Implementasi Sistem Operasi Embedded IGN pada Board ALIX3d2

Sahrul Arif
Pusat Penelitian Informatika - LIPI
sahrul@informatika.lipi.go.id

Lintang Dwi Febridiani
Pusat Penelitian Informatika - LIPI
lintang@informatika.lipi.go.id

Abstract

We propose an implementation of IGN embedded operating system for ALIX3d2 board. An embedded operating system have been successfully built before. This embedded operating system was rebuilt then to be implemented on ALIX3d2 board. Re-selection of hardware drivers and packages are done on kernel compilation process, to obtain a perfectly match kernel. Tests are conducted on system running to see correctness factor.Tests are also done by comparing various parameters with the previously developed embedded operating system. The result was a compact embedded operating system implemented on ALIX3d2 board.

Keywords : embedded linux, kernel, ALIX3d2

Abstrak

Penelitian ini dilakukan untuk mengimplementasikan sistem operasi Embedded IGN yang tepat untuk board ALIX3d2. Pada penelitian sebelumnya telah dihasilkan sistem operasi IGN embedded. Sistem operasi ini dikompilasi ulang kernelnya untuk diimplementasikan pada board ALIX3d2. Pemilihan driver serta paket-paket yang akan disertakan dalam kernel dilakukan pada saat proses kompilasi kernel untuk mendapatkan hasil kompilasi kernel yang sesuai dengan board ALIX3d2. Pengujian terhadap keberhasilan kompilasi kernel dilakukan dengan pengujian running kernel untuk melihat faktor correctness. Selain itu dilakukan pengujian dengan beberapa parameter dengan hasil kompilasi pada penelitian sebelumnya. Hasil yang diperoleh dari penelitian ini adalah sistem operasi linux embedded yang kompak sebagai IGN embedded yang terpasang pada board ALIX3d2.

.

Kata kunci : embedded linux, kernel, ALIX3d2.

1. Pendahuluan

Linux merupakan salah satu sistem operasi yang populer digunakan saat ini. Sistem operasi ini memiliki banyak kelebihan dibandingkan dengan sistem operasi lain, seperti bersifat *open source,* handal serta mempunyai dukungan *driver* yang lengkap untuk perangkat keras yang beredar di pasaran, [1]. Selain itu *Linux* juga memberikan kebebasan kepada penggunanya untuk memodifikasi *kernel* yang digunakan agar sesuai dengan peruntukan penggunanya.

Beberapa alasan yang menjadikan *linux* sebagai teknologi yang tepat untuk pengembangan *embedded system*, seperti dikemukakan oleh [2], adalah sebagai berikut :

a. *standard based*, yaitu sistem operasi beserta kelengkapannya sesuai dengan standarisasi pada industri.
b. *process isolation and control*, yaitu adanya layanan yang berfungsi sebagai *common API* dalam pengaksesan sumberdaya pada sistem.
c. *peripheral support*, yaitu memiliki dukungan *driver* yang lengkap.
d. *Security*, yaitu dengan adanya *source code* yang terbuka bagi siapapun maka setiap orang dengan mudah dapat menemukan serta memperbaiki celah-celah yang masih ada untuk meningkatkan keamanan sistem.

Dengan beberapa kelebihan yang dimiliki tersebut maka linux banyak digunakan di lingkungan embedded system, selanjutnya disebut embedded Linux. Embedded Linux sebenarnya merupakan salah satu distribusi Linux seperti yang digunakan pada komputer desktop. Bedanya, Embedded Linux disesuaikan dengan perangkat embedded yang memiliki sumber daya yang terbatas, terutama dalam media penyimpanan. Dengan keterbatasan ini maka dibutuhkan kernel dengan ukuran yang kompak/kecil sehingga dapat menghemat penggunaan media peyimpanan. Selain itu dengan ukuran kernel yang kecil akan membuat proses booting menjadi lebih cepat, salah satunya karena driver yang di-load pada saat boot menjadi lebih sedikit.

Pada [3] telah dihasilkan embedded linux pada board ALIX3d2 untuk aplikasi stasiun cuaca, yaitu sistem operasi Embedded IGN. Tujuan dari penelitian ini adalah melakukan konfigurasi ulang terhadap kernel dan melakukan implementasi sistem operasi embedded linux untuk board ALIX3d2.

