New Schedulers in Android Kernel

Project 2:Android scheduler

Haoyi You(519030910193, yuri-you@sjtu.edu.cn)

Department of Computer Science,
Shanghai Jiao Tong University, Shanghai, China

1 Introduction

In this project, my work consists of 3 parts.

In the first parts, I add a ”Weighted Round Robin”(WRR) scheduler into Android kernel. Firstly, I read the code of Android kernel in detail,especially the part of ’rt’. And then I learn about the implement details of Round Robin scheduler in Android kernel. Then I write my own scheduler ”WRR” with the help of ’rt.c’ file.

As for the second parts, I add another scheduler called ”Priority Weighted Round Robin”(PRR) into Android kernel. There is difference between this and previous one and the details I will show in the PRR parts.
Lastly, I manage to compare the performance of my 2 schedulers with the original schedulers in Android kernel.

2 Weighted Round Robin Scheduler:WRR

2.1 Brief Introduction

In the Round Robin Scheduler, it assigns 100 time slice to each real time process fairly, and executes them one by one. If one has used up all of its slice, it will assign 100 more time slice.
However, when the operating system running on the cellphone actually, we cannot regard these real time process fairly, since we prefer that the foreground programs response much faster and care less about the speed of the background programs.

As a result, I plan to assign more time slice to foreground process,(100 time slice), while assign less time slice to background process,(10 time slice). Here is the brief idea of WRR scheduler.

2.2 Implement Details

To implement WRR scheduler, I create a file ”wrr.c” to contain all the implement details. There are several parts in this file to help implement WRR, which I will show them one by one in the following sections.

I must affirm that in my code you may find several meaningless macros or functions. This is used to implement the multi-cpu but I do not complete it because of no device to test and debug.

several new structs and objects

1. **wrr_sched_class**

   In Android kernel, it use a struct called sched_class to create scheduler. Each object of this struct is a scheduler. The struct consists of function pointer, which point to the functions consisting of the scheduler.
   Here I create an object of this called wrr_sched_class.

```
const struct sched_class wrr_sched_class = {
    .next              = &fair_sched_class,         /*Required*/
    .enqueue_task        = enqueue_task_wrr,        /*Required*/
    .dequeue_task        = dequeue_task_wrr,        /*Required*/
    .yield_task      = yield_task_wrr,          /*Required*/
    .check_preempt_curr = check_preempt_curr_wrr,          /*Required*/
    .pick_next_task      = pick_next_task_wrr,          /*Required*/
    .put_prev_task       = put_prev_task_wrr,           /*Required*/
    .set_curr_task           = set_curr_task_wrr,           /*Required*/
    .task_tick       = task_tick_wrr,           /*Required*/
    .get_rr_interval     = get_wrr_interval_wrr,
    .prio_changed        = prio_changed_wrr,            /*Never need impl */
    .switched_to         = switched_to_wrr,             /*Never need impl */
};
```

The next is a pointer into next scheduler, so all the schedulers is in a cycle of linked list. It begins at the object fair_sched_class and end at my object. So I change the last object in original code (rt_sched_class) point to this object and make this pointer points to the head.

2. **sched_wrr_entity**

```
struct sched_wrr_entity {//newly-add
        struct list_head run_list;
        unsigned long timeout;
        unsigned int time_slice;
        int nr_cpus_allowed;
        struct sched_wrr_entity *back;
}
```

This represents an process in WRR scheduler. It contains a list_head object, which is a linked list of all the processes in the scheduler. It also has other members with different usage, such as time_slice records the remaining time slice of this process.

3. **wrr_rq**
It contains the processes in ready queue, which is ready to enter into the WRR scheduler.

```
struct wrr_rq {
        struct list_head wrr;
        unsigned long wrr_nr_running;//number
#if defined CONFIG_SMP || defined CONFIG_WRR_GROUP_SCHED
        struct {
                int curr; /* highest queued wrr task prio */
        } highest_prio;
#endif
        int wrr_throttled;
        u64 wrr_time;
        u64 wrr_runtime;
        /* Nests inside the rq lock: */
        raw_spinlock_t wrr_runtime_lock;
};
```

enqueue_task

This function is used to add a task(process) into this scheduler.

Firstly, it check the legality of the process. If so, it call the inner function **enqueue_prr_entity** and add the number of processes that are running.

In **enqueue_prr_entity**, it has two steps. First it dequeues the process from the ready queue. And after that it add it into the running list. There is a flag to decide which side to the list it add into, head of tail.

dequeue_task
Similarly to enqueue, and if the task still needs running, put it at the tail of the running list.Besides, it needs to renew running queue.

yield_task_wrr
It is used when the task want to yield the control of cpu. Then the scheduler put it at the tail of the linked list.

check_preempt_curr_wrr
There is no preemtive in WRR scheduler.

pick_next_task_wrr
It is used to pick the next process in the linked list. It need to check the fairness. After getting the wrr_entity, it used a function **wrr_task_of** to get responding task_struct valiable.

task_tick_wrr
The scheduler arranges the usage of this task. And in this part, it treats foreground and background process differently. It assigns 100 time slice to foreground but 10 to background.

get_wrr_interval_wrr
It returns the time slice of the process. If it is foreground,it returns 100, else returns 10.

