ (8 CPU)												
	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle	
23:13:41	all	13,84	0,00	0,12	0,00	0,00	0,12	0,00	0,00	0,00	85,91	
23:13:42	0	100,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	
23:13:42	1	2,04	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	97,96	
23:13:42	2	0,98	0,00	0,98	0,00	0,00	0,98	0,00	0,00	0,00	97,06	
23:13:42	3	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	99,00	
23:13:42	4	0,99	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	99,01	
23:13:42	5	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	99,00	
23:13:42	6	2,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	98,00	
23:13:42	7	2,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	98,00	
23:13:42	all	13,25	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	86,25	
23:13:43	0	100,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	
23:13:43	1	2,02	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	97,98	
23:13:43	2	0,99	0,00	0,99	0,00	0,00	0,00	0,00	0,00	0,00	98,02	
23:13:43	3	0,99	0,00	0,99	0,00	0,00	0,00	0,00	0,00	0,00	98,02	
23:13:43	4	1,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	98,99	
23:13:43	5	0,00	0,00	1,98	0,00	0,00	0,00	0,00	0,00	0,00	98,02	
23:13:43	6	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	100,00	
23:13:43	7	1,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	98,99	

### 1.1.5 Команда atop

С помощью команды *atop*, которая является более специфичным для Linux аналогом команды *top*, мы посмотрим общую загруженность системы.

ATOP - alexey-ubuntu 2023/10/22 23:35:09 10s elapsed																			
PRC   sys	0.37s	user	10.56s	#proc	263	#trun	2	#tslpi	480	#tslpu	76	#zombie	0	clones	33	#exit	30		
CPU   sys	4%	user	106%	irq	0%	idle	690%	wait	0%	steal	0%	guest	0%	curf	2.30GHz				
cpu   sys	0%	user	100%	irq	0%	idle	0%	cpu000	w	steal	0%	guest	0%	curf	2.30GHz				
cpu   sys	1%	user	1%	irq	0%	idle	98%	cpu005	w	steal	0%	guest	0%	curf	2.30GHz				
cpu   sys	1%	user	1%	irq	0%	idle	98%	cpu006	w	steal	0%	guest	0%	curf	2.30GHz				
cpu   sys	0%	user	2%	irq	0%	idle	98%	cpu007	w	steal	0%	guest	0%	curf	2.30GHz				
cpu   sys	1%	user	1%	irq	0%	idle	98%	cpu001	w	steal	0%	guest	0%	curf	2.30GHz				
cpu   sys	0%	user	1%	irq	0%	idle	99%	cpu004	w	steal	0%	guest	0%	curf	2.30GHz				
cpu   sys	1%	user	0%	irq	0%	idle	99%	cpu003	w	steal	0%	guest	0%	curf	2.30GHz				
cpu   sys	1%	user	0%	irq	0%	idle	99%	cpu002	w	steal	0%	guest	0%	curf	2.30GHz				
CPL   avg1	0.47	avg5	0.30	avg15	0.27	csw	7563	intr	8585	slab	177.9M	slrec	68.0M	shmem	135.9M	shrss	0.0M	numnode	1
MEM   tot	1.9G	free	145.6M	cache	850.3M	dirty	0.2M	buff	20.9M	slab	177.9M	slrec	68.0M	shmem	135.9M	shrss	0.0M	numnode	1
SWP   tot	5.9G	free	5.8G	swcac	7.2M	vmcom	4.0G	vmlim	6.9G	MBw/s	0.0	MBr/s	0.0	MBw/s	0.0	avio	0.48 ms		
DSK   sda	busy	0%	read	5	write	53	discrd	0	KiB/r	4	KiB/w	5	MBr/s	0.0	tcpie	0	udpie	0	
NET   transport	tcpip	0	tcpo	0	udpi	7	udp0	7	tcpao	0	tcppo	0	tcprs	0	icmpli	0	icmpo	0	
NET   network	ip1	7	ipo	7	ipfrw	0	deliv	7	so	0 Kbps	erri	0	erro	0	drpi	0	drpo	0	
NET   lo	---	pck1	4	pcko	4	sp	0 Mbps	si	0 Kbps	erri	0	erro	0	drpt	0	drpo	0		
NET   empos5	---	pck1	3	pcko	3	sp	0 Mbps	si	0 Kbps	erri	0	erro	0	drpt	0	drpo	0		
PID	SYSCPU	USRCPU	RDELAY	VGROW	RGROW	RUID	EUID	ST	EXC	THR	S	CPUUNR	CPU	CMD	1/4				
87336	0.00s	10.00s	0.00s	0B	0B	alexey	alexey	--	-	1	R	0	100%	stress-ng					
51920	0.05s	0.20s	0.00s	0B	0B	alexey	alexey	--	-	16	S	5	3%	gnome-shell					
83305	0.03s	0.08s	0.00s	0B	0B	alexey	alexey	--	-	12	S	5	1%	gjs					
52849	0.01s	0.09s	0.00s	0B	0B	alexey	alexey	--	-	4	S	7	1%	gnome-terminal					
52245	0.00s	0.08s	0.00s	0B	0B	alexey	alexey	--	-	8	S	7	1%	prlcc					
86739	0.04s	0.01s	0.00s	0B	0B	alexey	alexey	--	-	1	R	1	1%	atop					
614	0.02s	0.03s	0.00s	0B	0B	messageb	messageb	--	-	1	S	4	1%	dbus-daemon					
52261	0.03s	0.00s	0.00s	0B	0B	alexey	alexey	--	-	3	S	0	0%	prldnd					
1727	0.01s	0.01s	0.00s	0B	0B	geoclue	geoclue	--	-	3	S	2	0%	geoclue					
777	0.01s	0.01s	0.00s	0B	0B	root	root	--	-	1	S	3	0%	cupsd					
53713	0.02s	0.00s	0.00s	0B	0B	alexey	alexey	--	-	1	S	6	0%	top					
15	0.02s	0.00s	0.00s	0B	0B	root	root	--	-	1	I	2	0%	rcu_preempt					
87371	0.02s	0.00s	-	0K	0K	root	root	--	-	NE	0	0	0%	<lpstat>					
249	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	1	S	5	0%	systemd-journa					
52263	0.01s	0.00s	0.00s	0B	0B	alexey	alexey	--	-	7	S	4	0%	prlsga					
1123	0.00s	0.01s	0.00s	0B	0B	root	root	--	-	3	S	1	0%	cups-browsed					
51715	0.01s	0.00s	0.00s	0B	0B	alexey	alexey	--	-	1	S	7	0%	dbus-daemon					
401	0.01s	0.00s	0.00s	0B	0B	systemd-	systemd-	--	-	1	S	6	0%	systemd-oomd					
606	0.00s	0.01s	0.00s	0B	0B	avahi	avahi	--	-	1	S	4	0%	avahi-daemon					
1086	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	3	S	6	0%	prlshprint					
40	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	1	S	4	0%	migration/4					
66942	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	1	I	3	0%	kworker/u64:2-					

