Zihan Peng, 31269720, EECS211 Final Project

Dynamic strategy:

```
// dynamic strategy
acquire(&tick_policy.lock);
// based on number of active processes
if (tick_policy.active_procs > 4) {
  // high load: shorten interval (more frequent scheduling)
  tick_policy.dynamic_interval = 500000;
} else if (tick_policy.active_procs > 1) {
  tick_policy.dynamic_interval = 1000000;
} else {
  // low load: lengthen interval (less frequent scheduling)
  tick_policy.dynamic_interval = 2000000;
}
w_stimecmp(r_time() + tick_policy.dynamic_interval);
release(&tick_policy.lock);
```

When active processes number is over 4, we set the interval as 50000. If the number is in (1,4], we set the interval as 1000000. If the number is 1 or 0, the interval will be 2000000.

In order to get the performance of the strategy, I add some system call functions in the code. (usys.S will be deleted if recompile the code, so do not use “make clean” before “make qemu”, or the code below needs to be added into usys.S)

```
.global get_context_switch_count
get_context_switch_count:
 li a7, SYS_get_context_switch_count
 ecall
 ret
 .global rtime
rtime:
 li a7, SYS_rtime
 ecall
 ret
```

```
fork test
fork test OK
ticks: 1
context switches: 63
time: 1546892
```

The right one is an example of executing forktest.

Performance

Overall results: we can see that dynamic is better.

Averge	Ticks	Switches	Time
fork(RR)	2.6	65.3	3037631.7
fork(Dynamic)	1	63.2	1467974.1
ls(RR)	8.6	8.6	9295702.8
ls(Dynamic)	1.7	2.1	3836782.8
userts(RR)	14687	102851.8	3132070864
userts(Dynamic)	6048.5	99716.3	2814411304

Test results:

1) forktest:

fork(RR)	Ticks	Switches	Time
1	3	68	4087222
2	3	65	3131756
3	3	68	3818832
4	3	65	3303955
5	3	65	2951298
6	2	64	2697525
7	3	65	3297309
8	2	65	2954519
9	2	63	2158203
10	2	65	1975698

fork(Dynamic)	Ticks	Switches	Time
1	1	63	1440202
2	1	63	1409973
3	1	63	1526114
4	1	65	1481211
5	1	63	1719652
6	1	63	1290372
7	1	62	1457428
8	1	63	1482011
9	1	64	1605316
10	1	63	1267462

2) ls

ls(RR)	Ticks	Switches	Time
1	12	12	13089714
2	7	7	7533684
3	10	10	11067356
4	10	10	10865463
5	7	7	7237492
6	6	6	6153951
7	7	7	7951171
8	9	9	9220713
9	9	9	9679556
10	9	9	10157928

ls(Dynamic)	Ticks	Switches	Time
1	2	2	3792822
2	2	2	3752036
3	2	2	3875858
4	2	2	3646368
5	1	3	3671561
6	1	1	3787583
7	2	2	4010010
8	2	4	3742621
9	1	1	3589164
10	2	2	4499805

3) usertests

userts(RR)	Ticks	Switches	Time
1	17858	104921	2393559621
2	18905	106442	3562029589
3	19071	103796	3735301156
4	18912	105764	3554507332
5	14117	102508	2150897447
6	11304	100271	2887706046
7	11464	100708	3050865374
8	11223	100635	2803498778
9	12245	102387	3827902167
10	11771	101086	3354441132

userts(Dynamic)	Ticks	Switches	Time
1	5528	98269	1742579998
2	6401	98830	3523149657
3	6247	101126	3219633391
4	6092	102399	2914087048
5	6049	99010	2796375810
6	6076	98533	2829998458
7	6030	99952	2761313526
8	5970	99277	2732461322
9	6017	99766	2773101417
10	6075	100001	2851412412