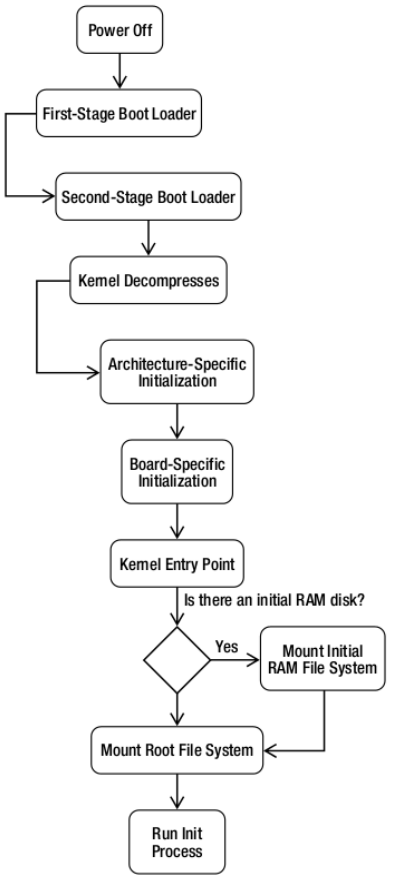
2. Dasar Teori

Untuk melakukan konfigurasi ulang sistem operasi *Linux Embedded*, komponen perangkat lunak dari sistem operasi yang perlu dikompilasi ulang diantaranya *boot loader*, *linux kernel*, *root file system*, dan aplikasi.

2.1 *Boot Loader*

Boot loader atau juga sering disebut *bootsrap loader* adalah sebuah program yang digunakan untuk memuat (me-*load*) program lain [4]. *Boot loader* berukuran kecil, spesifik terhadap sistem target yang akan dibangun. *Boot loader* memiliki kemampuan untuk menemukan *kernel* dan memuat *kernel* tersebut agar sistem operasi dapat dijalankan.

Boot loader dapat dikonfigurasi untuk menjalankan runtutan *script* secara otomatis saat proses *booting* berlangsung atau menunggu masukan dari *user* yang dikirimkan lewat komunikasi serial melalui *console*. Karena mayoritas perangkat *embedded system* tidak memiliki *keyboard* maupun layar monitor, maka interaksi melalui komunikasi serial lebih banyak digunakan. Proses *boot* di *Linux* dapat dilihat pada Gambar 1.



Gambar 1. *Linux boot process*,[2]

Beberapa contoh boot loader antara lain *RedBoot, YAMON, Das U-Boot, LILO,* dan *GRUB*. Pada kegiatan ini dipilih menggunakan *GRUB* (*Grand Unified Bootloader*) yang sesuai dengan arsitektur dari papan target.

2.2 *Linux Kernel*

Kernel Linux dibuat oleh Linus Torvalds, mahasiswa *computer science* berkebangsaan Finlandia, yang dirilis pertama kali pada Agustus tahun 1991 [3]. Pada awalnya sistem operasi ini hanya berjalan di atas *platform x86*. *Kernel Linux* pertama kali di-*porting* ke prosesor *Motorola* 68KB.

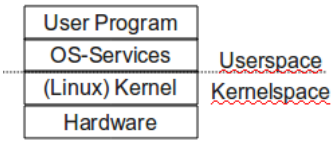
Kernel merupakan perangkat lunak dasar yang bertanggung jawab dalam pengelolaan perangkat keras yang digunakan pada suatu sistem komputer. Selain itu *kernel* juga digunakan sebagai perantara antara aplikasi dan perangkat keras.

Kelebihan yang dimiliki *kernel linux* antara lain bersifat *open source,* komponen-komponen *kernel* dapat dikustomisasi, dapat berjalan di berbagai *platform* perangkat keras, ukuran *kernel* dapat dibuat sangat kompak, kompatible dengan sistem operasi yang lain, dan memiliki dukungan yang penuh dari berbagai komunitas [2].

Kernel Linux bersifat monolitik [2], yaitu terdiri atas semua fungsi yang dibutuhkan untuk menjalankan dan mengatur sistem operasi.