Отсюда мы делаем вывод, что кроме нашего процесса самое высокое потребление у *gnome-shell*, но пока мы рассматриваем нагрузку только на одно ядро для нас это неважно, потому что *gnome-shell* будет выполняться на другом ядре.

ATOP - alexey-ubuntu															10s elapsed	
2023/10/22 23:57:49																
PRC	sys	0.22s	user	10.08s	#proc	192	#trun	2	#tslpi	204	#tslpu	76	#zombie	0	#exit	27
CPU	sys	3%	user	95%	irq	0%	idle	702%	wait	0%	steal	0%	guest	0%	curf	2.30GHz
cpu	sys	0%	user	100%	irq	0%	idle	0%	cpu000	w 0%	steal	0%	guest	0%	curf	2.30GHz
cpu	sys	1%	user	0%	irq	0%	idle	99%	cpu005	w 0%	steal	0%	guest	0%	curf	2.30GHz
cpu	sys	1%	user	0%	irq	0%	idle	99%	cpu006	w 0%	steal	0%	guest	0%	curf	2.30GHz
cpu	sys	0%	user	0%	irq	0%	idle	99%	cpu003	w 0%	steal	0%	guest	0%	curf	2.30GHz
cpu	sys	0%	user	0%	irq	0%	idle	99%	cpu002	w 0%	steal	0%	guest	0%	curf	2.30GHz
cpu	sys	0%	user	0%	irq	0%	idle	100%	cpu001	w 0%	steal	0%	guest	0%	curf	2.30GHz
cpu	sys	0%	user	0%	irq	0%	idle	100%	cpu004	w 0%	steal	0%	guest	0%	curf	2.30GHz
cpu	sys	0%	user	0%	irq	0%	idle	100%	cpu007	w 0%	steal	0%	guest	0%	curf	2.30GHz
CPL	avg1	0.55	avg5	0.71	avg15	0.45	csu	3240	intr	5347			numcpu	8		
MEM	tot	1.9G	free	1.06	cache	513.4M	dirty	0.2M	buff	34.5M	slab	126.2M	shrss	0.0M	numnode	1
SWP	tot	5.9G	free	5.9G	swcac	0.0M							vmcom	1.3G	vm11m	6.9G
PSI	cpusome	0%	memsome	0%	memfull	0%	iosome	0%	iofull	0%	cs	0/1/10	ms	0/0/0	is	0/0/0
DSK	sda	busy	0%	read	0	write	3	discrd	0	MBr/s	0.0	MBw/s	0.0	avio	2.67 ms	
NET	transport		tcp1	0	tcpo	0	udpi	1	udpo	1	tcpao	0	tcpoo	0	tcprs	0
NET	network		ipi	1	ipo	1	ipfrw	0	deliv	1			icmpl	0	icmpo	0
NET	enp0s5	----	pcki	2	pcko	2	sp	0 Mbps	si	0 Kbps	so	0 Kbps	erri	0	erro	0
PID	SYSCPU	USRCPU	RDELAY	VGROW	RGROW	RUID	EUID	ST	EXC	THR	S	CPUUNR	CPU	CMD	1/3	
3783	0.00s	10.00s	0.00s	0B	0B	alexey	alexey	--	-	1	R	0	100%	stress-ng		
3796	0.03s	0.02s	0.01s	6.5M	2.9M	alexey	alexey	--	-	1	R	6	1%	atop		
620	0.03s	0.02s	0.00s	0B	0B	messageb	messageb	--	-	1	S	6	1%	dbus-daemon		
441	0.02s	0.00s	0.00s	0B	0B	systemd-	systemd-	--	-	1	S	7	0%	systemd-oomd		
613	0.01s	0.01s	0.00s	0B	0B	avahi	avahi	--	-	1	S	2	0%	avahi-daemon		
295	0.02s	0.00s	0.00s	0B	0B	root	root	--	-	1	I	5	0%	kworker/5:2-ev		
3802	0.01s	0.01s	-	OK	OK	root	-	NE	0	0	E	-	0%	<lpstat>		
807	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	1	S	7	0%	cupsd		
1254	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	3	S	5	0%	cups-browsed		
640	0.00s	0.01s	0.00s	0B	0B	root	root	--	-	3	S	6	0%	polkitd		
1303	0.01s	0.00s	0.00s	0B	0B	rtkit	rtkit	--	-	3	S	0	0%	rtkit-daemon		
930	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	4	S	1	0%	prltoolsd		
616	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	1	S	5	0%	atopaccctl		
15	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	1	I	7	0%	rcu_preempt		
73	0.00s	0.01s	0.00s	0B	0B	root	root	--	-	1	S	7	0%	Kcompactd0		
98	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	1	I	4	0%	kworker/4:1-ev		
3205	0.01s	0.00s	0.00s	0B	0B	root	root	--	-	1	I	6	0%	kworker/6:0-ev		
3805	0.01s	0.00s	-	OK	OK	root	-	NE	0	0	E	-	0%	<lpstat>		
3814	0.01s	0.00s	-	OK	OK	root	-	NE	0	0	E	-	0%	<lpstat>		
2005	0.00s	0.00s	0.00s	0B	0B	alexey	alexey	--	-	6	S	2	0%	tracker-miner-		
623	0.00s	0.00s	0.00s	0B	0B	root	root	--	-	3	S	4	0%	NetworkManager		
458	0.00s	0.00s	0.00s	0B	0B	systemd-	systemd-	--	-	2	S	4	0%	systemd-timesy		
997	0.00s	0.00s	0.00s	0B	0B	root	root	--	-	3	S	5	0%	prlshprint		
1060	0.00s	0.00s	0.00s	0B	0B	kernoops	kernoops	--	-	1	S	4	0%	kerneloops		
600	0.00s	0.00s	0.00s	0B	0B	root	root	--	-	1	S	4	0%	acpid		
9	0.00s	0.00s	0.00s	0B	0B	root	root	--	-	1	I	3	0%	kworker/u64:0-		
34	0.00s	0.00s	0.00s	0B	0B	root	root	--	-	1	S	3	0%	migration/3		

