FreeBSD勉強会

第1回 システム/カーネル構築の高速化 Tips

FreeBSD勉強会実行委員
後藤大地

勉強会 基本情報

本日のスピーカプロフィール

- 後藤大地 (GOTO Daichi) 1980- (日本)
  - o LinkedIn http://www.linkedin.com/in/daichigoto
  - o Twitter http://twitter.com/daichigoto
  - o Wassr http://wassr.jp/user/daichigoto
- オングス 代表取締役 2002-

  Homepage http://www.ongs.co.jp
  情報システム構築、情報系ニュース執筆、雑誌や書籍の企画立案および執筆、
  情報コンサルティング、保守 etc etc
  最近の後藤の活動
  - o FreeBSD Daily Topics http://gihyo.jp/admin/clip/01/fdt
  - o MJ Enterprise http://journal.mycom.co.jp/enterprise/
- FreeBSD src / ports committer 2002-

  Homepage http://people.freebsd.org/~daichi
  最近の活動
  Unionfs再実装、USB2関連のバグパッチ etc etc

FreeBSD勉強会実行委員

- 佐藤広生

  東京理科大学／FreeBSD Foundation
- 後藤大地

  オングス代表取締役／FreeBSD committer
- 馮富久

  技術評論社 クロスメディア事業部

- 問い合わせ: freebsd@gihyo.co.jp

http://gihyo.jp/event/01/freebsd

gihyo.jp > イベント > FreeBSD勉強会

gihyo.jp FreeBSD勉強会

5月21日(木) 第1回 開催!

概 要 | お申し込み | レポート | 資 料 | 実行委員

技術評論社では，過去，『Software Design』や『FreeBSD Expert』といった雑誌，最近の FreeBSD Daily Topics をはじめ，さまざまなメディアでFreeBSDを取り上げ，多くのコミッタの方にご協力いただいております。

今回，紙・Webといったメディアを超えて，リアルなコミュニケーションの場として，定期的な勉強会を開催することになりました。

本サイトでは，勉強会の開催情報や事後レポートなど，開発者とユーザのコミュニケーションを活性化するためのさまざまな情報を発信していきます。

ニュース

2009/5/13
gihyo.jp FreeBSD勉強会 第1回のご案内
2009/4/16
第0回gihyo.jp FreeBSD勉強会レポート公開しました
2009/4/10
gihyo.jp FreeBSD勉強会 第0回のご案内
2009/4/10
本サイトをオープンしました。

対象

FreeBSDに興味がある方，FreeBSDを活用したいと考えている，または活用なさっている方などが対象です。学生，社会人かどうかは問いません。

また，FreeBSDを活用なさっている企業関係者の方々が，新しいつながりを生み出す場としても機能させていきたいと考えています。ふるってご参加ください。

ご注意

本勉強会では，勉強会の模様を動画・写真として撮影する予定です。

撮影された動画・写真は勉強会サイト上にて公開する予定があるほかに，gihyo.jpをはじめ各種メディアにて報道・宣伝等のため使用される可能性があることをご承知おきください。

トラックバック

このエントリのトラックバック URI

開催スケジュール

第1回 システム／カーネル構築の高速化Tips（2009/5/21）
第0回 gihyo.jp FreeBSD勉強会設立のご案内（2009/4/15）

技術評論社イベントスケジュール

FreeBSD勉強会 第1回（2009/5/21）
エンジニアの未来サミット 0905（2009/5/23）
FreeBSD勉強会 第0回（2009/4/15）
第1回 Webエンジニア料理対決（2009/3/27）
使う人にやさしいMovable Typeを考える（2009/3/26）
戦略的Webマーケティングセミナー（2009/3/18）
第2回ソフトウェアテストセミナー（2009/3/11）
組込みプレスセミナー（2009/3/10）
ソフトウェアテストセミナー（2008/9/19）
エンジニアの未来サミット（2008/9/13）
サーバ/インフラ Tech Meeting（2008/8/8）

gihyo.jp「FreeBSD」関連記事

2009年5月19日 VirtualBox for FreeBSD動作紹介，Python2.6へ移行，YouTube BSDチャンネルにTheo de Raadt氏特別講演登場，Dr.Web on FreeBSD/amd64

2009年5月18日 Silverlight対応のMoonshine登場，VirtualBox for FreeBSDの試験port登場，AsiaBSDCon2009論文公開，YouTube BSDチャンネル AsiaBSDCon2009基調講演公開

開催告知

- gihyo.jp http://gihyo.jp/
- FreeBSD Daily Topics
  http://gihyo.jp/admin/clip/01/fdt
- Twitter http://twitter.com/daichigoto
- Mixi FreeBSDコミュニティ http://c.mixi.jp/freebsd
- Google Calendar “FreeBSD勉強会”で検索

etc. etc.