Sistem *kernel* memiliki fungsi utama sebagai berikut:

a. pengelolaan terhadap *file system*
b. *process management*, terdiri atas proses dalam *user mode* dimana ruang dalam memori dimana semua proses *user* dijalankan, maupun *kernel mode* yaitu ruang dalam memori dimana proses *kernel* dijalankan.
c. pengelolaan *device driver*
d. manajemen memori, baik secara *real* maupun *virtual*
e. *networking*



Gambar 2. Arsitektur sistem *UNIX*,[8]

Arsitektur sistem *UNIX* secara umum dapat dilihat pada Gambar 2. Layer paling bawah adalah *hardware*, terdiri dari semua perangkat yang terkoneksi ke sistem, seperti prosesor, memori, *harddisk* dan sebagainya. Layer diatasnya adalah layer *Kernel*. Layer ini befungsi sebagai mediator untuk mengkases *resource* yang dimiliki oleh *hardware* dari *user space*. *User space* terdiri atas layer *OS-Service* dan *User Program*. *OS-service* adalah layanan-layanan yang diberikan oleh sistem operasi sedangkan *User Program* berisi aplikasi maupun program yang dijalankan oleh *user*.

2.3 *Root File System*

Root File System berisikan *library*, aplikasi dan data sistem. Pada proses pengembangan sistem operasi pada *embedded system*, diperlukan integrasi antara *root file system* dengan divais pada *embedded system.*

2.4 *Cross-compile*

Cross-compile adalah suatu proses kompilasi yang dilakukan pada suatu sistem untuk menghasilkan kode-kode yang dapat dijalankan pada sistem yang lain [4]. proses *croos-compile* adalah hal yang umum dilakukan jika sistem yang akan dikembangkan merupakan *embedded system*.

Pada sistem operasi *Linux*, *cross-compile* sering dikaitkan dengan *toolchain*. Hal ini disebabkan ketika

proses *compile* dilakukan, beberapa *tool* dilibatkan bersama-sama untuk menghasilkan elemen-elemen *file executable*, yaitu *compiler, assembler* dan *linker*.

Salah satu cara yang mudah dalam melakukan proses *cross-compile* adalah menggunakan *pre-built toolchain*. *Buildroot* adalah salah satu *pre-built toolchain*, yang merupakan sekumpulan *Makefile* dan *patch* yang digunakan untuk menghasilkan *cross-compilation toolchain* dan *root filesystem* untuk target *board* yang akan digunakan [5,6]. Proses konfigurasi *buildroot* sangat menentukan *Linux embedded system* yang akan terbentuk

3. Metodologi

Tahap-tahap optimasi *kernel Linux* meliputi pembangunan *embedded linux* menggunakan *buildroot*, pengujian *running kernel* pada *board*, dan perbandingan *kernel* embedded IGN yang telah dihasilkan pada penelitian sebelumnya [1].

Pembangunan *kernel* menggunakan *buildroot* terdiri atas :

a. Konfigurasi *buildroot,* yaitu proses pemilihan paket aplikasi yang akan disertakan.
b. Kompilasi *buildroot* untuk mendapatkan *image root filesystem*.
c. Modifikasi *source image* untuk memaketkan hasil kompilasi sehingga dapat diinstalasi ke dalam *Compact Flash (CF) card.*
d. Instalasi *image file system* ke *CF card.*

Pengujian *running kernel* dilakukan untuk melihat faktor *correctness* dari hasil kompilasi *kernel.* Selain itu dilakukan pengujian dengan membandingkan beberapa parameter pada *kernel* baru dengan *kernel* embedded IGN versi sebelumnya (kernel lama). Parameter tersebut adalah ukuran *kernel*, kecepatan *boot,* dan *throughput* dari prosesor, *cache*, dan memori sistem (*timing buffer-cache read*), serta kecepatan dalam mengkompilasi paket.

Konfigurasi yang dikerjakan melalui *buildroot* antara lain adalah :

1. menentukan arsitektur dan varian prosesor
2. *build option*, digunakan untuk mengatur lingkungan pengembangan.
3. *toolchain*, menyediakan *interface* untuk mengatur terutama *cross-compiler.*
4. *System configuration,* mengatur *system hostname*, *system banner*, *baudrate*, ttyS0, dll.
5. *Packet selection,* pemilihan paket yang akan disertakan pada proses kompilasi.
6. *Filesystem images,* untuk menentukan tipe *filesystem* akan akan digunakan.
7. *Bootloader,* bagian yang akan berjalan pertama kali saat mesin dinyalakan, untuk selanjutnya *bootloader* berfungsi untuk meletakkan *kernel* ke memori.