Запустили без графической оболочки.

Как и ожидалось прироста производительности нет, пока мы рассматриваем тесты для одного ядра.

```
taskset -c 0 stress-ng --cpu 1 --cpu-method fft --metrics --timeout 120
stress-ng: info: [4312] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [4312] dispatching hogs: 1 cpu
stress-ng: info: [4312] successful run completed in 120.00s (2 mins, 0.00 secs)
stress-ng: info: [4312] stressor      bogo ops real time   usr time   sys time   bogo ops/s   bogo ops/s CPU used per
stress-ng: info: [4312]                      (secs)    (secs)    (secs)    (real time) (usr+sys time) instance (%) 
stress-ng: info: [4312]  cpu          108802    120.00     0.00    906.68    906.68    100.00
[1]+ Done taskset -c 0 stress-ng --cpu 1 --cpu-method fft --metrics --timeout 120
```

### 1.1.6 Команда stress-ng

Воспользуемся встроенным параметрами *stress-ng*.

С помощью опции *-times* можно получить представление о том, сколько пользовательского и системного (ядро) времени используется.

Мы можем запустить *stress-ng* с опцией *-perf*, чтобы увидеть более подробную информацию о том, что делает машина во время работы

```

1 alexey@alexey-ubuntu:~$ taskset -c 0 sudo stress-ng --cpu 1 --cpu-method fft --metrics --timeout 120 --times --perf
[sudo] password for alexey:
3 stress-ng: info: [3347] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [3347] dispatching hogs: 1 cpu
5 stress-ng: info: [3347] successful run completed in 121.46s (2 mins, 1.46 secs)
stress-ng: info: [3347] stressor      bogo ops real time   usr time   sys time   bogo ops/s      bogo ops/s CPU
     used per
7 stress-ng: info: [3347]                               (secs)    (secs)    (secs)    (real time) (usr+sys time)
     instance (%) 
stress-ng: info: [3347]  cpu          109925    121.46    119.88     0.05      905.06      916.58
     98.74
9 stress-ng: info: [3347]  cpu:
stress-ng: info: [3347]    119 873 637 761 CPU Clock           0.99 B/sec
11 stress-ng: info: [3347]    119 867 619 227 Task Clock           0.99 B/sec
stress-ng: info: [3347]    44 Page Faults Total           0,36 /sec
13 stress-ng: info: [3347]    44 Page Faults Minor           0,36 /sec
stress-ng: info: [3347]    0 Page Faults Major           0,00 /sec
15 stress-ng: info: [3347]    763 Context Switches          6,28 /sec
stress-ng: info: [3347]    761 Cgroup Switches          6,27 /sec
17 stress-ng: info: [3347]    0 CPU Migrations          0,00 /sec
stress-ng: info: [3347]    0 Alignment Faults          0,00 /sec
19 stress-ng: info: [3347]    0 Emulation Faults          0,00 /sec
stress-ng: info: [3347]    43 Page Faults User           0,35 /sec
21 stress-ng: info: [3347]    1 Page Faults Kernel          0,01 /sec
stress-ng: info: [3347]    92 System Call Enter          0,76 /sec
23 stress-ng: info: [3347]    91 System Call Exit           0,75 /sec
stress-ng: info: [3347]    1 Kmalloc           0,01 /sec
25 stress-ng: info: [3347]    1 Kfree             0,01 /sec
stress-ng: info: [3347]    3 Kmem Cache Alloc          0,02 /sec
27 stress-ng: info: [3347]    2 Kmem Cache Free           0,02 /sec
stress-ng: info: [3347]    35 MM Page Alloc           0,29 /sec
29 stress-ng: info: [3347]    0 MM Page Free            0,00 /sec
stress-ng: info: [3347]    79.202 RCU Utilization        652.10 /sec
31 stress-ng: info: [3347]    9 Sched Migrate Task          0,07 /sec
stress-ng: info: [3347]    0 Sched Move NUMA           0,00 /sec
33 stress-ng: info: [3347]    765 Sched Wakeup           6,30 /sec
stress-ng: info: [3347]    0 Sched Proc Exec           0,00 /sec
35 stress-ng: info: [3347]    0 Sched Proc Exit           0,00 /sec
stress-ng: info: [3347]    0 Sched Proc Fork           0,00 /sec
37 stress-ng: info: [3347]    0 Sched Proc Free            0,00 /sec
stress-ng: info: [3347]    0 Sched Proc Hang           0,00 /sec
39 stress-ng: info: [3347]    0 Sched Proc Wait           0,00 /sec
stress-ng: info: [3347]    763 Sched Switch           6,28 /sec
41 stress-ng: info: [3347]    2 Signal Generate          0,02 /sec
stress-ng: info: [3347]    1 Signal Deliver           0,01 /sec
43 stress-ng: info: [3347]    32 IRQ Entry             0,26 /sec
stress-ng: info: [3347]    32 IRQ Exit              0,26 /sec
45 stress-ng: info: [3347]    11.185 Soft IRQ Entry         92.09 /sec
stress-ng: info: [3347]    11.185 Soft IRQ Exit          92.09 /sec
47 stress-ng: info: [3347]    0 NMI handler           0,00 /sec
stress-ng: info: [3347]    1 Writeback Dirty Inode        0,01 /sec
49 stress-ng: info: [3347]    0 Migrate MM Pages          0,00 /sec
stress-ng: info: [3347]    1 SKB Consume           0,01 /sec
51 stress-ng: info: [3347]    0 SKB Kfree             0,00 /sec
stress-ng: info: [3347]    0 IOMMU IO Page Fault        0,00 /sec
53 stress-ng: info: [3347]    0 IOMMU Map              0,00 /sec
stress-ng: info: [3347]    0 IOMMU Unmap           0,00 /sec
55 stress-ng: info: [3347]    0 Filemap page-cache add       0,00 /sec
stress-ng: info: [3347]    0 Filemap page-cache del       0,00 /sec
57 stress-ng: info: [3347]    0 OOM Compact Retry          0,00 /sec
stress-ng: info: [3347]    0 OOM Wake Reaper          0,00 /sec
59 stress-ng: info: [3347]    0 Thermal Zone Trip          0,00 /sec
stress-ng: info: [3347] for a 121,46s run time:
61 stress-ng: info: [3347]    971,66s available CPU time

