



## 第 20 回 東京エリア Debian 勉強会 事前資料

Debian 勉強会会場係 上川純一\*

2006 年 9 月 16 日

---

\* Debian Project Official Developer

# 目次

1	Introduction To Debian 勉強会	2
1.1	講師紹介	2
1.2	事前課題紹介	2
2	Debian Weekly News trivia quiz	3
2.1	2006 年 XX 号	3
3	最近の Debian 関連のミーティング報告	4
3.1	東京エリア Debian 勉強会 19 回目報告	4
3.2	Lightweight Language Ring	5
4	あなたが知らないうちに使っている Debian specific	6
4.1	はじめに	6
4.2	おわりに	6
5	翻訳へのさそい	7
5.1	po 関連	7
5.2	debconf-po 関連	7
5.3	Debian JP WWW メーリングリスト	7
5.4	参考文献	7
6	apt/dpkg のプロファイリング	8
6.1	oprofie のインストールと設定方法	8
6.2	oprofile が自分の利用している CPU をサポートしていない場合	8
6.3	デバッグシンボルを収集する : dpkg と apt をコンパイルしなおす	9
6.4	テスト環境の作成	11
6.5	最適化の必要な部分の解析	11
6.6	最適化例	11
6.7	結果の検証	11
6.8	回帰テストの貢献	12
6.9	修正をフィードバック	12
6.10	参考文献	12
7	次回	13

# 1 Introduction To Debian 勉強会

上川純一

今月の Debian 勉強会へようこそ。これから Debian のあやしい世界に入るという方も、すでにどっぴりとつかっているという方も、月に一回 Debian について語りませんか？

目的として下記の二つを考えています。

- メールではよみとれない、もしくはよみとってられないような情報を情報共有する場をつくる
- まとまっていない Debian を利用する際の情報をまとめて、ある程度の塊として出してみる

また、東京には Linux の勉強会はたくさんありますので、Debian に限定した勉強会にします。Linux の基本的な利用方法などが知りたい方は、他でがんばってください。Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりと作りながらスーパーハッカーになれるような姿を妄想しています。

Debian をこれからどうするという能動的な展開への土台としての空間を提供し、情報の共有をしたい、というのが目的です。今回は違うこと言ってるかもしれませんが、御容赦を。

## 1.1 講師紹介

- 上川純一 宴会の幹事です。

## 1.2 事前課題紹介

今回の事前課題は「XXX」というタイトルで 200-800 文字程度の文章を書いてください。というものでした。その課題に対して下記の内容を提出いただきました。

### 1.2.1

### 1.2.2 上川

## 2 Debian Weekly News trivia quiz

上川純一

ところで、Debian Weekly News (DWN) は読んでいますか？Debian 界隈でおきていることについて書いている Debian Weekly News. 毎回読んでいるといろいろと分かって来ますが、一人で読んでいても、解説が少ないので、意味がわからないところもあるかも知れません。みんなで DWN を読んでみましょう。

漫然と読むだけではおもしろくないので、DWN の記事から出題した以下の質問にこたえてみてください。後で内容は解説します。

### 2.1 2006 年 XX 号

<http://www.debian.org/News/weekly/2006/XX/> にある 5 月 XX 日版です。

問題 1.

- A
- B
- C

問題 2.

- A
- B
- C

問題 3.

- A
- B
- C

問題 4.

- A
- B
- C

## 3 最近の Debian 関連のミーティング報告

上川純一

### 3.1 東京エリア Debian 勉強会 19 回目報告

東京エリア Debian 勉強会報告。8 月の第 19 回 Debian 勉強会を実施しました。今回は岩松さんが Debian Conference 開催進捗報告をしました。また、Lightning talk を開催しました。

今回の参加人数は 17 人でした。

参加者は gotom さん、山下さん、山根さん (中央線の事故の影響で結局宴会の最後になって到着)、岩井さん、南谷さん、前田耕平さん、たかやさん、きたはらさん、岩松さん、あけどさん、平川さん、小林さん、えとーさん、吉田@板橋さん、みつかさん、野首さん、上川です。