4. Hasil dan pembahasan

Pembangunan *embedded Linux* dengan *buildroot* pada papan ALIX3d2 yang akan digunakan untuk aplikasi stasiun cuaca ini menggunakan konfigurasi sebagai berikut :

a. *target architechture i386*
b. *target architechture variant geode*
c. *Toolchain*, dengan spesifikasi :
   - *buildroot toolchain*
   - *kernel linux* 3.0.1
   - *uClibc* 0.9.31.x
   - *binutils* 2.21
   - *gcc compiler* 4.5.x
d. *Busybox* 1.18.x
e. Paket-paket yang diinstall : *minicom, lighttpd, ethtool, iw, vi*
f. *Filesystem ext2*
g. *Bootloader GRUB*

Pemilihan paket-paket dan perangkat lunak seperti tersebut di atas disesuaikan dengan spesifikasi papan target. Perbedaan dengan *embedded* IGN yang telah dibuat sebelumnya [1] adalah pada versi *kernel linux* yang digunakan. Pada versi menggunakan *kernel linux* versi 2.6.38.5 sedangkan pada penelitian ini kernel yang digunakan adalah versi 3.0.1. karena versi *kernel linux* ini adalah versi terakhir saat penelitian ini dilakukan.

Kernel Linux versi 3.0.1 diperoleh dengan cara mengunduh dari *http://kernel.org*. Perlu dilakukan kustomisasi pada *kernel* 3.0.1 agar sesuai dengan papan ALIX3d2 yang akan digunakan sebagai target. Untuk melakukan terhadap kustomisasi *kernel*, jalankan menu konfigurasi *kernel* dari terminal konsol menggunakan perintah

```
#make menuconfig
```

Berikut *resume* konfigurasi *kernel linux 3.0.1* untuk *board* ALIX3d2:

1. *General setup,* pilihan *optimize for size diaktifkan* karena sistem yang akan dibuat diinginkan memiliki ukuran kernel yang kecil.
2. *Processor type and features*, masuk ke menu *processor family* kemudian dipilih *Geode GX/LX*. Fitur *HPET timer* dan *high resolution timer* tidak diaktifkan karena *processor Geode* tidak memiliki fitur tersebut.
3. *Power management and ACPI options* tidak diaktifkan.
4. *Bus option*, pilihan *PCI support, ISA support* dan *NatSemi SCx200 support* diaktifkan.
5. *Networking support*, hanya pilihan *wireless* yang diaktifkan.
6. Menu *Device driver* merupakan bagian yang krusial karena semua perangkat yang ada pada *board* target dan yang akan dipasang pada *board* harus tersedia *driver*nya di dalam *kernel*. Pilihan yang diaktifkan pada menu ini adalah sebagai berikut :
   a. *ATA/ATAPI/MFM/RLL support*, pilihan *AMD CS5535 support* dan *CS5536 chipset support* diaktifkan.
   b. *Serial ATA and Paralel ATA support,* pilihan *CS5535 PATA support* dan *CS5536 support* diaktifkan
   c. *Netwok device support*, pilihan *EISA, VLB, PCI and on board controllers* dan *VIA*

Rhine support diaktifkan. Pilihan ini ada di dalam menu *Ethernet (10 or 100Mbit).*

d. *Character devices,* pilihan *AMD Geode HW Random Number Generator support* diaktifkan. Pilihan *Support more than 4 legacy serial ports* dan *Extended 8250/16550 serial driver options* diaktifkan, pilihan ini terdapat pada menu *Serial drivers.*
e. *Hardware monitoring support,* pilihan *National Semiconductor LM90 and compatibles* diaktifkan.
f. *HID devices*, pilihan *USB Human Interface Device (full HID) support* diaktifkan.
g. *USB support,* pilihan *EHCI HCD (USB 2.0) support, OHCI HCD support* dan *USB Mass Storage support* diaktifkan.

7. *File systems*, pilihan *Second extended fs support* diaktifkan.
8. *Cryptographic API,* pilihan *Support for the Geode LX AES engine* diaktifkan. Pilihan ini ada dalam menu *Hardware crypto devices*.