```

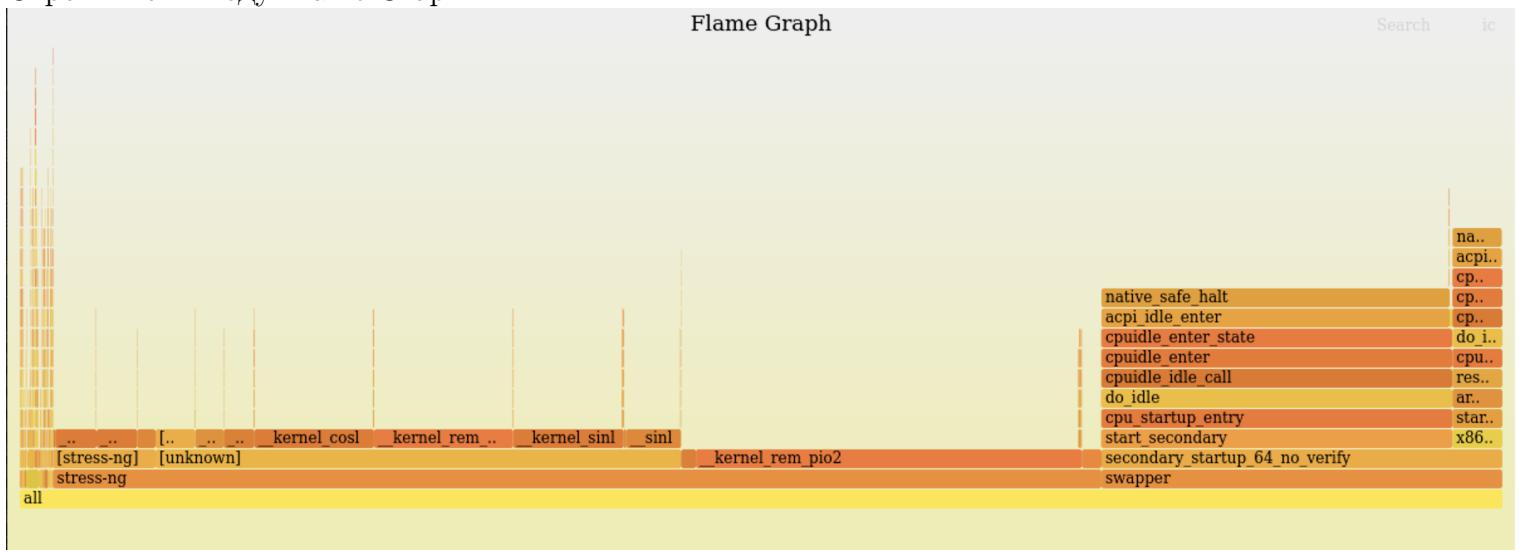
```
stress--ng: info: [3347]      119,88s user time ( 12,34%)
63 stress--ng: info: [3347]       0,05s system time ( 0,01%)
stress--ng: info: [3347]      119,93s total time ( 12,34%)
65 stress--ng: info: [3347] load average: 1,09 0,69 0,30
```

### 1.1.7 Flame Graph

Выполняем профилирование в течении 10 секунд.

```
alexey@alexey-ubuntu:~$ sudo perf record -a -g -F 99 -- sleep 10
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 2,175 MB perf.data (7907 samples) ]
```

## Строим по выводу Flame Graph.



## 1.2 Переходим к активным действиям: nice

Повысим приоритет нашего процесса с помощью команды *nice*.