まず、Debian のソフトウェアに関するポリシーである、Social Contract の内容を確認しました。

異様な盛り上がりを見せながらユーザの声を紹介。事前課題をねたにしました。たかやさんが howm の Debian パッケージをオフィシャルパッケージにすることなので、期待です。

Debian Weekly News Quiz は今回は正解者には Debian シールをプレゼントしました。問題数を少なめに、早押しならぬ、早手あげで競争してもらいました。

岩松さんが北海道で下見をしてきた内容を踏まえて Debconf in Japan の検討事項について報告しました。東京や大阪など国際的にトランジットの利便のよい場所がよいですね、という話などが出ました。

Lightning talks の一発目は野首さん。プロフェッショナルな感じで、しっかりと執念を感じさせてくれる話でした、ありがとうございます。

吉田さんには、大衆向け IPv6 サービスを利用してみた話をしていただきました。まだいろいろと問題があるようで、すぐに使えるのかなあ、という印象はうけましたが、今後オフィシャルパッケージにマージされたりするとより利用しやすくなるでしょうから、今後の活躍に期待です。

Henrich さんは鉄道事故の関係で間に合いませんでした。残念。

山下さんには誕生日に思うことに付いて語っていただきました。淡々と語っていただきありがとうございます。

上川が module-assistant の使いかたについて話しました。module-assistant は意外にもつかっている人が少なく、話をしがいがありません。kernel module は必要な場合もあるので、ぜひパッケージできるようにしておくといえます。特に、module-assistant は使う側からすると従来の手法より非常に使い易くなっているのでよいよ、という話でした。

上川が最後に「board@jp お仕事日記」として発表しました。ここにかけるような内容ではないので、割愛。

今後のイベントにどのような参加方法をとるかということを検討しました。OSC-Fall にはブースも出して、いろいろなデバイスをハックしている姿を展示しましょう、ということになりました。また、OSC の沖縄には岩松さんとわたしは参加する、ということになりました。さらなるメンバー募集します。KOF については 4 人くらい参加すると宣言していたので、参加することになりそうです。

宴会は「土間土間」にて開催。けっこうよい場所を独占できたので、よい感じでした。

### 3.2 Lightweight Language Ring

上川が、軽量言語の祭典、LLRING に参加してきました。

発表した内容は、real kernel shell についてです。C 言語をライトウェイトに使いましょう、という話題です。

Haskell が異様にはやっていたのと、ウェブフレームワークでがりがりとやっているかたがたを横目にカーネルの話をしてきました。



## 4 あなたが知らないうちに使っている Debian specific

澤田さん

---

### 4.1 はじめに

### 4.2 おわりに

## 5 翻訳へのさそい

小林さん

### 5.1 po 関連

現状の ML についての Web は、<http://www.debian.or.jp/MailingList.html> です。

この中で翻訳関連を行なっているのは Debian JP Documentation メーリングリストです。

`debian-doc-ctl@debian.or.jp` に `fml` の方式で subscribe してください。過去記事は <http://lists.debian.or.jp/debian-doc/> にアクセスすると見ることができますので、ご参考にどうぞ。こちらは、`man`、`debconf-po`、`po`、および 付属ドキュメントなどの翻訳を行なっています。

Debian の `po` の各国のランキングです。 <http://www.debian.org/international/l10n/po/rank>

### 5.2 debconf-po 関連

Debian のパッケージインストールの際に `debconf` から質問されます。その質問の文字列を国際化するのが `debconf-po` です。

Debian の `debconf-po` の各国のランキングです。 <http://www.debian.org/international/l10n/po-debconf/rank>

ちなみに `debconf-po` についての国内での作業は作業がかぶらないように <http://kmuto.jp/debian/po-trans/> を使いながらやるとよいと思われます。