Selain pilihan diatas, pilihan lain dapat di non-aktifkan, terutama pada menu *device driver*. Sebab jika tetap diaktifkan maka hanya akan menambah ukuran dari *kernel* hasil kompilasi karena pilihan yang diaktifkan akan dikompilasi menjadi modul-modul di dalam *kernel.* Konfigurasi disimpan kedalam sebuah file *.config* didalam direktori *linux-3.0.1*. File *.config* ini yang akan dijadikan referensi oleh *buildroot* saat melakukan kompilasi *kernel*.

Setelah konfigurasi *kernel* dilakukan, selanjutnya dilakukan kompilasi *kernel* melalui *buildroot*, dengan cukup mengetikkan perintah di bawah ini melalui *terminal* konsol di dalam direktori *buildroot*:

```
#make
```

Hasil dari proses kompilasi tersebut adalah file *rootfs.ext2* dan *bZimage* yang terdapat pada direktori */output/images*. File ini yang akan dipasang sebagai *embedded Linux* pada papan ALIX3d2 dengan menggunakan *CF card*.

Pengujian faktor *correctness* dilakukan dengan melihat running *embedded linux* yang telah dipasang ke papan target ALIX3d2 melalui komunikasi serial menggunakan *personal computer (PC)*. Aplikasi *minicom* dijalankan di *PC* untuk memasukan perintah ke papan target. Skema pengujian sistem seperti pada Gambar 3.

Gambar 3. Skema pengujian sistem

Embedded linux yang terpasang di papan target harus mampu mengenali perangkat keras/*chipset* yang dipasang ke papan target.

Salah satu pengujian untuk membuktikan bahwa *board* telah mengenali perangkat keras yang terpasang, adalah dengan pengujian pada perangkat keras tambahan yang dipasang di papan target. Pengujian dilakukan pada *port mini PCI,* yaitu modul *wifi atheros* dan modul *serial expansio*n. Untuk mengecek apakah perangkat tersebut telah dikenali oleh *kernel* menggunakan perintah

```
#lspci -k
```

Sehingga papan target akan merespon perintah dengan menampilkan perangkat-perangkat keras yang dikenali *kernel* seperti pada Gambar 4.

```
# lspci -k
00:01.0 Class 0600: 1022:2080
00:01.1 Class 0300: 1022:2081 lxfb
00:01.2 Class 1010: 1022:2082 Geode LX AES
00:09.0 Class 0200: 1106:3053 via-rhine
00:0c.0 Class 0280: 168c:0029 ath9k
00:0e.0 Class 0700: 1415:9501 serial
00:0e.1 Class 0680: 1415:9511 serial
00:0f.0 Class 0601: 1022:2090
00:0f.2 Class 0101: 1022:209a cs5536
00:0f.3 Class 0401: 1022:2093
00:0f.4 Class 0c03: 1022:2094 ohci_hcd
00:0f.5 Class 0c03: 1022:2095 ehci_hcd
#
```

Gambar 4. Perangkat keras yang dikenali

Dari hasil pengujian tersebut didapatkan bahwa *kernel* dapat mengenali perangkat keras dan *chipset* yang terpasang di papan target. Hasil perintah *lspci* tersebut menampilkan *ath9k* (modul *wifi*), *serial (serial expansion), Geode LX AES, via-rhine, lxfb, cs5536, ohci_hcd, ehci_hcd*

Pengujian lain dilakukan dengan pengujian *driver.* Pengujian ini dilakukan pada modul *wifi* dengan memasang konfigurasi untuk koneksi *wireless* dengan alamat *ip 192.168.1.55.* Dari *PC* dilakukan pengetesan koneksi menggunakan perintah *ping* ke alamat *ip 192.168.1.5.* Dari hasil yang diperoleh terlihat bahwa *reply* diterima oleh *PC* dari alamat tersebut

Dari pengujian ini mengindikasikan bahwa konfigurasi *kernel* sudah benar dimana setiap *driver* yang dibutuhkan telah ter*install* dengan baik.

Kernel yang dihasilkan dari penelitian ini berukuran 53,8 *Megabyte* sedangkan *kernel* sebelumya berukuran 137 *Megabyte.* Artinya, optimasi ukuran kernel berhasil dilakukan sehingga menghasilkan ukuran *kernel* yang jauh lebih kecil.