Чтобы была конкуренция за ресурс процессора, мы запустим тест на все ядра (8 ядер)

```
alexey@alexey-ubuntu:~$ sudo stress-ng --cpu 0 --cpu-method fft --metrics -t 2m
stress-ng: info: [14672] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [14672] dispatching hogs: 8 cpu
stress-ng: info: [14672] successful run completed in 120.00s (2 mins, 0.00 secs)
stress-ng: info: [14672] stressor      bogo ops real time  usr time  sys time  bogo ops/s      bogo ops/s CPU used per
stress-ng: info: [14672]                      (secs)   (secs)   (secs)   (real time) (usr+sys time) instance (%)
stress-ng: info: [14672] cpu          417426    120.00    940.19     3.97    3478.51      442.11     98.35
alexey@alexey-ubuntu:~$ nice -n 19 sudo stress-ng --cpu 0 --cpu-method fft --metrics -t 2m
[sudo] password for alexey:
stress-ng: info: [17281] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [17281] dispatching hogs: 8 cpu
stress-ng: info: [17281] successful run completed in 120.01s (2 mins, 0.01 secs)
stress-ng: info: [17281] stressor      bogo ops real time  usr time  sys time  bogo ops/s      bogo ops/s CPU used per
stress-ng: info: [17281]                      (secs)   (secs)   (secs)   (real time) (usr+sys time) instance (%)
stress-ng: info: [17281] cpu          443405    120.00    942.91     3.12    3695.01      468.70     98.54
```

Видим, что с увеличением приоритета мы немного выиграли в производительности.

### 1.2.1 Команда chrt

Попробуем разные политики планирования для наших процессов.

**RR (Round Robin):** *SCHED\_RR* — это алгоритм планирования циклическим перебором. Как только поток исчерпает свой квант времени, он перемещается в конец очереди на выполнение для своего уровня приоритета. Это позволяет запускать другие потоки с таким же приоритетом. Использует класс планирования *RT*.

```
alexey@alexey-ubuntu:~$ sudo chrt -r 99 nice -n 19 sudo stress-ng --cpu 0 --cpu-method fft --metrics -t 2m
stress-ng: info: [24669] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [24669] dispatching hogs: 8 cpu

stress-ng: info: [24669] successful run completed in 120.01s (2 mins, 0.01 secs)
stress-ng: info: [24669] stressor      bogo ops real time   usr time   sys time   bogo ops/s      bogo ops/s CPU used per
stress-ng: info: [24669]                           (secs)   (secs)   (secs)   (real time) (usr+sys time) instance (%)
stress-ng: info: [24669]   cpu          434915    120.00     912.34     0.01    3624.27      476.70      95.04
```

**FIFO (First In, First Out):** *SCHED\_FIFO* — это алгоритм планирования «первым пришел, первым вышел», который продолжает выполнять поток, находящийся в голове очереди на выполнение, пока тот не оставит процессор добровольно или не появится поток с более высоким приоритетом. Поток продолжает выполняться, даже если в очереди на выполнение есть другие потоки с таким же приоритетом. Использует класс планирования *RT*.

```
alexey@alexey-ubuntu:~$ sudo chrt -f 99 sudo stress-ng --cpu 0 --cpu-method fft --metrics -t 2m
[sudo] password for alexey:
stress-ng: info: [31099] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [31099] dispatching hogs: 8 cpu
stress-ng: info: [31099] successful run completed in 120.00s (2 mins, 0.00 secs)
stress-ng: info: [31099] stressor      bogo ops real time   usr time   sys time   bogo ops/s      bogo ops/s CPU used per
stress-ng: info: [31099]                           (secs)   (secs)   (secs)   (real time) (usr+sys time) instance (%)
stress-ng: info: [31099]   cpu          430410    120.00     911.94     0.25    3586.73      471.84      95.02
```

**NORMAL:** *SCHED\_NORMAL* (ранее известный как *SCHED\_OTHER*) — это алгоритм планирования с разделением времени. Используется по умолчанию для пользовательских процессов. Планировщик динамически корректирует приоритет в зависимости от класса планирования. Для  $O(1)$  длительность кванта времени устанавливается на основе статического приоритета: чем выше приоритет, тем большие квант времени. Для класса *CFS* размер кванта выбирается динамически. Использует класс планирования *CFS*.

```
alexey@alexey-ubuntu:~$ sudo chrt -o 0 sudo stress-ng --cpu 0 --cpu-method fft --metrics -t 2m
stress-ng: info: [26934] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [26934] dispatching hogs: 8 cpu
stress-ng: info: [26934] successful run completed in 120.01s (2 mins, 0.01 secs)
stress-ng: info: [26934] stressor      bogo ops real time   usr time   sys time   bogo ops/s      bogo ops/s CPU used per
stress-ng: info: [26934]                           (secs)   (secs)   (secs)   (real time) (usr+sys time) instance (%)
stress-ng: info: [26934]   cpu          424715    120.00     939.58     4.19    3539.24      450.02      98.31
```

**BATCH:** *SCHED\_BATCH* — этот алгоритм действует как *SCHED\_NORMAL*, но ожидается, что поток будет привязан к процессору и не должен прерывать другие интерактивные задачи, связанные с вводом/выводом. Использует класс планирования *CFS*.

```
alexey@alexey-ubuntu:~$ sudo chrt -b 0 nice -n 19 sudo stress-ng --cpu 0 --cpu-method fft --metrics -t 2m
stress-ng: info: [25315] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [25315] dispatching hogs: 8 cpu
stress-ng: info: [25315] successful run completed in 120.01s (2 mins, 0.01 secs)
stress-ng: info: [25315] stressor      bogo ops real time   usr time   sys time   bogo ops/s      bogo ops/s CPU used per
stress-ng: info: [25315]                           (secs)   (secs)   (secs)   (real time) (usr+sys time) instance (%)
stress-ng: info: [25315]   cpu          441249    120.00     943.89     2.96    3677.02      466.02      98.63
```

**IDLE:::** *SCHED\_IDLE* использует класс планирования *Idle*.

```
alexey@alexey-ubuntu:~$ sudo chrt -i 0 sudo stress-ng --cpu 0 --cpu-method fft --metrics -t 2m
stress-ng: info: [26563] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [26563] dispatching hogs: 8 cpu
stress-ng: info: [26563] successful run completed in 120.01s (2 mins, 0.01 secs)
stress-ng: info: [26563] stressor      bogo ops real time   usr time   sys time   bogo ops/s      bogo ops/s CPU used per
stress-ng: info: [26563]                           (secs)   (secs)   (secs)   (real time) (usr+sys time) instance (%)
stress-ng: info: [26563]   cpu          431462    120.00     935.41     5.60    3595.42      458.51      98.02
```

Как мы видим, выбор политики планирования не сильно влияет на производительность нашего теста.