システム・カーネルビルド

ビルド時間短縮の基本戦術案

- キャッシュの活用
- 並列ビルドの有効化
- メモリディスクの活用

キャッシュを有効にする

- キャッシュを有効にすることでビルド時間を短縮できる可能性がある。たとえばちょっとした変更がおこなわれただけでバージョンが更新されたシステムをビルドする場合などに機能することになるだろう
- /usr/ports/devel/ccache/ を使う
- PATHの先頭に /usr/local/libexec/ccache/ を追加するだけで設定自動的にキャッシュが有効になったビルドが実行されるようになる。

```
% ls -l /usr/local/libexec/ccache/
total 26
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 c++ -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 cc -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 g++ -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 g++-ooo -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 g++32 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 g++33 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 g++34 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 g++40 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 g++41 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 g++42 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 g++43 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 g++44 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 gcc -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 gcc-ooo -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 gcc32 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 gcc33 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 gcc34 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 gcc40 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 gcc41 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 gcc42 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 gcc43 -> /usr/local/bin/ccache
lrwxr-xr-x  1 root  wheel  21  2  1 09:35 gcc44 -> /usr/local/bin/ccache
-r-xr-xr-x  1 root  wheel  96  2  1 09:35 world-c++
-r-xr-xr-x  1 root  wheel  95  2  1 09:35 world-cc
%
```

キャッシュ情報

```
% ccache -s
cache directory                     /home/ccache
cache hit                          254213
cache miss                         158924
called for link                     27210
multiple source files                  20
compile failed                      3124
preprocessor error                  2625
not a C/C++ file                    8924
autoconf compile/link              29193
unsupported compiler option         2013
no input file                      11404
files in cache                     317848
cache size                           9.1 Gbytes
max cache size                      10.0 Gbytes
%
```

並列ビルド

- コアそのものの性能は以前ほどには時間とともには向上しない。これからはプロセッサに搭載するコアの数を増やすといったように並列化で性能の向上をはかっていく必要がある
- このため、ビルドの時間を短縮するには、複数のコアをいかに効率よく使うかが鍵になってくるだろう

システム/カーネルビルドの並列化

- make -jN でビルドをN個まで並列化して処理するように振る舞うようになる。ただしそれほど細かく並列化が実現できるものではない
- システム/カーネルのビルドは並列セーフ

Ports Collection ≠ 並列セーフ

- Ports Collection そのものは並列ビルドが安全に実施できるようには設計されていない。Ports Collection自体の並列ビルド対応はこれからの課題
- Ports Collectionは並列セーフではないが、登録されているアプリケーションは個別に対応していたり、対応していなくても並列ビルドが可能なものもある

OpenOffice.org の場合

- もっともビルド時間がかかる単体アプリケーション
  OpenOffice.org は独自に並列ビルドに対応している。
- OpenOffice.org 3系は次の変数で並列数を指定

```
MAXPROCESSES=並列数
MAXMODULES=並列数
```

- OpenOffice.org 2系は次の変数で並列数を指定

```
NUMOFPROCESSES=並列数
```

gmake / bsd make の場合

- 明示的に並列ビルドに対応していなくても、作りのいいアプリケーションは bsdmake / gmake の並列処理オプションで並列化が可能。
- make で指定すると Ports Collection の処理そのものが並列化されて失敗する
- 次のオプションを指定して、Ports Collection から呼び出されるビルド処理が並列化されるようにする

```
MAKE_ARGS+=-j8
```

MAKE_ARGS+=-j並列数 の指針

MAKE_ARGS+=-j並列数による並列化はすべての場合において使える場合ではない。対応しているアプリケーションを調査して個別に指定する必要がある

並列化が可能な場合には効果あり。特に大規模アプリケーションで有効。キャッシュと並列処理、メモリファイルシステムなどを組み合わせると、OpenOffice.orgであれば三十分以内のビルドも可能という報告もある

メモリディスクの活用

- FreeBSDにはtmpfs(5)によるメモリディスクと、mdconfig(8)/mdmfs(8)/mount_mfs(8)によるメモリディスクの2つの方法が提供されている
- tmpfs(5)はNetBSDから移植。手軽でより高速と謳われている
- ディスク入出力、とくにランダムデータアクセスやディスクへのデータ書き込みをメモリディスクで実施するようにすることでビルド時間の短縮が狙えるのではないか

tmpfs(5)

```
# mount -t tmpfs tmpfs /memdisk
# mount | grep /memdisk
tmpfs on /memdisk (tmpfs, local)
# df | grep /memdisk
tmpfs       8.6G    4.0K    8.6G      0%    /memdisk
#
```

md(4)

```
# mdmfs -s 10g md /memdisk
# mount | grep /memdisk
/dev/md6 on /memdisk (ufs, local, soft-updates)
#

# mdmfs -S -s 10g -o async,noatime md /memdisk
# mount | grep /memdisk
/dev/md5 on /memdisk (ufs, asynchronous, local, noatime)
#
```

/usr/src と /usr/obj

- システムおよびカーネルのソースコード /usr/src
- コンパイル時に生成されるファイルは /usr/obj に書き込まれる
- /usr/obj をメモリディスクにするか、環境変数MAKEOBJDIRPREFIXで指定したディレクトリをメモリディスクにしておけば、書き込み先にメモリが使われる（空き領域は1.7GBほど必要）
- メモリディスクは再起動すると消えるので、再起動してmake installworldするには、永続性のあるディスクへ最後に書き戻しておく必要がある

