ICSを
ビルドしてみた

横浜Androidプラットフォーム部
2011.11.19
kinneko@gmail.com

前回やらなかったので、
自己紹介

そんなのイラナイ！
早く先にススメ！

って人、挙手！

でもヤル！

- **今日の役割：横浜PF部メンバ**
  日本Androidの会
    運営委員/金沢支部長/組み込みWG/
- **税務署的な職業：**
  サラリーマン/組み込みLinux製品企画開発
  ７月からTabletやってますが...
- **keyword:**
  Linux, iohack, Zope, bittorrent, KNOPPIX, LANTANK, GLANTANK, MAKAI, web.py, CryptoNAS, FreeTANK, Chumby, Android

最近の活動

- オライリー『初めてのAndroid 第３版』

  監修チーム

- 日経Linux 2011年３月号 (2/8)

  最新 Android 2.3 大解剖
  Part4: PC向け最新版「Android 2.2」をネットブックで動かす

- ASCII technologies 2011年３月号 (2/24)

  徹底解剖 Android 2.3＆3.0
  Part 2 ここが変わった！Androidアプリケーション開発環境構築法
  Part 3 Gingerbreadで大きく変わった
  Androidのフルシステムビルド環境を作る

- 週刊アスキー 2011年11月1日号（10/18）

  Android-x86のご意見版 きんねこ氏にハニカム版について聞いてみた！

お約束！

転職先募集中!

では、本題。

ICS、syncして
ビルドしてみた人、
挙手！

まぁ、そんなに
いないよね…

ってか、いたら驚く！

前回のAOSPミラーを
repo syncで
アップデートする

まる２日くらい
かかりました…

EMobileなので
帯域が狭いだけです…

error: RPC failed; result=28, HTTP code = 0
fatal: The remote end hung up unexpectedly

今回も多発しました...

error: Exited sync due to fetch errors

今回も多発しました...

error: Exited sync due to fetch errors
error: Exited sync due to fetch errors
error: Exited sync due to fetch errors
error: Exited sync due to fetch errors
error: Exited sync due to fetch errors
error: Exited sync due to fetch errors
error: Exited sync due to fetch errors
error: Exited sync due to fetch errors
error: Exited sync due to fetch errors
error: Exited sync due to fetch errors
error: Exited sync due to fetch errors
error: Exited sync due to fetch errors

使用容量は？

kinneko@BuildSV:~/AOSP$ du -sh ./

8.1G ./

Gingerbreadでは、3.3GBだったのよ...

ローカルミラーから repo syncする

```
kinneko@BuildSV:~$ mkdir ICS
kinneko@BuildSV:~$ cd ICS/
kinneko@BuildSV:~/ICS$ ../repo init -u ../
AOSP/platform/manifest.git -b
android-4.0.1_r1
kinneko@BuildSV:~/ICS$ time ../repo sync
(略)
repo initialized in /home/kinneko/ICS
```

- 前回の教訓 -

ミラーのsyncがうまく
いったからといって、
ファイルが壊れていない
保証はない！

今回は、
リポジトリの破損は
ありませんでした

- 前回の教訓 -

快適すぎです。

AOSPは
ミラーして使いましょう

手順は前回資料で！

fetchするプロジェクト数は？

221

Gingerbreadでは、170だったのよ...

当然ながらsyncの時間も長くなる。

syncにかかる時間は？

real 6m43.091s

user 12m13.370s
sys　1m0.880s

Gingerbreadでは、
real 2m12.760sだったのよ…
３倍弱…

とてもネット経由で
単体syncやる気にならないな...

