Jurnal Nasional Informatika dan Teknologi Jaringan
E-ISSN : 2540-7600
P-ISSN : 2540-7597

I/O (INPUT/OUTPUT) SCHEDULING PADA PERANGKAT ANDROID DENGAN FLASH MEMORY

Sumarno
AMIK Tunas Bangsa
Jl. Sudirman Blok A No. 1-3, Kota Pematang Siantar, Sumatera Utara 21127
smr.str84@gmail.com

***Abstrak*—Sistem operasi Android telah mendapatkan pangsa pasar yang besar. Hal ini telah menjadi salah satu platform yang paling penting dalam sistem komputer saat ini. Dalam perangkat Android, memory flash digunakan sebagai perangkat penyimpanan. Dalam jurnal ini, saya akan mengeksplorasi efek dari I/O scheduling dalam memory flash dan menunjukkan bahwa I/O scheduling memiliki pengaruh pada memory flash saat ini dan akan menganalisis karakteristik kinerja memory flash dan menunjukkan hubungan antara I/O scheduling dan kinerja. Kemudian, akan menunjukkan bahwa kinerja I/O dapat ditingkatkan dengan memilih yang sesuai dengan I/O scheduler.**

***Keywords*— Android, Memory Flash, I/O scheduler.**

I. PENDAHULUAN

Pada saat ini, perangkat memory flash yang berbasis flash Solid-State Drive (SSD), telah banyak digunakan. Di smartphone dan Tablet PC, sebagian besar dari semua perangkat menggunakan memory flash. Namun, kinerja yang bagus belum banyak yang mengetahuinya. Terutama, I/O scheduling pada memory flash karena I/O scheduling tidak berdampak terhadap kinerja memory flash yang tidak memiliki operasi mekanis. Android adalah sistem operasi berbasis Linux untuk perangkat mobile seperti smartphone dan tablet PC. Sistem Android terutama terdiri dari lima bagian, yaitu kernel, library, Android runtime, aplikasi framework , dan aplikasi [3]. Kernel bergantung pada kernel Linux untuk sistem inti seperti proses manajemen, manajemen memory, dan jaringan stack. Oleh karena itu, ini sangat mirip dengan sistem operasi Linux. Library perangkat lunak yang digunakan oleh berbagai komponen sistem Android. Kernel ini dapat digunakan juga oleh aplikasi melalui aplikasi framework. Aplikasi ini menggunakan syntax bahasa pemrograman C/C. Ini semua termasuk komponen fundamental seperti sistem C library dan SQLite. Sistem C library adalah BSD yang diturunkan pelaksanaan sistem library C standar, dirancang untuk embedded perangkat berbasis Linux. SQLite adalah sangat terkenal pelaksanaan RDBMS. Komponen ini juga diharapkan untuk berperilaku sama dengan yang di sistem operasi lain. Runtime Android dan kerangka aplikasi termasuk komponen baru dikembangkan untuk sistem Android. Runtime Android termasuk library inti dan Dalvik VM. Library inti ini merupakan library dasar untuk Java programing. Dalvik VM adalah mesin virtual untuk mengeksekusi Aplikasi android. aplikasi framework termasuk komponen untuk menyediakan platform open source. Ini memberikan kemampuan untuk mengambil keuntungan dari perangkat keras, akses lokasi informasi dan sebagainya. Runtime Android dan framework yang baru dikembangkan untuk Android. I/O Scheduler adalah suatu kemampuan kernel dalam mengontrol pengaksesan terhadap penyimpanan data. Tujuan utamanya adalah mengecilkan latensi pencarian data pada media penyimpanan, prioritas permintaan I/O dari proses yang berjalan, mengalokasikan jumlah ruang penyimpanan untuk setiap proses yang berjalan, dan menjamin setiap permintaan penggunaan media penyimpanan data terpenuhi secara keseluruhan sebelum Deadlinenya habis. Android memiliki tujuh I/O scheduler yaitu : Noop, CFQ, BFQ, Deadline, VR dan SIO. Noop merupakan I/O scheduler yang sederhana untuk android. Noop ini biasanya digunakan sebagai I/O default dari ROM. CFQ (Completely Fair Queuing). I/O ini hampir sama dengan seperti OnDemand pada CPU Governor karena menghasilkan kinerja yang baik dan juga seimbang. BFQ (Budget Fair Queuing) merupakan I/O yang bagus dalam hal transfer USB. Ini adalah pengembangan dari CGQ. Deadline I/O ini sangat sering digunakan pengguna android karena kinerjanya yang baik. VR adalah pengembangan dari Deadline, dan apabila diuji menggunakan benchmark score yang diperoleh akan tinggi. Namun kekurangan dari VR adalah sifatnya yang naik turun sehingga tidak stabil. Simple. Ini adalah I/O Scheduler untuk android yang cukup simple. Ini bisa diandalkan karena baik dalam segi performa dan juga baik dalam segi ketahanan baterai. SIO. Pada I/O SIO: Terdapat unsur Deadline dan Noop pada SIO. Dengan kata lain, SIO adalah Deadline versi ringan, tetapi SIO tidak melakukan penyortiran apapun. Jadi SIO ditujukan untuk Random-Access Devices (seperti SSD hard disks) dimana permintaan untuk penyortiran tidak diperlukan..

