CS411: Final
Comparison of Android x86 kernel and Linux stable kernel

Trevor Bramwell

December 5, 2012

1 Introduction

I will briefly describe my changes to the process scheduler, I/O scheduler. Following that I will compare and contrast the Android-x86's implementations of those concepts with my own implementations. Finally I will explain why I think the Android Developers choose to use the implementations they did.

2 My Designs

Process Scheduling

Since there was already a Round Robin (RR) and First In First Out (FIFO) scheduler in the Linux kernel, nothing was actually designed in this project. We merely replaced what was missing. This included a policy check, enabling of priority scheduling, setting the min and max values of the priority, and updating of the process timeslice.

I/O Scheduling

To implement the Shortest Seek Time algorithm for I/O scheduling, my solution was to add all requests to the end of the list. Then, when a new request needs to be dispatched, the last dispatched request is compared to all the requests in the list until the smallest difference in sectors is found. That request is then put on the dispatch queue.

3 Android-x86 Designs

Process Scheduling

The Android process scheduler has no differences from the Linux kernel. This is probably because CFS works quite well as a general purpose scheduler. Since CFS is focused on response time, it is a good choice for mobile devices.

I/O Scheduling

The differences between the I/O Scheduler of the Android-x86 kernel and the Linux kernel are extremely minimal (as you can see from the following patch). These changes also are more along the lines of bug fixes than actual implementation.

```
--- /nfs/stak/students/b/bramwelt/project3/block/cfq-iosched.c        2012-11-05 15:24:43.003719000
+++ /scratch/Android-x86/Android/kernel/block/cfq-iosched.c        2012-11-28 17:55:55.000000000 -08
@@ -3169,7 +3169,7 @@
                 }
         }

-       if (ret)
```

```
+        if (ret && ret != -EEXIST)
                 printk(KERN_ERR "cfq: cic link failed!\n");

         return ret;
@@ -3185,6 +3185,7 @@
 {
         struct io_context *ioc = NULL;
         struct cfq_io_context *cic;
+        int ret;

         might_sleep_if(gfp_mask & __GFP_WAIT);

@@ -3192,6 +3193,7 @@
         if (!ioc)
                 return NULL;

+retry:
         cic = cfq_cic_lookup(cfqd, ioc);
         if (cic)
                 goto out;
@@ -3200,7 +3202,12 @@
         if (cic == NULL)
                 goto err;

-        if (cfq_cic_link(cfqd, ioc, cic, gfp_mask))
+        ret = cfq_cic_link(cfqd, ioc, cic, gfp_mask);
+        if (ret == -EEXIST) {
+                /* someone has linked cic to ioc already */
+                cfq_cic_free(cic);
+                goto retry;
+        } else if (ret)
                 goto err_free;

 out:
@@ -4015,6 +4022,11 @@

         if (blkio_alloc_blkg_stats(&cfqg->blkg)) {
                 kfree(cfqg);
+
+                spin_lock(&cic_index_lock);
+                ida_remove(&cic_index_ida, cfqd->cic_index);
+                spin_unlock(&cic_index_lock);
+
                 kfree(cfqd);
                 return NULL;
         }
```

4 Conclusion

There is very little to no differences, that I could find, between the Android-x86 kernel and Linux kernel in terms of process and I/O scheduling.

I find this somewhat surprising considering that Android is specifically targeted at mobile devices. Perhaps if we were to go back a few years prior to smart phones, when devices had a lot fewer resources available, we would see a much larger change.

Addendum: Group evaluation

Matthew Okazaki

Matthew contributed a lot to our group. We would not have been able to complete most of the assignments had he not stepped up and done a decent bit of research for each project. He did a great job finding useful and relevant information for our group. As a group member he worked well reminding Brian and I when deadlines were coming up, and scheduling meeting times.

During Project 2 Matthew was instrumental in understanding the shortest seek time first algorithm, and it's implementation.

Brian Cox

Brian worked well in our group. He primarily handled the compiling of our writeup, and submission of our documents. As a member of our group Brian was extremely helpful with his strong knowledge of C and Unix programming. Brian designed almost all of the user space program we used to test our shortest seek time first I/O scheduler.

During the last assignment Brian was also extremely helpful in his work on USB packet sniffing.