### 5.3 Debian JP WWW メーリングリスト

`debian-www-ctl@debian.or.jp` に `fml` の方式で subscribe してください。過去記事は <http://lists.debian.or.jp/debian-www/> にアクセスすると見ることができますので、ご参考にどうぞ。

こちらは、主に `debian.org` の web サイトの翻訳および、`debian.or.jp` の web サイトについて活動しています。

以下私見ですなにかの参考になりましたら。偉そうかもしれませんが、ご容赦を、、、

おそらくは、`debian-doc` などで活動を行なうと `ubuntu` のほうにも当然のように波及しますので、こちらの作業を行なうのもよいかなと思いました。

`ddtp` というプロジェクトで以前パッケージディスクリプション部分 (本日もう昨日か、見せていただいたアプリケーション一覧の説明部分とか?) を翻訳しているプロジェクトがありましたが、一旦停止しています。近々復帰しそうですので、DWN (Debian Weekly News) などを注視していただくとそのうちなんか出てくるかもです。

Debian プロジェクトはあなたの参加を心待ちにしております。でわでわ

### 5.4 参考文献

- ドキュメント翻訳手順 <http://kmuto.jp/d/index.cgi/debian/debian-doc-procedure.htm>



## 6 apt/dpkg のプロファイリング

上川

apt や dpkg のどの部分が一番遅いのか、実際にプロファイリングしてみます。この例をケーススタディーとして一般的にどういう作業をすればパフォーマンスチューニングができるのか、をあきらかにしてみましょう。

### 6.1 oprofile のインストールと設定方法

Debian のデフォルトのカーネルは oprofile をサポートしています<sup>\*1</sup>。もし、自分でコンパイルしていたりして oprofile サポートを追加していない場合は、カーネルを oprofile サポート付きでコンパイルしなおします。オプションは CONFIG\_OPROFILE です。メニューでは

Intrumentation support : Profiling Support : Oprofile system profiling (experimental)<sup>\*2</sup> にあります。

カーネルがサポートしている場合、oprofile を利用するのに追加で必要なのは oprofile パッケージです。apt-get install oprofile でインストールしましょう。

カーネルのシンボルのプロファイリング<sup>\*3</sup>をするために、vmlinux ファイルが必要です。カーネルを自分でコンパイルした場合には、ビルドしたディレクトリに vmlinux ファイルがあります。make-kpkg を利用してビルドしたのであれば、

```
/lib/modules/$(uname -r)/build
```

から適切にリンクがはられているはずです。探してみてください。<sup>\*4</sup>

### 6.2 oprofile が自分の利用している CPU をサポートしていない場合

残念ながら 7 月現在時点で、Intel core duo CPU 上では oprofile が動作しません。oprofile は認識できていない場合、cpu\_type 変数が unset という値になります。カーネル側は cpu\_type として i386/core を出力しているので、この時点でどうやらカーネル側のサポートは追加されているらしいということがわかります。

```
$ sudo opcontrol --init
cpu_type 'unset' is not valid
$ opcontrol --list-events
Unable to open cpu_type file for reading
Make sure you have done opcontrol --init
cpu_type 'unset' is not valid
$ cat /dev/oprofile/cpu_type
i386/core
$ uname -a
Linux coreduo 2.6.18-rc1dancer #2 SMP Sun Jul 9 09:57:01 JST 2006 i686 GNU/Linux
```

プロファイルを取得するという目的を考えると手段としてはいくつか考えられます。

- プロファイルの仕組みはあまりかわらないだろうと見込み、arch/i386/oprofile/nmi-int.c の ppro-init を

<sup>\*1</sup> i386, amd64 などのアーキテクチャ以外での利用は現時点では難しい可能性があるので確認してください。

<sup>\*2</sup> 2.6.18-rc1 現在

<sup>\*3</sup> 無い場合はカーネルの内部のどこかで実行していることはわかるが、実際どの関数で時間がかかっているのか、ということがわからない

<sup>\*4</sup> vmlinux ファイルよりは普及している System.map を利用するパッチというのも存在するので、それを適用してみるのもよいかもしれません。