II. IDENTIFIKASI MASALAH

Berdasarkan latar belakang diatas maka dapat dirumuskan beberapa permasalah diantaranya adalah:

1. Bagaimana I/O scheduling di perangkat Android dengan memory flash.
2. Bagaimana mengevaluasi kinerja I/O dan efek dari I/O scheduling
3. Bagaimana perbandingan kinerja I/O scheduling Android dan bagaimana memilih Deadline untuk meningkatkan kinerja I/O scheduling.

III. METODOLOGI PENELITIAN

Metode yang digunakan yaitu menggunakan noop I/O scheduling, merupakan semua permintaan I/O yang datang dan mengatur sesuai dengan logika " First In First Out" dan menerapkan permintaan yang digabungkan sekaligus. Sangat cocok digunakan untuk media penyimpanan yang tidak tergantung kepada perpindahan fisik untuk mengakses data seperti memory flash [1]. Memilih I/O elevator yang terbaik, tidak hanya tergantung pada beban kerja, tapi juga hardware. Sebuah disk ATA, SSD, RAID atau sistem storage (memory flash).

Berikut tabel 1 spesifikasi perangkat dari android nexus S dan Nexus 7 untuk contoh perbandingan.

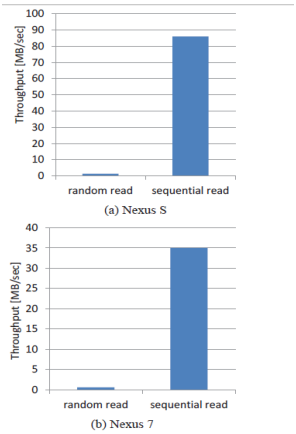
TABEL I
SPESIFIKASI PERANGKAT

Nama	OS	CPU	Memory	FS
Nexus S	Android 4.0.3	Cortex A8 (Hummingbird) Processor 1 GHz	512 MB	VFAT
Nexus 7	Android 4.1.2	NVIDIA tegra 3 mobile processor 1.3 GHz	1 GB	FUSE

IV. HASIL DAN PEMBAHASA

A. Dasar I/O Evaluasi Kinerja

Setelah menguji kinerja sequential read performance dan random read performance perangkat Android dengan memory flash. Spesifikasi perangkat yang digunakan seperti ditunjukkan pada Tabel 1. Dalam percobaan, telah menciptakan sebuah file 1GB dan mengeluarkan 4KB Sistem membaca panggilan 1.000 kali dengan thread tunggal. CFQ, yang tetap I/O scheduler, digunakan. Hasil eksperimen yang ditunjukkan pada Gambar 1 dibawah ini.