## 1.2.2 Финал

Отключаем графический интерфейс и применяем все настройки выше.

```
alexey@alexey-ubuntu:~$ sudo chrt -o 0 nice -n 19 sudo stress-ng --cpu 0 --cpu-method fft --metrics -t 2m
stress-ng: info: [8753] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [8753] dispatching hogs: 8 cpu
stress-ng: info: [8753] successful run completed in 120.00s (2 mins, 0.00 secs)
stress-ng: info: [8753] stressor      bogo ops real time   usr time   sys time   bogo ops/s      bogo ops/s CPU used per
stress-ng: info: [8753]                      (secs)    (secs)    (secs)    (real time) (usr+sys time) instance (%)
stress-ng: info: [8753] cpu          496438     120.00    954.72     1.11      4136.94      519.38     99.56
```

Итого мы с **417426** bogo ops смогли повысить производительность до **496438** bogo ops.

## 1.3 cdouble

Так как *cdouble* тоже вычислительная задача, то попробуем применить все настройки выше.

Но сначала тест без настроек.

```
alexey@alexey-ubuntu:~$ stress-ng --cpu 0 --cpu-method cd़ouble --metrics -t 2m
stress-ng: info: [16163] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [16163] dispatching hogs: 8 cpu
stress-ng: info: [16163] successful run completed in 120.11s (2 mins, 0.11 secs)
stress-ng: info: [16163] stressor      bogo ops real time   usr time   sys time   bogo ops/s      bogo ops/s CPU used per
stress-ng: info: [16163]                      (secs)    (secs)    (secs)    (real time) (usr+sys time) instance (%)
stress-ng: info: [16163] cpu          1844831    120.01    914.30     7.99      15372.70     2000.27     96.07
```

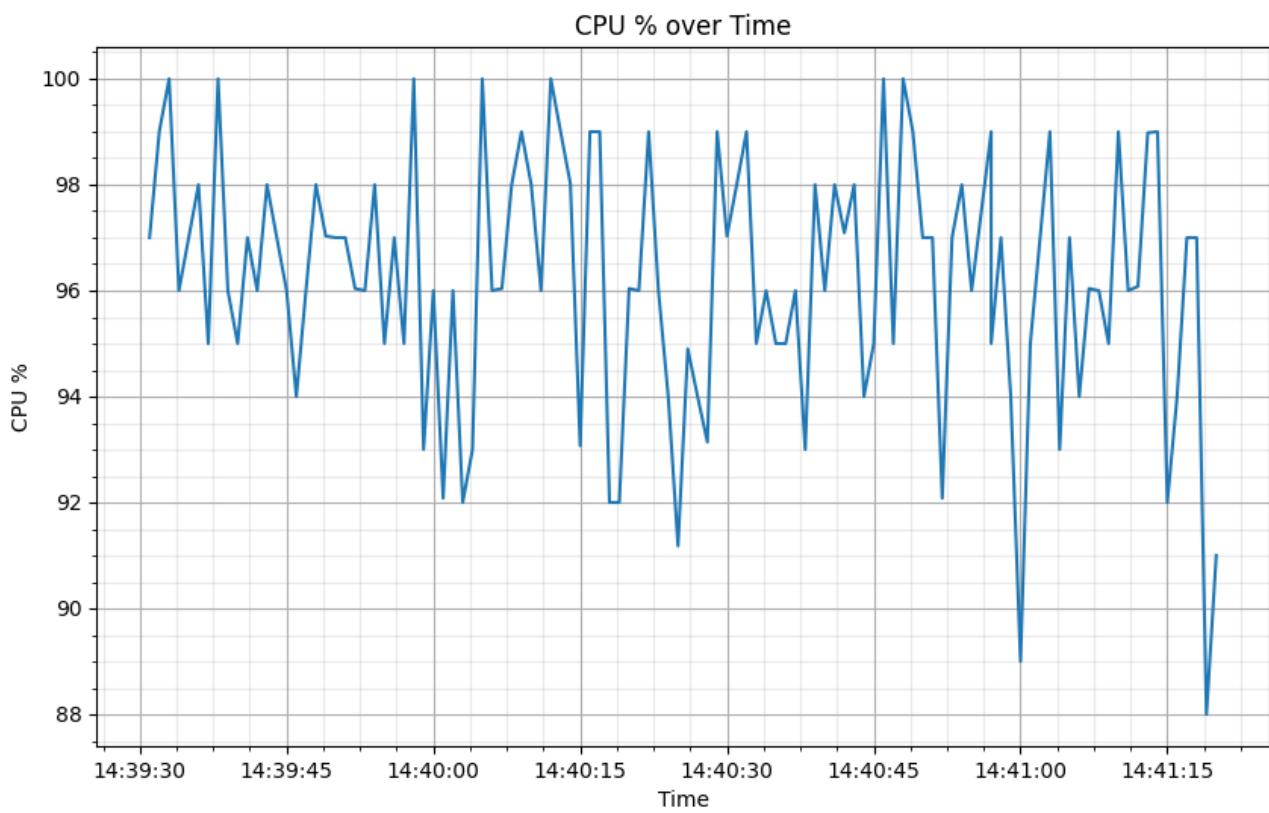
Посмотрим потребление процессора.