修正、piii とかに見せてしまう

- まじめに oprofile のユーザ空間アプリケーションを修正、core duo の仕様書を読み、対応を追加
- おそらくすでに修正されていることを見越して、oprofile の CVS レポジトリをみにいく
- 実験することが目的なのでサポートされている CPU のマシンを準備する

今回は oprofile の開発メーリングリストを見たところ、5 月の時点でだれかがパッチを書いているのを発見したので、それを取りこみます。念のため、今後作業する人のために BTS にも登録しました。380462<sup>\*5</sup>

確認してみると、どうやら動作してくれてそうです。ここで、当面重要なのは、CPU\_CLK\_UNHALTED でしょう。CPU サイクルがどの関数で消費されているのかということをトラッキングできます。まず CPU の処理がかたまっている部分を目指して、何か問題がないかを眺めてみて、何も問題なく、それなりに問題が追求できにくくなった後に、L2 キャッシュのイベントの発生度合とかを確認していけばよいでしょう。

```
$ sudo opcontrol --init
$ sudo opcontrol --list-events
oprofile: available events for CPU type "Core Solo / Duo"

See Intel Architecture Developer's Manual Volume 3, Appendix A and
Intel Architecture Optimization Reference Manual (730795-001)

CPU_CLK_UNHALTED: (counter: all)
  Unhalted clock cycles (min count: 6000)
  Unit masks (default 0x0)
  -----
  0x00: Unhalted core cycles
  0x01: Unhalted bus cycles
  0x02: Unhalted bus cycles of this core while the other core is halted
INST_RETIRED: (counter: all)
  number of instructions retired (min count: 6000)
L2_RQSTS: (counter: all)
  number of L2 requests (min count: 6000)
  Unit masks (default 0xf)
  -----
  0x08: (M)odified cache state
  0x04: (E)xclusive cache state
  0x02: (S)hared cache state
  0x01: (I)nvalid cache state
  0x0f: All cache states
  0x10: HW prefetched line only
  0x20: all prefetched line w/o regarding mask 0x10.

[省略]
```

### 6.3 デバッグシンボルを収集する：dpkg と apt をコンパイルしなおす

まず、デバッグ情報がすでにあるパッケージについては、インストールします。今回では、大きいものとして、libc6-dbg パッケージがあるので、それはインストールします。プロファイルの結果、上位に出現するなど、必要そうであれば、あとでデバッグ情報のあるバージョンを追加します。

今回プロファイル対象の dpkg と apt はデフォルトではデバッグ情報がありません、プロファイル出力を確認しやすいように、デバッグシンボルを追加してコンパイルしなおします。

```
$ debuild -e DEB_BUILD_OPTIONS=nostrip
```

その後、インストールします。

まず、oprofile を実行するのを便利にするために、スクリプトを仕込みます。入力されたコマンドを 10 回実行してそのプロファイルを取得するというものです。

```
read CMD
sudo opcontrol --shutdown
sudo opcontrol --reset
sudo opcontrol --setup \
  --vmlinux=/lib/modules/$(uname -r)/build/vmlinux \
  --event=CPU_CLK_UNHALTED:180000:0:1:1 --separate=library
sudo opcontrol --start
for A in $(seq 1 10); do
  $CMD
done
opcontrol --dump && \
opreport -l -p /lib/modules/$(uname -r)/kernel 2>/dev/null \
| head -30
```