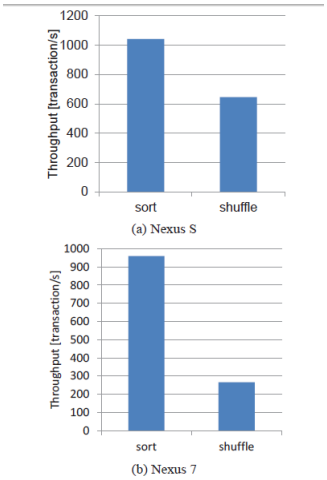


Gbr.1 Random read and sequential read performance

Dengan ini bahwa memory flash dapat memberikan kinerja akses acak tinggi, tetapi hasilnya menunjukkan bahwa kinerja akses sekuensial secara signifikan melebihi kinerja akses acak. Perbedaan kinerja antara akses berurutan dan akses random mungkin disebabkan oleh tidak hanya perangkat penyimpanan tetapi sistem operasi juga, yang buffering yang disediakan oleh sistem operasi.

B. Pengaruh (Efek) I/O scheduling

Pengaruh dari I/O scheduling di Android perangkat dengan memory flash. Setelah menghitung kinerja membaca atau untuk mengakses alamat yang diurutkan dan alamat yang tidak diurutkan. Didalam percobaan, 1 byte membaca akses ke file 1 GB 1.000 kali. 1GB berkas telah dibagi menjadi 1.024 1MB blok dan alamat yang dimulai dari blok telah diakses. Jarak antara alamat diakses telah lebih besar dari 1MB. Dalam kasus "sort", alamat telah diakses dalam urutan ascending. Dalam kasus "shuffle", alamat telah diacak dan akses dilakukan secara acak juga. Hasil pengujian dapat ditunjukkan pada Gambar. 2 dibawah ini:

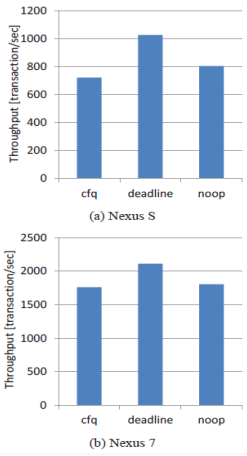


Gbr.2 Access for sorted and shuffled addresses

Grafik tersebut menunjukkan bahwa setelah memasukkan alamat maka dapat dilihat alamat yang diakses dapat meningkatkan kinerja I/O di perangkat Android dengan memory flash.

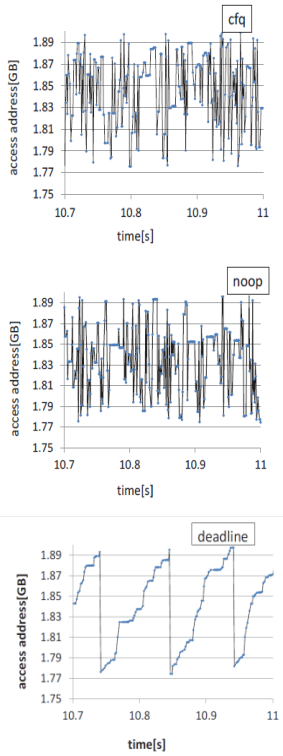
C. I/O Scheduler Evaluasi

Evaluasi kinerja I/O schedulers di Android dan menunjukkan bahwa kinerja I/O dapat ditingkatkan dengan memilih scheduler yang sesuai. Selain itu, memberikan Hasil analisis perilaku schedulers dan mendiskusikan Alasan peningkatan kinerja yang disebabkan oleh I/O schedulers. Benchmark ditulis dalam bahasa C dan dieksekusi di command line shell. Percobaan ini tidak terpengaruh oleh framework Android. Kemudian, efek memory flash perangkat dapat dilihat. Spesifikasi dari perangkat yang digunakan ditunjukkan pada Tabel 1. Untuk pengujian ini telah dibuat system call read 1GB sebuah file dalam memory flash. Ukuran file sistem 1 byte dan alamat akses telah dipilih secara acak di 128 MB file pertama. Proses Benchmark dijalankan dengan 16 threads.. Hasil eksperimen yang menunjukkan pada Gambar. 3 dibawah ini:



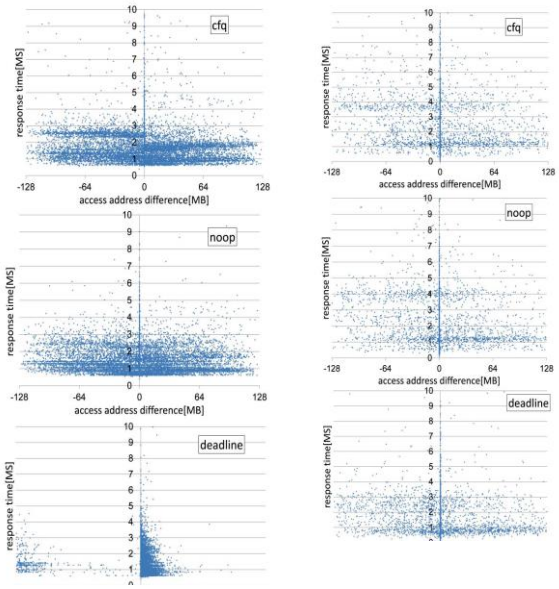
Gbr.3 I/O Scheduler evaluation with native benchmark

Grafik menunjukkan bahwa pilihan I/O scheduler memiliki efek pada kinerja perangkat Android dengan memory flash. Gambar 3 juga menunjukkan bahwa Deadline kinerja lebih tinggi dari pada CFQ, yang merupakan scheduler default, dan noop. Untuk menganalisis hubungan antara I/O scheduler dan kinerja, telah menggunakan monitoring function ke driver MMC Android dan dilihat pada request I/O di Layer MMC [3][4]. Fungsi ini untuk mencatat metode akses (baca/tulis), sebuah alamat akses, dan menampilkan waktu setiap request. karena layer MMC ada di bawah lapisan blok, yang memiliki I/O scheduler, Layer MMC menerima request setelah kembali request oleh I/O scheduler. Untuk melihat hubungan antara waktu yang dikeluarkan dan alamat akses, dapat dilihat pada gambar 4 dibawah ini:



Gbr.4 I/O scheduling results

Hubungan antara alamat akses, perbedaan dan waktu respon, perbedaan alamat akses adalah dihitung dengan mengurangkan alamat akses sebelumnya request dari alamat akses request saat ini. Dapat dilihat pada gambar 5 dibawah ini:



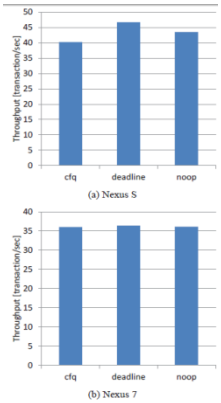
Gbr.5 Monitored I/O request in MMC Layer (Nexus S) kiri dan Monitored I/O request in MMC Layer (Nexus 7) kanan

Dari Gambar 4, kita dapat melihat bahwa akses dieksekusi dalam urutan dalam kasus Deadline scheduler. Di sisi lain, akses tidak dieksekusi dalam rangka seperti dalam kasus CFQ dan noop. Gambar 5 menunjukkan bahwa waktu respon rata-rata dengan positif Perbedaan alamat akses adalah kurang dari itu alamat akses negatif berbeda. Dari ini, kita dapat melihat bahwa alasan utama mengapa Deadline kinerja lebih baik dari yang lainnya yaitu pola akses, urutan, dan kinerja memory flash. Waktu respon yang lebih singkat untuk alamat akses positif yang berbeda.

D. Aplikasi Android Benchmark

Dari Gambar 4, kita dapat melihat bahwa akses dieksekusi dalam urutan dalam kasus Deadline scheduler. Di sisi lain, akses tidak dieksekusi dalam rangka seperti dalam kasus CFQ dan noop. Gambar 5 menunjukkan bahwa waktu respon rata-rata dengan positif Perbedaan alamat akses adalah kurang dari itu alamat akses negatif berbeda. Dari ini, kita dapat melihat bahwa alasan utama mengapa Deadline kinerja lebih baik dari yang lainnya yaitu pola akses, urutan, dan kinerja memory flash. Waktu respon yang lebih singkat untuk alamat akses positif yang berbeda.