For more details please refer to the code.

2.3 Other files modified

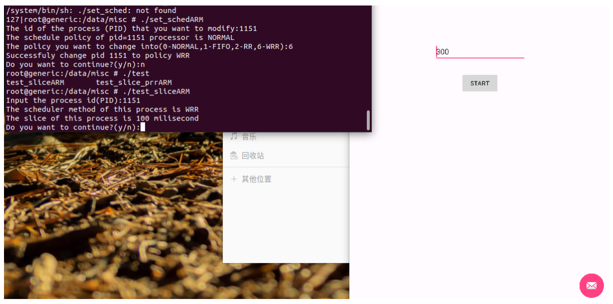
1. **linux\ sched.h**
   (a) Add an macro representing the policy of WRR.
   ```
   #define SCHED_WRR       6
   ```
   (b) Add struct sched_wrr_entity
   (c) Add two macro reprensenting time slice of two type.
   ```
   #define F_WRR_TIMESLICE        (100*HZ/1000)//foreground
   #define B_WRR_TIMESLICE        (10*HZ/1000)//background
   ```
   (d) Add member wrr in task_struct. If this process is in WRR scheduler, it will use this variable.
   ```
   struct sched_wrr_entity wrr;
   ```
2. **kernel\sched\core.c**
   (a) In function **__sched_setscheduler**, add another policy number legal. Then if the policy is equal to this, it does not report error, either.
   ```
   if (policy != SCHED_FIFO && policy != SCHED_RR &&
       policy != SCHED_NORMAL && policy != SCHED_BATCH &&
       policy != SCHED_IDLE&& policy != SCHED_WRR &&
       policy !=SCHED_PRR)        return -EINVAL;
   ```
   (b) In function **___setscheduler**, when changing scheduler, also let the WRR to be legal.
   ```
   if (p->policy == SCHED_WRR)
               p->sched_class = &wrr_sched_class;
   ```
3. **kernel\sched\core.c**
   (a) Add struct **wrr_rq**.
   (b) Add a member in struct **rq**
   ```
       wrr_rq wrr;
   ```
   (c) Declaration of variables, structs in other files.
4. **kernel\sched\rt.c** let the .next pointer point to the object wrr_sched_class rather than fair_sched_class.

2.4 Running performance

To test whether this scheduler can work, I write two files, **set_sched.c** and **test_slice.c**. The former is used to change the scheduler of certain process, and the latter one is used to get the time slice of the processor.

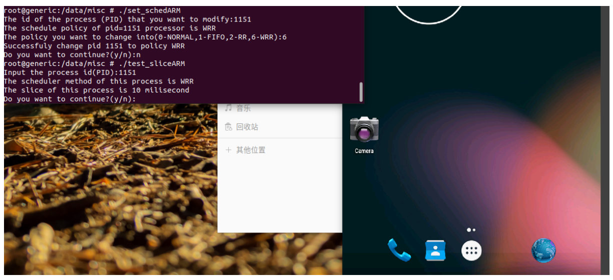
Besides, I create a process that can change between foreground and background with the file **processtest.apk**.

The results are in figure. 1, 2, 3. And you can see the time slice is 100 when foreground,10 when



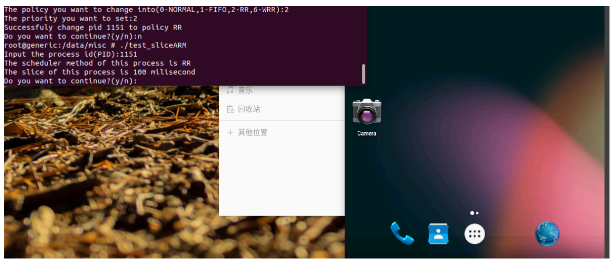
```
/system/bin/sh: ./set_sched: not found
127|root@generic:/data/misc # ./set_schedARM
The id of the process (PID) that you want to modify:1151
The schedule policy of pid=1151 processor is NORMAL
The policy you want to change into(0-NORMAL,1-FIFO,2-RR,6-WRR):6
Successfuly change pid 1151 to policy WRR
Do you want to continue?(y/n):n
root@generic:/data/misc # ./test
test_sliceARM       test_slice_prrARM
root@generic:/data/misc # ./test_sliceARM
Input the process id(PID):1151
The scheduler method of this process is WRR
The slice of this process is 100 milisecond
Do you want to continue?(y/n):
```