鯖側の負荷もバカにならないだろうし、
AOSPのミラー手順が準備されるわけだね。

そろそろ、重いターゲット部分は
分離式にしたほうがいいのじゃないだろうか。

あと、-j指定しなくても
repoが２スレッド動作している疑惑。

ミラーの時も１つずつ取得ではなかった。
前から？

ビルドしてみる

- 前回の教訓 -

ビルド時間は
お金で買える！

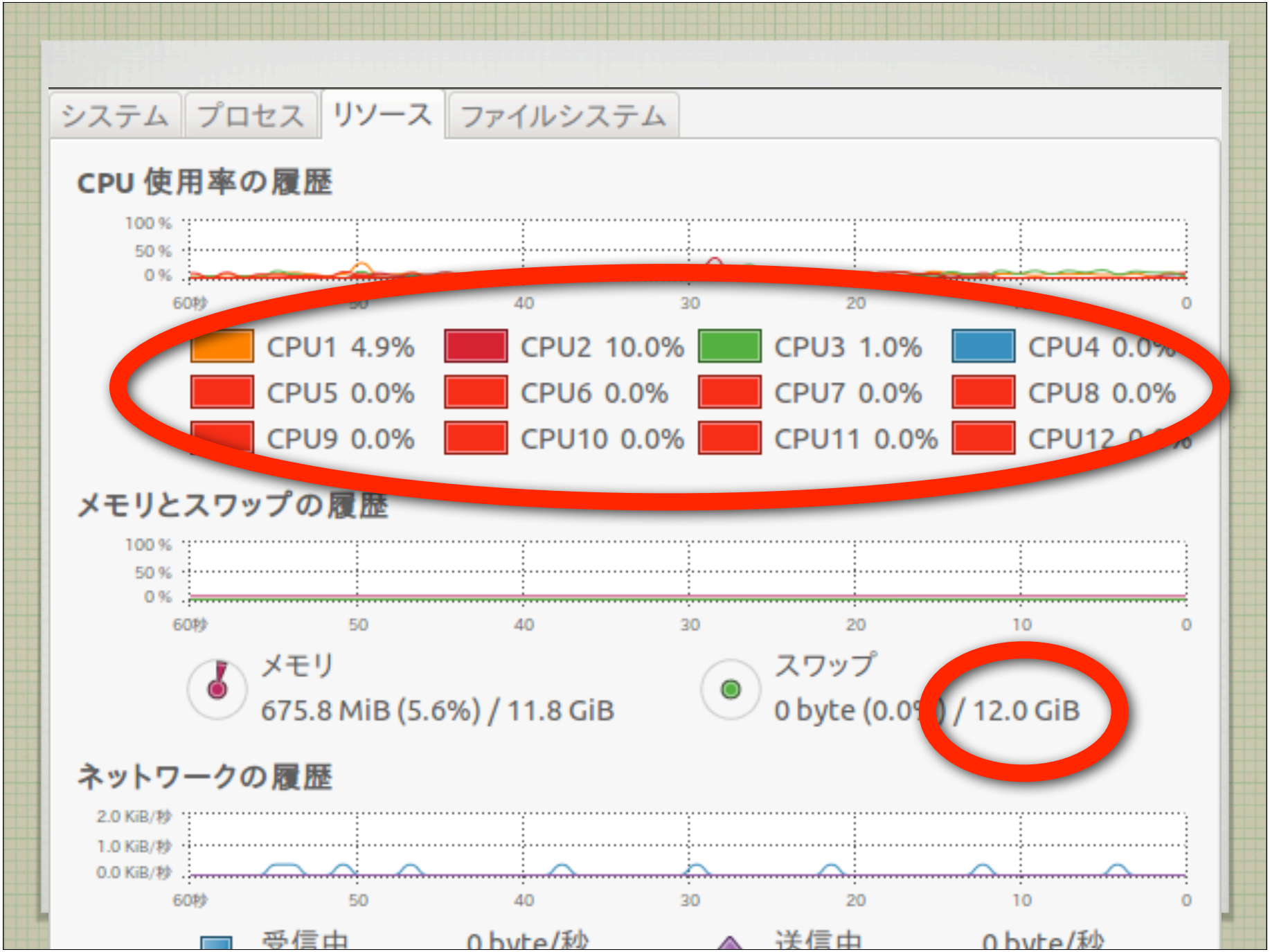
model name :

Intel(R) Core(TM) i7 CPU 980 @

3.33GHz

MemTotal:

12323396 kB


システム
プロセス
リソース
ファイルシステム
CPU 使用率の履歴
100 %
50 %
0 %
60秒
50
40
30
20
0
CPU1 4.9%
CPU2 10.0%
CPU3 1.0%
CPU4 0.0%
CPU5 0.0%
CPU6 0.0%
CPU7 0.0%
CPU8 0.0%
CPU9 0.0%
CPU10 0.0%
CPU11 0.0%
メモリとスワップの履歴
100 %
50 %
0 %
60秒
50
40
30
20
10
0
メモリ
675.8 MiB (5.6%) / 11.8 GiB
スワップ
0 byte (0.0
/ 12.0 GiB
ネットワークの履歴
2.0 KiB/秒
1.0 KiB/秒
0.0 KiB/秒
60秒
50
40
30
20
10
0
受信中
0 byte/秒
送信中
0 byte/秒