14:39:30	UID	PID	%usr	%system	%guest	%wait	%CPU	CPU	Command
14:39:31	1000	16167	96,00	1,00	0,00	4,00	97,00	0	stress-ng
14:39:32	1000	16167	99,01	0,00	0,00	0,99	99,01	0	stress-ng
14:39:33	1000	16167	100,00	0,00	0,00	0,00	100,00	0	stress-ng
14:39:34	1000	16167	96,00	0,00	0,00	2,00	96,00	0	stress-ng
14:39:35	1000	16167	96,00	1,00	0,00	4,00	97,00	0	stress-ng
14:39:36	1000	16167	97,00	1,00	0,00	3,00	98,00	0	stress-ng
14:39:37	1000	16167	95,00	0,00	0,00	4,00	95,00	0	stress-ng
14:39:38	1000	16167	99,00	1,00	0,00	1,00	100,00	0	stress-ng
14:39:39	1000	16167	95,00	1,00	0,00	4,00	96,00	5	stress-ng
14:39:40	1000	16167	95,00	0,00	0,00	5,00	95,00	5	stress-ng
14:39:41	1000	16167	97,00	0,00	0,00	3,00	97,00	5	stress-ng
14:39:42	1000	16167	96,00	0,00	0,00	3,00	96,00	5	stress-ng
14:39:43	1000	16167	97,00	1,00	0,00	3,00	98,00	5	stress-ng
14:39:44	1000	16167	96,00	1,00	0,00	3,00	97,00	5	stress-ng
14:39:45	1000	16167	95,00	1,00	0,00	4,00	96,00	5	stress-ng
14:39:46	1000	16167	93,00	1,00	0,00	6,00	94,00	5	stress-ng
14:39:47	1000	16167	95,05	0,99	0,00	2,97	96,04	5	stress-ng
14:39:48	1000	16167	98,00	0,00	0,00	3,00	98,00	5	stress-ng
14:39:49	1000	16167	95,05	1,98	0,00	1,98	97,03	5	stress-ng
14:39:50	1000	16167	95,00	2,00	0,00	3,00	97,00	3	stress-ng
14:39:51	1000	16167	97,00	0,00	0,00	3,00	97,00	3	stress-ng
14:39:52	1000	16167	95,05	0,99	0,00	3,96	96,04	3	stress-ng
14:39:53	1000	16167	96,00	0,00	0,00	3,00	96,00	3	stress-ng
14:39:54	1000	16167	97,00	1,00	0,00	4,00	98,00	3	stress-ng
14:39:55	1000	16167	95,00	0,00	0,00	3,00	95,00	3	stress-ng
14:39:56	1000	16167	95,00	2,00	0,00	5,00	97,00	3	stress-ng
14:39:57	1000	16167	92,00	3,00	0,00	5,00	95,00	3	stress-ng

14:39:58	1000	16167	100,00	0,00	0,00	0,00	100,00	3	stress-ng
14:39:59	1000	16167	93,00	0,00	0,00	7,00	93,00	3	stress-ng
14:40:00	1000	16167	96,00	0,00	0,00	3,00	96,00	3	stress-ng
14:40:01	1000	16167	92,08	0,00	0,00	6,93	92,08	3	stress-ng
14:40:02	1000	16167	94,00	2,00	0,00	6,00	96,00	3	stress-ng
14:40:03	1000	16167	91,00	1,00	0,00	8,00	92,00	3	stress-ng
14:40:04	1000	16167	93,00	0,00	0,00	6,00	93,00	3	stress-ng
14:40:05	1000	16167	99,00	1,00	0,00	1,00	100,00	3	stress-ng
14:40:06	1000	16167	96,00	0,00	0,00	4,00	96,00	3	stress-ng
14:40:07	1000	16167	96,04	0,00	0,00	2,97	96,04	3	stress-ng
14:40:08	1000	16167	97,00	1,00	0,00	1,00	98,00	3	stress-ng
14:40:09	1000	16167	98,00	1,00	0,00	2,00	99,00	3	stress-ng
14:40:10	1000	16167	98,00	0,00	0,00	2,00	98,00	3	stress-ng
14:40:11	1000	16167	93,00	3,00	0,00	5,00	96,00	3	stress-ng
14:40:11	UID	PID	%usr	%system	%guest	%wait	%CPU	CPU	Command
14:40:12	1000	16167	100,00	0,00	0,00	0,00	100,00	3	stress-ng
14:40:13	1000	16167	99,00	0,00	0,00	1,00	99,00	3	stress-ng
14:40:14	1000	16167	97,03	0,99	0,00	1,98	98,02	3	stress-ng
14:40:15	1000	16167	92,08	0,99	0,00	5,94	93,07	3	stress-ng
14:40:16	1000	16167	99,00	0,00	0,00	2,00	99,00	3	stress-ng
14:40:17	1000	16167	99,00	0,00	0,00	1,00	99,00	5	stress-ng
14:40:18	1000	16167	91,00	1,00	0,00	9,00	92,00	5	stress-ng
14:40:19	1000	16167	92,00	0,00	0,00	6,00	92,00	5	stress-ng
14:40:20	1000	16167	96,04	0,00	0,00	3,96	96,04	5	stress-ng
14:40:21	1000	16167	96,00	0,00	0,00	4,00	96,00	5	stress-ng
14:40:22	1000	16167	99,00	0,00	0,00	2,00	99,00	5	stress-ng
14:40:23	1000	16167	96,00	0,00	0,00	4,00	96,00	5	stress-ng
14:40:24	1000	16167	93,00	1,00	0,00	6,00	94,00	5	stress-ng
14:40:25	1000	16167	88,24	2,94	0,00	9,80	91,18	5	stress-ng
14:40:26	1000	16167	93,88	1,02	0,00	5,10	94,90	5	stress-ng
14:40:27	1000	16167	93,00	1,00	0,00	5,00	94,00	5	stress-ng
14:40:28	1000	16167	93,14	0,00	0,00	5,88	93,14	5	stress-ng
14:40:29	1000	16167	99,00	0,00	0,00	1,00	99,00	5	stress-ng
14:40:30	1000	16167	97,03	0,00	0,00	2,97	97,03	5	stress-ng
14:40:31	1000	16167	97,00	1,00	0,00	2,00	98,00	5	stress-ng
14:40:32	1000	16167	97,00	2,00	0,00	2,00	99,00	5	stress-ng
14:40:33	1000	16167	95,00	0,00	0,00	5,00	95,00	5	stress-ng
14:40:34	1000	16167	93,00	3,00	0,00	4,00	96,00	5	stress-ng
14:40:35	1000	16167	95,00	0,00	0,00	5,00	95,00	5	stress-ng
14:40:36	1000	16167	95,00	0,00	0,00	5,00	95,00	5	stress-ng
14:40:37	1000	16167	95,00	1,00	0,00	4,00	96,00	5	stress-ng
14:40:38	1000	16167	91,00	2,00	0,00	7,00	93,00	5	stress-ng
14:40:39	1000	16167	98,00	0,00	0,00	2,00	98,00	5	stress-ng
14:40:40	1000	16167	93,00	3,00	0,00	4,00	96,00	5	stress-ng
14:40:41	1000	16167	96,00	2,00	0,00	1,00	98,00	5	stress-ng
14:40:42	1000	16167	97,09	0,00	0,00	2,91	97,09	0	stress-ng
14:40:43	1000	16167	97,00	1,00	0,00	3,00	98,00	0	stress-ng