まず、デバッグ用のバイナリが正常に作成できているか簡単に確認します。まず、apt-get update をループでまわ

<sup>\*5</sup> <http://bugs.debian.org/380462>

してみます。libapt-pkg のシンボルレベルで確認できているので、デバッグシンボルが存在しているということがわかります。

```
sudo apt-get update
[中略]
CPU: Core Solo / Duo, speed 1833 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Unhalted clock cycles) with a unit mask of 0x00 (Unhalted core cycles) count 180000
samples %      image name      app name      symbol name
23823    46.3519  libapt-pkg-libc6.3-6.so.3.11.0 apt-get
SHA1Transform(unsigned int*, unsigned char const*)
12732    24.7724  libapt-pkg-libc6.3-6.so.3.11.0 apt-get
MD5Transform(unsigned int*, unsigned int const*)
4282     8.3314   processor.ko    processor      acpi_processor_idle
2584     5.0276   libc-2.3.6.so   apt-get        (no symbols)
2012     3.9147   vmlinux         vmlinux        __copy_to_user_ll
503      0.9787   gpgv           gpgv          (no symbols)
222      0.4319   vmlinux         vmlinux        timer_interrupt
166      0.3230   libapt-pkg-libc6.3-6.so.3.11.0 apt-get
MD5Summation::Add(unsigned char const*, unsigned long)
160      0.3113   vmlinux         vmlinux        page_fault
158      0.3074   libapt-pkg-libc6.3-6.so.3.11.0 apt-get
SHA1Summation::Add(unsigned char const*, unsigned long)
140      0.2724   libstdc++.so.6.0.8 apt-get        (no symbols)
134      0.2607   vmlinux         vmlinux        find_get_page
125      0.2432   ld-2.3.6.so     http           do_lookup_x
123      0.2393   vmlinux         vmlinux        sysenter_past_esp
95       0.1848   libapt-pkg-libc6.3-6.so.3.11.0 apt-get        .plt
94       0.1829   ld-2.3.6.so     gpgv           do_lookup_x
89       0.1732   libc-2.3.6.so   http           (no symbols)
89       0.1732   vmlinux         vmlinux        do_generic_mapping_read
88       0.1712   ld-2.3.6.so     file           do_lookup_x
87       0.1693   vmlinux         vmlinux        memcpy
72       0.1401   ld-2.3.6.so     http           strcmp
69       0.1343   vmlinux         vmlinux        _spin_lock
65       0.1265   vmlinux         vmlinux        vfs_read
64       0.1245   ld-2.3.6.so     gpgv           _dl_elf_hash
62       0.1206   vmlinux         vmlinux        __handle_mm_fault
60       0.1167   ld-2.3.6.so     http           _dl_elf_hash
58       0.1128   oprofiled       oprofiled      (no symbols)
```

dpkg についてもプロファイリングしてみます。<sup>\*6</sup>

```
sudo dpkg -i ../dselect_1.13.22_i386.deb
[中略]
CPU: Core Solo / Duo, speed 1833 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Unhalted clock cycles) with a unit mask of 0x00 (Unhalted core cycles) count 180000
samples %      image name      app name      symbol name
41009    32.6009   libc-2.3.6.so   dpkg           (no symbols)
27485    21.8497   processor.ko    processor      acpi_processor_idle
14863    11.8156   dpkg            dpkg           parsedb
4845     3.8516   dpkg            dpkg           findnamenode
2691     2.1393   dpkg            dpkg           findpackage
1994     1.5852   dpkg            dpkg           f_dependency
1742     1.3848   vmlinux         vmlinux        get_page_from_freelist
1660     1.3196   dpkg-deb        dpkg-deb       inflate_fast
1552     1.2338   dpkg            dpkg           iterpkgnext
1520     1.2084   dpkg            dpkg           .plt
1500     1.1925   dpkg            dpkg           varbufaddbuf
1192     0.9476   dpkg            dpkg           filesdbinit
1080     0.8586   dpkg            dpkg           w_dependency
1001     0.7958   dpkg            dpkg           varbufdependency
988      0.7854   dpkg            dpkg           nfmalloc
884      0.7028   dpkg            dpkg           illegal_package_name
849      0.6749   vmlinux         vmlinux        page_fault
802      0.6376   vmlinux         vmlinux        __copy_from_user_ll_nocache_nzero
687      0.5461   dpkg            dpkg           ensure_package_files_available
633      0.5032   dpkg            dpkg           varbufaddc
568      0.4515   dpkg            dpkg           f_filecharf
516      0.4102   vmlinux         vmlinux        __copy_to_user_ll
473      0.3760   dpkg            dpkg           copy_dependency_links
458      0.3641   dpkg            dpkg           parseversion
427      0.3395   dpkg            dpkg           ensure_package_clientdata
406      0.3228   dpkg            dpkg           nfstrsave
384      0.3053   dpkg            dpkg           varbufrecord
```