Setelah mengevaluasi I/O scheduler dengan aplikasi android. Aplikasi android benchmark ditulis di bahasa java dan dieksekusi pada mesin virtual Dalvik dengan aplikasi framework android. Benchmark menampilkan 10.000 kali 1 byte membaca request ke file 1GB. Alamat akses dipilih secara acak dalam 1GB. Aplikasi ini dijalankan dengan 16 threads. Hasil eksperimen ditunjukkan pada Gambar 7 dibawah ini:



Gbr.6 I/O Scheduler evaluation with application benchmark

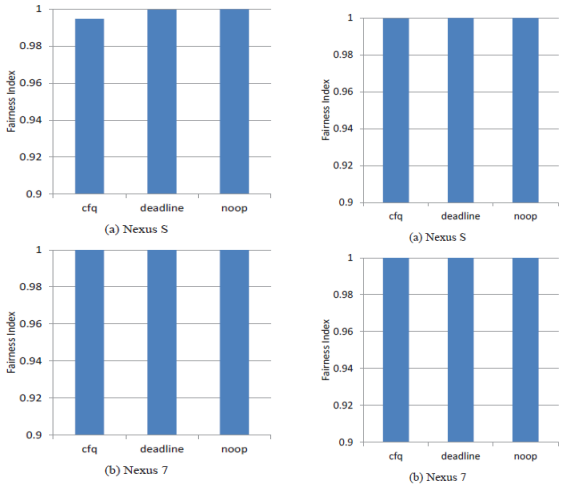
Dari gambar diatas, kita dapat melihat Deadline yang lebih baik dalam hal Nexus S. Kita bisa melihat juga bahwa grafik kinerjanya sangat mirip dalam kasus Nexus 7. Hasil pengujian pada bagian 4.1 dan bagian 4.2 menunjukkan bahwa pilihan I/O scheduler sangat berpengaruh pada kinerja dalam banyak kasus dan Deadline kinerja sudah lebih baik dari schedulers yang lain.

E. Fairness Evaluation

Fairness kinerja I/O schedulers harus dievaluasi. Kita dapat menghitung *jain's fairness index*[4] yang diperoleh melalui rumus sebagai berikut.

$$\text{Fairness Index} = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i} \quad 2$$

Keterangan:
Hasilnya kira – kira dari 1 / n 1. 1 / n adalah kasus terburuk dan 1 adalah kasus terbaik. n adalah jumlah threads. Dalam percobaan ini, itu adalah 16. x1, x2, ... x16 diperoleh dari 16 thread dalam percobaan di bagian 4.1 dan 4.2. Fairness index ditunjukkan pada Gambar 7 dibawah ini:



Gbr.7 Fairness evaluation with native benchmark kiri dan Fairness evaluation with android application benchmark kanan

V. KESIMPULAN

Berdasarkan hasil pengujian pada penelitian yang telah dilakukan, dapat dihasilkan kesimpulan, yaitu:

1. Memory flash tidak melakukan seperti operasi mekanis, tergantung pada flash memory implementasi kontroler. Seperti disebutkan dalam bagian 4, hubungan yang sama diperoleh dengan dua perangkat.
2. Fairness Deadline lebih baik dari penjadwal lainnya. Oleh karena itu, kita dapat mengatakan bahwa memilih Deadline dapat meningkatkan kinerja I/O menjadi lebih baik.
3. Banyak perangkat android saat ini dan mungkin dalam waktu dekat, perangkat android memiliki karakteristik yang sama dalam hal pemakaian kontroler flash memory akan berubah secara drastis di masa depan. Dalam hal ini, karakteristik kinerja harus dianalisis lagi.

REFERENSI

[1] S. Iyer and P. Druschel, 2001 “Anticipatory scheduling: A disk scheduling framework to overcome deceptive idleness in synchronous I/O.” Proc of ACM SOSP.

[2] What is Android?|Android Developers: http://developer.android.com/guide/basics/what is android. html

[3] Steven L. Pratt, Dominique A. Heger, 2004 “Workload Dependent Performance Evaluation of the Linux 2.6 I/O Schedulers,” Ottawa Linux Symposium Proceedings.

[4] TuningI/OPerformance,http://doc.opensuse.org/products /draft/SLES/SLEStuning_sddraft/cha.tuning.io.html

[5] R. Jain, D-M. Chiu and W. Hawe. September, 1984 "A Quantitative Measure of Fairness and Discrimination For Resource Allocation in Shared Conputer Systems," Technical Report TR-301, DEC Research Report