14:40:44	1000	16167	94,00	0,00	0,00	6,00	94,00	0	stress-ng
14:40:45	1000	16167	95,00	0,00	0,00	0,00	95,00	0	stress-ng
14:40:46	1000	16167	96,00	4,00	0,00	5,00	100,00	0	stress-ng
14:40:47	1000	16167	94,00	1,00	0,00	5,00	95,00	0	stress-ng
14:40:48	1000	16167	97,00	3,00	0,00	1,00	100,00	0	stress-ng
14:40:49	1000	16167	99,00	0,00	0,00	0,00	99,00	0	stress-ng
14:40:50	1000	16167	97,00	0,00	0,00	2,00	97,00	0	stress-ng
14:40:51	1000	16167	96,00	1,00	0,00	4,00	97,00	4	stress-ng
14:40:52	1000	16167	90,10	1,98	0,00	7,92	92,08	4	stress-ng
14:40:52	UID	PID	%usr	%system	%guest	%wait	%CPU	CPU	Command
14:40:53	1000	16167	97,00	0,00	0,00	2,00	97,00	4	stress-ng
14:40:54	1000	16167	97,00	1,00	0,00	1,00	98,00	4	stress-ng
14:40:55	1000	16167	95,00	1,00	0,00	5,00	96,00	4	stress-ng
14:40:57	1000	16167	99,00	0,00	0,00	2,00	99,00	4	stress-ng
14:40:57	1000	16167	94,00	1,00	0,00	4,00	95,00	4	stress-ng
14:40:58	1000	16167	97,00	0,00	0,00	3,00	97,00	4	stress-ng
14:40:59	1000	16167	94,00	0,00	0,00	6,00	94,00	4	stress-ng
14:41:00	1000	16167	88,00	1,00	0,00	12,00	89,00	4	stress-ng
14:41:01	1000	16167	93,00	2,00	0,00	4,00	95,00	4	stress-ng
14:41:03	1000	16167	98,00	1,00	0,00	2,00	99,00	1	stress-ng
14:41:04	1000	16167	93,00	0,00	0,00	6,00	93,00	1	stress-ng
14:41:05	1000	16167	96,00	1,00	0,00	4,00	97,00	1	stress-ng
14:41:06	1000	16167	94,00	0,00	0,00	5,00	94,00	1	stress-ng
14:41:07	1000	16167	96,04	0,00	0,00	3,96	96,04	1	stress-ng
14:41:08	1000	16167	92,00	4,00	0,00	4,00	96,00	1	stress-ng
14:41:09	1000	16167	94,00	1,00	0,00	5,00	95,00	1	stress-ng
14:41:10	1000	16167	99,00	0,00	0,00	1,00	99,00	1	stress-ng
14:41:11	1000	16167	95,00	1,00	0,00	4,00	96,00	5	stress-ng
14:41:12	1000	16167	96,08	0,00	0,00	2,94	96,08	5	stress-ng
14:41:13	1000	16167	98,98	0,00	0,00	2,04	98,98	5	stress-ng
14:41:14	1000	16167	99,00	0,00	0,00	1,00	99,00	5	stress-ng
14:41:15	1000	16167	91,00	1,00	0,00	9,00	92,00	5	stress-ng
14:41:16	1000	16167	94,00	0,00	0,00	5,00	94,00	5	stress-ng
14:41:17	1000	16167	96,00	1,00	0,00	4,00	97,00	5	stress-ng
14:41:18	1000	16167	96,00	1,00	0,00	2,00	97,00	6	stress-ng
14:41:19	1000	16167	82,00	6,00	0,00	13,00	88,00	1	stress-ng
14:41:20	1000	16167	89,00	2,00	0,00	9,00	91,00	1	stress-ng

Построим график потребление программой CPU:



Применяем все настройки, как для *fft*. Ставим приоритет 19, пробуем разные политики планирования (самой выгодной опять оказалась NORMAL), закрепляем процессы за конкретными процессорами, отключаем графический интерфейс.

```
alexey@alexey-ubuntu:~$ taskset -c 0-7 sudo chrt -o 0 nice -n 19 stress-ng --cpu 0 --cpu-method cdouble --metrics -t 2m
stress-ng: info: [4789] setting to a 120 second (2 mins, 0.00 secs) run per stressor
stress-ng: info: [4789] dispatching hogs: 8 cpu
stress-ng: info: [4789] successful run completed in 120.01s (2 mins, 0.01 secs)
stress-ng: info: [4789] stressor      bogo ops real time   usr time   sys time   bogo ops/s      bogo ops/s CPU used per
stress-ng: info: [4789]                      (secs)    (secs)    (secs)    (real time) (usr+sys time) instance (%)
stress-ng: info: [4789] cpu          3428179     120.00     956.64      0.27    28568.14      3582.55      99.68
```

Итого с **1'844'831** bogo ops мы повысили производительность до **3'428'179** bogo ops.