パッケージをインストールして削除する、というループを回してみましょう。apt-listbugs と apt-listchanges が含まれており、ruby と python の処理負荷が高いことがわかります。また、libc6 のなかで何か重たい処理をしているのがわかります。

<sup>\*6</sup> ここで問題が出ました。libc6 の dbg パッケージの情報を oprofile が処理できていないようです。strace で解析してみましたが、ファイルを開くところまでは何かできているようです。これは別途バグ報告してみます。385704<sup>\*7</sup>

```

sudo apt-get install -y dsh; sudo apt-get remove -y libdshconfig1
[中略]
Counted CPU_CLK_UNHALTED events (Unhalted clock cycles) with a unit mask of 0x00 (Unhalted core cycles) count 180000
samples % image name app name symbol name
195383 24.3783 processor processor (no symbols)
114285 14.2595 libc-2.3.6.so dpkg (no symbols)
67157 8.3793 libruby1.8.so.1.8.4 ruby1.8 (no symbols)
48893 6.1005 libc-2.3.6.so dpkg-query (no symbols)
41537 5.1826 dpkg dpkg parsedb
30685 3.8286 perl perl (no symbols)
28353 3.5377 dpkg-query dpkg-query parsedb
26135 3.2609 python2.4 python2.4 (no symbols)
13951 1.7407 libc-2.3.6.so ruby1.8 (no symbols)
10023 1.2506 libc-2.3.6.so apt-get (no symbols)
9138 1.1402 dpkg dpkg findnamenode
7914 0.9874 vmlinux vmlinux get_page_from_freelist
7656 0.9553 dpkg dpkg findpackage
5963 0.7440 vmlinux vmlinux read_hpet
5777 0.7208 libc-2.3.6.so perl (no symbols)
5465 0.6819 dpkg dpkg f_dependency
5108 0.6373 dpkg-query dpkg-query findpackage
4525 0.5646 dpkg dpkg filesdbinit
4452 0.5555 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
pkgDepCache::CheckDep(pkgCache::DepIterator, int,
pkgCache::PkgIterator&)
4237 0.5287 vmlinux vmlinux page_fault
4182 0.5218 dpkg dpkg .plt
4101 0.5117 dpkg dpkg varbufaddbuf
3874 0.4834 dpkg-query dpkg-query f_dependency
3744 0.4671 dpkg dpkg iterpkgnext
3645 0.4548 libstdc++.so.6.0.8 apt-get (no symbols)
3287 0.4101 libapt-pkg-libc6.3-6.so.3.11.0 apt-get
pkgProblemResolver::MakeScores()
3277 0.4089 vmlinux vmlinux delay_tsc

```

## 6.4 テスト環境の作成

テスト用の環境を作成します。今回は chroot 内部で大量の apt-get update, apt-get install と apt-get remove をループで実行してベンチマークをとってみましょう。

dpkg と apt を変更すると最悪システムが動作しなくなるため、テスト用に環境を準備することは大切です。

chroot 内部では、chroot 外部のファイルにアクセスすることができません。そのため、bind-mount を行い、外部のファイルを見せます。中で必要になるファイルとしては、apt/dpkg のデバッグ版、oprofile の修正版パッケージ（あれば）そして実行中の Linux kernel に対応する vmlinux ファイルです。

```

$ sudo cowbuilder --login --bindmount $(pwd)
# apt-get install gnupg
# apt-get update
# apt-get install oprofile libc6-dbg
# dpkg -i (bind-mount したところにおいた事前準備した apt/dpkg/oprofile)
# apt-get -y install gnome; apt-get -y remove libglib2.0-0
# unset LD_PRELOAD
# unset COWDANCER_ILISTFILE
# mount -t oprofilefs nodev /dev/oprofile >/dev/null