Fig. 1. wrr_foreground



```
root@generic:/data/misc # ./set_schedARM
The id of the process (PID) that you want to modify:1151
The schedule policy of pid=1151 processor is WRR
The policy you want to change into(0-NORMAL,1-FIFO,2-RR,6-WRR):6
Successfuly change pid 1151 to policy WRR
Do you want to continue?(y/n):n
root@generic:/data/misc # ./test_sliceARM
Input the process id(PID):1151
The scheduler method of this process is WRR
The slice of this process is 10 milisecond
Do you want to continue?(y/n):
```

Fig. 2. wrr_background



```
The policy you want to change into(0-NORMAL,1-FIFO,2-RR,6-WRR):2
The priority you want to set:2
Successfuly change pid 1151 to policy RR
Do you want to continue?(y/n):n
root@generic:/data/misc # ./test_sliceARM
Input the process id(PID):1151
The scheduler method of this process is RR
The slice of this process is 100 milisecond
Do you want to continue?(y/n):
```

Fig. 3. RR

background,100 when in RR scheduler.

3 Priority Weighted Round Robin Scheduler:PRR

3.1 Brief Introduction

In the previous part, we have realised the Round Robin scheduler, which is more suitable for the cellphone processes that have two types, foreground and background.

However,in the original real time scheduler, it distinguishes the processes from the priority, that is, some processes are more important while the others are no as important as those. But we do not case the priority in WRR scheduler. I hope to optimize the WRR scheduler, so I create Priority Weighted Round Robin Scheduler(PRR).

In this scheduler, it inherits the characteristic of WRR scheduler and real time scheduler. Not only does it distinguish the foreground and background process, for each process, it also has its own priority.

3.2 Implement Details

Just as the WRR scheduler, I also create a file names "prr.c" to maintain all the implement details. Most of the methods and algorithms is similarly to the WRR scheduler. So here I force on the difference of the PRR scheduler.

member priority

In the struct **task_struct**, it has many member to record the priority of the process, such as prio, static_prio, normal_prio,rt_priority.but each of them is used in different scheduler, for instance, rt_priority is used in rt scheduler and normal_prio is used in fair scheduler. All of them is useless for us since it will change when change to its corresponding scheduler.

As a result, I add an new member **priority** used for PRR scheduler.

modify the function___sched_setscheduler

In WRR scheduler, when we call the system call sched_setscheduler, we only use it to change the schedule policy. However, we also need to change the priority of the task.

Since in PRR scheduler, only the member priority reprensents the priority of the process, when it setscheduler to PRR scheduler, it modifies the member priority by the priority-parameter param.

```
if(policy==SCHED_PRR){
        p->priority=param->sched_priority;
        if(param->sched_priority <0||
        param->sched_priority>=PRR_PRIORITY)
        return -EINVAL;
}
```

define slice and priority range

It needs to define macros to determine the priority and slice range. In this part, I also distinguish the foreground and background. I add following in file **linux\ sched.h**.The range can be easily modified, here is just used for test.

```
#define F_MAX_TIMESLICE     (200*HZ/1000)
#define F_MIN_TIMESLICE     (100*HZ/1000)
#define B_MAX_TIMESLICE     (50*HZ/1000)
#define B_MIN_TIMESLICE     (10*HZ/1000)
#define PRR_PRIORITY        (6)
```

change time slice based on the priority.

This involves two functions **task_tick_prr** and **get_prr_interval_prr**. We assign the time slice based on the foreground/background type and priority.

For instance, for foreground process, we assign the F_MAX_TIMESLICE(200) to the highest priority(0), and F_MIN_TIMESLICE(100) to lowest priority(5). The priority between them the time slice is arithmetic.

For the code, we institute the F_WRR_TIMESLICE to:

```
F_MIN_TIMESLICE+(F_MAX_TIMESLICE-F_MIN_TIMESLICE)
 *(PRR_PRIORITY-1-prio)/(PRR_PRIORITY-1)
```

and institute F_WRR_TIMESLICE

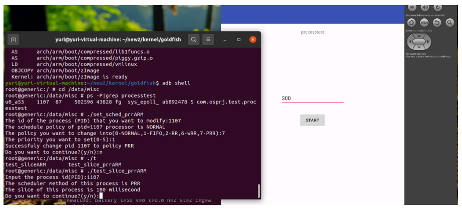
```
B_MIN_TIMESLICE+(B_MAX_TIMESLICE-B_MIN_TIMESLICE)
 *(PRR_PRIORITY-1-prio)/(PRR_PRIORITY-1)
```

Here the we have:

```
int prio=task->priority
```

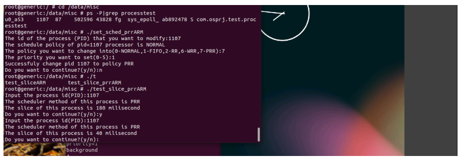
3.3 Running performance

The results are in figure. 4, 5, 6. 7.And you can see when priority=1, foreground slice=180,background slice=40. And when priority=5, foreground slice=100,background slice=10.