Selain itu, keberhasilan optimasi juga dilihat dari kecepatan proses *boot* yang dapat ditampilkan dengan menggunakan perintah

```
#dmesg
```

Sedangkan *throughput* dari prosesor, cache, dan memori sistem diketahui dengan melihat kecepatan *timming buffer-cache reads* menggunakan perintah

```
#hdparm -T /dev/hda1
```

atau

```
#hdparm -T /dev/sda1
```

Terdapat perbedaan penamaan media penyimpanan data, dimana pada *kernel* lama media penyimpanan dikenal dengan nama *sda1*, sedangkan pada *kernel* baru dikenal dengan *hda1*.

Pada *kernel* sebelumnya proses *boot* membutuhkan waktu 6,492 detik, sedangkan untuk *kernel* yang baru membutuhkan waktu 5,934 detik. Kecepatan *timming buffer-cache reads* pada *kernel* lama 343.006 kB/s

sedangkan pada *kernel* baru lebih cepat menjadi 366.750 kB/s.

Perbandingan parameter ukuran, kecepatan *boot kernel,* dan *throughput* dari prosesor, cache, dan memori sistem dapat dilihat pada Tabel 1.

Tabel 1. Perbandingan Kernel

Kernel	Ukuran (MB)	Kecepatan Boot (detik)	*Throughput* (kB/s)
Lama	137	6,492	343.006
Baru	53,8	5,934	366.750

Pada *kernel* sebelumnya [1] masih terdapat beberapa konfigurasi yang belum optimal, seperti pemasangan beberapa *driver* yang tidak digunakan sehingga ukuran *kernel* menjadi lebih besar. Selain itu proses *boot* menjadi lebih lambat karena selama proses *boot kernel* menjalankan proses pencarian *driver* yang lebih banyak.

Throughput pada *kernel* baru yang ditunjukkan oleh *timming buffer-cache reads* lebih baik daripada *kernel* lama, hal ini dikarenakan proses yang terjadi pada *kernel* baru lebih sedikit.

Parameter perbandingan berikutnya, adalah kecepatan dalam melakukan kompilasi paket. Pada uji coba ini digunakan paket *Ethtool6*, yaitu paket untuk melihat dan mengkonfigurasi divais *ethernet* pada *board.* Agar hasil lebih akurat, dilakukan beberapa kali kompilasi dan diambil nilai rata-ratanya. Dengan perintah `#time` untuk mengukur durasi proses kompilasi, akan diperoleh tiga buah hasil catatan waktu, yaitu *real, user* dan *sys*. *Real time* menunjukkan keseluruhan waktu yang digunakan untuk kompilasi sejak perintah dipanggil. *User time* adalah lamanya waktu yang digunakan CPU untuk melakukan proses dalam *user-mode* (di luar *kernel*). Sedangkan *Sys time* adalah waktu yang digunakan kernel pada CPU untuk melakukan proses. Tabel 2 menunjukkan hasil perbandingan waktu yang dibutuhkan untuk kompilasi paket *Ethtool6*.

Tabel 2. Perbandingan waktu yang dibutuhkan untuk kompilasi paket

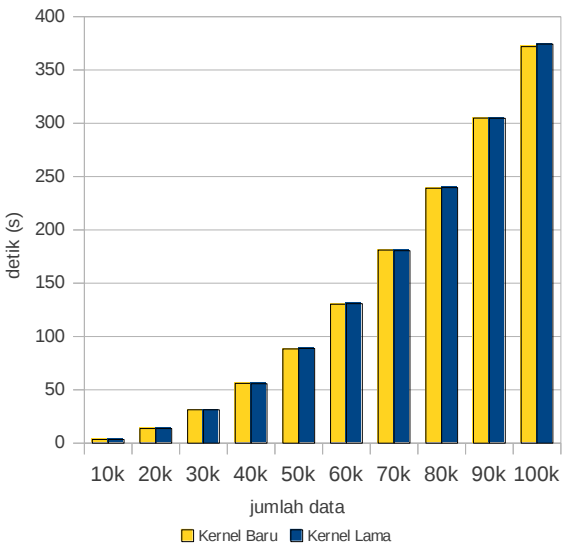
Kernel	Real time (detik)	User time (detik)	Sys time (detik)
Lama	35,07	33,34	1,72
Baru	34,88	32,61	1,64
Selisih	-0,19	-0,73	-0,08
%	0,54 %	2,19%	4,65%

Waktu (*real time)* rata-rata yang dibutuhkan kernel baru untuk kompilasi paket adalah 34, 88 detik, lebih cepat 0,19 detik atau 0,54% dibandingkan waktu yang diperlukan oleh kernel lama. Untuk rata-rata *user time*, *kernel* baru menyelesaikan dalam waktu 32,61 detik atau lebih cepat 0,73 detik atau 2,19%. Sedangkan untuk rata-rata *sys time*, *kernel* baru membutuhkan waktu 1,64 detik, lebih cepat 0,08 detik atau 4,65% dari *kernel* lama. Ada korelasi antara optimasi *kernel* dengan kecepatan kompilasi paket.