```

chroot で調べてみると、perl が一番重い処理をしているということがわかりました。こりゃチューニングしにくいですね。依存関係の解決などの処理に時間がかかっているかと仮説をたてていたのですが、実はそんなところでは時間はとられておらず、perl が時間をとっているということがわかりました。

```

apt-get install -y dsh; apt-get remove -y libdshconfig1
[中略]
CPU: Core Solo / Duo, speed 1833 MHz (estimated)
Counted CPU_CLK_UNHALTED events (Unhalted clock cycles) with a unit mask of 0x00 (Unhalted core cycles) count 180000
samples %      image name      app name      symbol name
48879    32.1023  processor      processor     (no symbols)
20915    13.7364    perl          perl         (no symbols)
7787     5.1143    libc-2.3.6.so apt-get      (no symbols)
6775     4.4496    libc-2.3.6.so dpkg         (no symbols)
5198     3.4139    vmlinux       vmlinux      read_hpet
4202     2.7598    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
pkgDepCache::CheckDep(pkgCache::DepIterator, int,
pkgCache::PkgIterator&)
4167     2.7368    libc-2.3.6.so perl         (no symbols)
2778     1.8245    libstdc++.so.6.0.8 apt-get      (no symbols)
2587     1.6991    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
debVersioningSystem::CmpFragment(char const*, char const*, char const*,
char const*)
2561     1.6820    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
pkgProblemResolver::MakeScores()
2103     1.3812    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
debVersioningSystem::DoCmpVersion(char const*, char const*, char
const*, char const*)
1865     1.2249    dpkg         dpkg         parsedb
1669     1.0962    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
pkgDepCache::Update(OpProgress*)
1506     0.9891    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
debVersioningSystem::CheckDep(char const*, int, char const*)
1384     0.9090    dpkg         dpkg         findnamenode
1211     0.7954    vmlinux       vmlinux      __copy_to_user_ll
1148     0.7540    vmlinux       vmlinux      get_page_from_freelist
1107     0.7270    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
OpProgress::CheckChange(float)
955      0.6272    vmlinux       vmlinux      page_fault
912      0.5990    ldconfig      ldconfig     (no symbols)
872      0.5727    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
pkgPolicy::GetCandidateVer(pkgCache::PkgIterator)
826      0.5425    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
pkgDepCache::DependencyState(pkgCache::DepIterator&)
787      0.5169    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
pkgProblemResolver::ScoreSort(void const*, void const*)
752      0.4939    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
748      0.4913    dpkg         dpkg         .plt
680      0.4466    vmlinux       vmlinux      filesdbinit
679      0.4459    libapt-pkg-libc6.3-6.so.3.11.0 apt-get
pkgCache::PkgIterator::operator++(int)

```

## 6.5 最適化の必要な部分の解析

プロファイル結果を利用して、解析します。

## 6.6 最適化例

今回の結果で適用できる最適化を分析します。

## 6.7 結果の検証

さきほどのベンチマークを利用して、状況が改善していることを確認します。

## 6.8 回帰テストの貢献

できるようであれば、今回の修正が二度と必要ないように、パフォーマンスのデグレードがすぐに検出しやすいよう、再現しやすいテストケースを追加してみるのもよいでしょう。

## 6.9 修正をフィードバック

パフォーマンスの改善のための修正をフィードバックします。Debian の場合、BTS にパッチを登録します。該当するバグ番号は下記です。

- XXX
- YYY

## 6.10 参考文献

- rpm のプロファイリング <https://www.redhat.com/magazine/012oct05/features/oprofile/>



未定です。内容は本日決定予定です。  
参加者募集はまた後程。



Debian 勉強会資料

2006 年 9 月 16 日 初版第 1 刷発行

東京エリア Debian 勉強会（編集・印刷・発行）

---