- 前回の教訓 -

どのみち、
ICSのビルドには、
このくらいのリソースが
必要です。

その検証

```
kinneko@BuildSV:~/ICS$ export ARCH=arm
kinneko@BuildSV:~/ICS$ export PATH=/home/kinneko/
panda/L27.12.1-P2/build_tools/arm-2010q1/bin:/usr/
bin:/bin
kinneko@BuildSV:~/ICS$ export CROSS_COMPILE=arm-
none-linux-gnueabi-
kinneko@BuildSV:~/ICS$ . build/envsetup.sh
including device/samsung/maguro/vendorsetup.sh
including device/samsung/tuna/vendorsetup.sh
including device/ti/panda/vendorsetup.sh
including sdk/bash_completion/adb.bash
```

```
kinneko@BuildSV:~/ICS$ lunch

You're building on Linux

Lunch menu... pick a combo:
     1. full-eng
     2. full_x86-eng
     3. vbox_x86-eng
     4. full_maguro-userdebug
     5. full_tuna-userdebug
     6. full_panda-eng

Which would you like? [full-eng] 1
```

マグロにツナ？
トロもあるらしい…

Panda標準かよ！

```
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.0.1
TARGET_PRODUCT=full
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=ITL41D
============================================
```

kinneko@BuildSV:~/ICS$ time make -j12

==========

(略)

real 27m59.741s

user 248m55.610s

sys 14m48.130s

Gingerbreadでは、12m2.785s。2.3倍くらい。

RAMの使用は?

ピーク時で**9.1GB**

コンスタントに3G中盤〜後半あたりは使う。
Ubuntu込みで、RAM 8GBが実用ミニマムだろう。
実際は12GBほしいところ。

4GB台は普通に利用があるので、
4GRAMだとswapしてさらに遅くなる。

必要なCPU数は？

CPUは常に上に張り付いているわけではないけど、
７割くらいはいっぱい。

ビルド環境にシングルコアではお話にならないだろう。

時間かかってもよければ、仮想含めて最低４コア？

普通は **８コアはほしい。**

リポジトリサイズと、ソースコードサイズは？

kinneko@BuildSV:~/ICStest$ du -sh ./

9.0G ./

kinneko@BuildSV:~/ICStest$ rm -rf ./*
kinneko@BuildSV:~/ICStest$ du -sh ./
4.4G ./

4.6GBがソースコードのサイズか。

ビルドに必要な容量は？

full-engのビルド上がりだと、

kinneko@BuildSV:~/ICS$ du -sh ./

22G ./

Gingerbreadでは、仮想マシンに16GB割り当てれば、
結構余裕があったのに...

複数プロジェクトを維持するには、
結構なディスク容量を必要とするようになりました。

マシンパワーないと
死ぬほどでもない。

でも、やっぱり
「力は正義」

おしまい

おまけ

のりつなたん

うちのノートさんだと、

$ make -j8 で、

50m21sか・・・

倍増だな。

まごろくさん

$ time make -j 12

real 40m28.223s

user 223m53.860s
sys 13m56.310s
6coreなので、40.5mx6=243m ,(223.9+13.9)/243=0.97

CPU使用率97%。メモリ16Gあると、waitなし。

こっちも倍増です。

てつこばさん

私の環境だとICSのビルドは

make -j8 で

36分。

solaさん

core-i7 2.8GHz、

メモリ 16GB で

make -j8 した結果は

43m7.136s。
メモリは 9GB くらい使っていた時があった。
（ずっとは見てないので、最大かは不明）

TACさめ

3GHz x 4, 3GB のマシン

でICSビルドに

j4 で

269分 掛かった.

おそろしや

恐竜先生

make -j4。

real 71m6.482s

user 253m49.840s
sys 10m54.710s

RAM 6GBでもほぼswapせずに完了。

MAX 4.5GBくらいだった。

恐竜先生

-j4でx86ビルドしたら、
swapに1.5Gくらい飛んで進まなくなって涙目。

jarからdex作るところで-Xmx1536Mになってるのが
一番メモリ喰ってる。
並列数x1.5GBのメモリが必須。
javaプログラムなのでCPUはフルに8スレッド使ってくれてる。

real　102m50.419s
user　344m53.480s
sys　11m38.870s
9GBというのは、1.5GBフルに使うデカブツjar->dex変換が6個
並列するタイミングがあるということか。