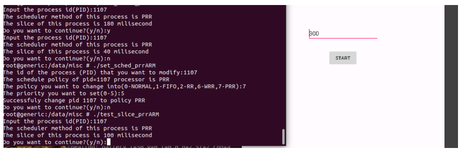
```
yuri@yuri-virtual-machine: ~/new2/kernel/goldfish
  AS      arch/arm/boot/compressed/lib1funcs.o
  AS      arch/arm/boot/compressed/piggy.gzip.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
yuri@yuri-virtual-machine:~/new2/kernel/goldfish$ adb shell
root@generic:/ # cd /data/misc
root@generic:/data/misc # ps -P|grep processtest
u0_a53    1107  87    502596 43828 fg  sys_epoll_ ab892478 S com.osprj.test.proc
esstest
root@generic:/data/misc # ./set_sched_prrARM
The id of the process (PID) that you want to modify:1107
The schedule policy of pid=1107 processor is NORMAL
The policy you want to change into(0-NORMAL,1-FIFO,2-RR,6-WRR,7-PRR):7
The priority you want to set(0-5):1
Successfuly change pid 1107 to policy PRR
Do you want to continue?(y/n):n
root@generic:/data/misc # ./t
test_sliceARM       test_slice_prrARM
root@generic:/data/misc # ./test_slice_prrARM
Input the process id(PID):1107
The scheduler method of this process is PRR
The slice of this process is 180 milisecond
Do you want to continue?(y/n):
```

Fig. 4. priority=1, foreground



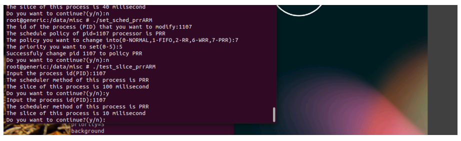
```
root@generic:/ # cd /data/misc
root@generic:/data/misc # ps -P|grep processtest
u0_a53    1107  87    502596 43828 fg  sys_epoll_ ab892478 S com.osprj.test.proc
esstest
root@generic:/data/misc # ./set_sched_prrARM
The id of the process (PID) that you want to modify:1107
The schedule policy of pid=1107 processor is NORMAL
The policy you want to change into(0-NORMAL,1-FIFO,2-RR,6-WRR,7-PRR):7
The priority you want to set(0-5):1
Successfuly change pid 1107 to policy PRR
Do you want to continue?(y/n):n
root@generic:/data/misc # ./t
test_sliceARM       test_slice_prrARM
root@generic:/data/misc # ./test_slice_prrARM
Input the process id(PID):1107
The scheduler method of this process is PRR
The slice of this process is 180 milisecond
Do you want to continue?(y/n):y
Input the process id(PID):1107
The scheduler method of this process is PRR
The slice of this process is 40 milisecond
Do you want to continue?(y/n):
```

Fig. 5. priority=1, background



```
Input the process id(PID):1107
The scheduler method of this process is PRR
The slice of this process is 180 milisecond
Do you want to continue?(y/n):y
Input the process id(PID):1107
The scheduler method of this process is PRR
The slice of this process is 40 milisecond
Do you want to continue?(y/n):n
root@generic:/data/misc # ./set_sched_prrARM
The id of the process (PID) that you want to modify:1107
The schedule policy of pid=1107 processor is PRR
The policy you want to change into(0-NORMAL,1-FIFO,2-RR,6-WRR,7-PRR):7
The priority you want to set(0-5):5
Successfuly change pid 1107 to policy PRR
Do you want to continue?(y/n):n
root@generic:/data/misc # ./test_slice_prrARM
Input the process id(PID):1107
The scheduler method of this process is PRR
The slice of this process is 100 milisecond
Do you want to continue?(y/n):
```

Fig. 6. priority=5, foreground



```
The slice of this process is 40 milisecond
Do you want to continue?(y/n):n
root@generic:/data/misc # ./set_sched_prrARM
The id of the process (PID) that you want to modify:1107
The schedule policy of pid=1107 processor is PRR
The policy you want to change into(0-NORMAL,1-FIFO,2-RR,6-WRR,7-PRR):7
The priority you want to set(0-5):5
Successfuly change pid 1107 to policy PRR
Do you want to continue?(y/n):n
root@generic:/data/misc # ./test_slice_prrARM
Input the process id(PID):1107
The scheduler method of this process is PRR
The slice of this process is 100 milisecond
Do you want to continue?(y/n):y
Input the process id(PID):1107
The scheduler method of this process is PRR
The slice of this process is 10 milisecond
Do you want to continue?(y/n):
```

Fig. 7. priority=5, background