Satu parameter lagi yang diuji adalah kecepatan dalam menjalankan aplikasi. Aplikasi yang diuji adalah aplikasi sederhana untuk mengurutkan data yang di-*generate* secara acak, dengan variasi jumlah data, yaitu antara 10.000 (10k) sampai 100.000 (100k) data dengan interval 10.000. Dari percobaan didapatkan data-data waktu yang dibutuhkan (*real time*) untuk mengeksekusi program pengurutan data seperti pada Tabel 3 dan Gambar 5.

Tabel 3. Perbandingan waktu yang dibutuhkan untuk eksekusi program pengurutan

Jumlah data	Kernel Lama (detik)	Kernel Baru (detik)	Selisih (detik)	Selisih (%)
10k	3,410	3,404	0,006	0,2
20k	13,680	13,686	-0,065	0,0
30k	31,280	31,235	0,045	0,1
40k	55,720	56,047	-0,328	-0,6
50k	88,785	88,375	0,410	0,5
60k	130,795	130,240	0,555	0,4
70k	180,500	181,080	-0,580	-0,3
80k	239,625	239,000	0,625	0,3
90k	304,665	304,850	-0,185	-0,1
100k	374,290	371,890	2,400	0,6



Gambar 5: Perbandingan waktu yang dibutuhkan untuk eksekusi program

Saat menjalankan program untuk 10k data, *kernel* baru menyelesaikan dalam 3,404 detik, sedangkan *kernel* lama selesai dalam 3,410 detik, lebih lama 0,006 detik. Saat data ditambah menjadi 20.000 data, *kernel* baru

mengerjakan lebih lama 0,0065. Ternyata *kernel* baru tidak selalu lebih cepat dalam mengeksekusi program dibandingkan *kernel* lama. Pada beberapa percobaan, *kernel* baru menyelesaikan sedikit lebih cepat dari *kernel* lama, namun pada percobaan lainnya, *kernel* baru membutuhkan lebih banyak waktu, seperti pada percobaan dengan 20k, 40k, 70k dan 90k data. Hal ini dapat dipengaruhi beberapa faktor pada saat percobaan, diantaranya data yang diberikan untuk pengurutan tidak persis sama, spesifikasi CF yang digunakan tidak persis sama, minimnya jumlah data sampling dan kondisi papan target yang mulai *over-heat.*

5. Kesimpulan

Dari pengujian yang telah dilakukan dapat disimpulkan bahwa kompilasi kernel berhasil dan dapat berjalan dengan baik pada board ALIX3d2. Dengan konfigurasi yang tepat dihasilkan kernel yang lebih kompak daripada kernel sebelumnya. Sehingga pada parameter-parameter yang dibandingkan, kernel baru menunjukkan performa yang lebih baik daripada kernel lama dalam hal ukuran, kecepatan boot, dan throughput. Namun tidak ditemukan adanya korelasi antara optimasi kernel yang kompak dengan kecepatan eksekusi program yang diujikan.

6. Daftar Pustaka

[1] Febridiani, Lintang Dwi dan Arif, Sahrul, "Membangun Embedded Linux pada ALIX3d2 dengan Buildroot pada Aplikasi Stasiun Cuaca". *Jurnal INKOM*, Vol 5 no 2. 2011.

[2] P. Bovet, Daniel and Cesati, Marco, *Understanding the Linux Kernel,* USA : O'Reilly, 2000.

[3] Sally, Gene, *Pro Linux Embedded Systems,* USA: Apress, 2010.

[4] Seebach, Peter, *Build an embedded Linux distro from scratch,* IBM Corporation, 2008.

[5] Andersen, Erick, "Buildroot: making Embedded Linux easy". Buildroot, 2005. http://buildroot.uclibc.org/ (Diakses 01.2011)

[6] Hintermann, Martin, *Operating System Components for an Embedded Linux System,* German : Institute for Real-Time Computer System, TU Munchen, 2007.