Ports Collection

- /usr/ports/カテゴリ名/アプリ名/work がビルド時の作業ディレクトリ
- 環境変数WRKDIRPREFIXを指定すると、そのディレクトリ以下に/usr/ports/カテゴリ名/アプリ名/workが展開されるようになるため、WRKDIRPREFIXで指定したディレクトリをメモリディスクに指定すればサードパーティアプリケーションのビルドにメモリディスクを使えるようになる

ベンチ環境

FreeBSD-current/amd64

% uname -a

FreeBSD parancell.ongs.co.jp 8.0-CURRENT FreeBSD 8.0-CURRENT #2: Wed May 20 00:03:27 JST 2009 root@parancell.ongs.co.jp:/usr/obj/usr/src/sys/PARANCELL amd64

%

CPU / Memory

```
% sysinfo cpu mem
CPU information

Machine class: amd64
CPU Model: Intel(R) Core(TM)2 Quad CPU    Q8300  @ 2.50GHz
No. of Cores:  4
Cores per CPU:

CPU usage statistics:
CPU:  3.5% user,  0.0% nice,  2.2% system,  0.2% interrupt, 94.2% idle

RAM information

Memory information from dmidecode(8)
WARNING: /dev/mem is not readable!
WARNING: Running /usr/local/share/sysinfo/modules/mem as an unprivileged
user may prevent some features from working.
System memory summary
Total real memory available:  4081 MB
Logically used memory:    2430 MB
Logically available memory:   1651 MB

Swap information
Device          512-blocks     Used    Avail Capacity
/dev/ad0s1b       16777216     3.7M     8.0G     0%
%
```

df(1)/mount(1)

```
% df
Filesystem                                          Size    Used   Avail Capacity  Mounted on
/dev/ad0s1a                                         1.9G    692M    1.1G    38%    /
devfs                                               1.0K    1.0K      0B   100%    /dev
tmpfs                                               8.6G     20K    8.6G     0%    /tmp
/dev/ad0s1f                                         4.4G    2.6G    1.5G    63%    /usr
/dev/ad0s1d                                         3.9G    195M    3.4G     5%    /var
tmpfs                                               8.6G    4.0K    8.6G     0%    /memdisk
procfs                                              4.0K    4.0K      0B   100%    /proc
linprocfs                                           4.0K    4.0K      0B   100%  /usr/compat/linux/proc
tank/ccache                                          48G     10G     38G    21%    /home/ccache
tank/usr/local                                       44G    6.6G     38G    15%    /usr/local
tank/usr/ports                                       41G    3.2G     38G     8%    /usr/ports
dacolm.ongs.co.jp:/usr/ongs/nobackup/nfshome/daichi 898G    259G    567G    31%    /home/daichi
dacolm.ongs.co.jp:/usr/ongs/nobackup/nfshome/sasaki 898G    259G    567G    31%    /home/sasaki
dacolm.ongs.co.jp:/usr/ongs/nobackup/nfshome/takasyou 898G  259G    567G    31%    /home/takasyou
dacolm.ongs.co.jp:/backup/portsdistfiles            271G    120G    129G    48%  /usr/ports/distfiles
dacolm.ongs.co.jp:/usr/ongs/backup/common           898G    259G    567G    31%
/netdisk/share-backupfull
dacolm.ongs.co.jp:/usr/ongs/nobackup/common         898G    259G    567G    31%   /netdisk/share-
backupless
% mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
tmpfs on /tmp (tmpfs, local)
/dev/ad0s1f on /usr (ufs, local, soft-updates)
/dev/ad0s1d on /var (ufs, local, soft-updates)
tmpfs on /memdisk (tmpfs, local)
procfs on /proc (procfs, local)
linprocfs on /usr/compat/linux/proc (linprocfs, local)
tank/ccache on /home/ccache (zfs, local)
tank/usr/local on /usr/local (zfs, local)
tank/usr/ports on /usr/ports (zfs, local)
dacolm.ongs.co.jp:/usr/ongs/nobackup/nfshome/daichi on /home/daichi (nfs)
dacolm.ongs.co.jp:/usr/ongs/nobackup/nfshome/sasaki on /home/sasaki (nfs)
dacolm.ongs.co.jp:/usr/ongs/nobackup/nfshome/takasyou on /home/takasyou (nfs)
dacolm.ongs.co.jp:/backup/portsdistfiles on /usr/ports/distfiles (nfs)
dacolm.ongs.co.jp:/usr/ongs/backup/common on /netdisk/share-backupfull (nfs)
dacolm.ongs.co.jp:/usr/ongs/nobackup/common on /netdisk/share-backupless (nfs)
```

ベンチマーク内容

- ノーマルビルド
- CCACHEビルド
- 並列ビルド (-j2 から -j8まで)
- CCACHE+並列ビルド (-j2 から -j8まで)
- メモリディスク活用 (tmpfs / md[soft-updates] / md [async,noatime])で上記4つの組み合わせ

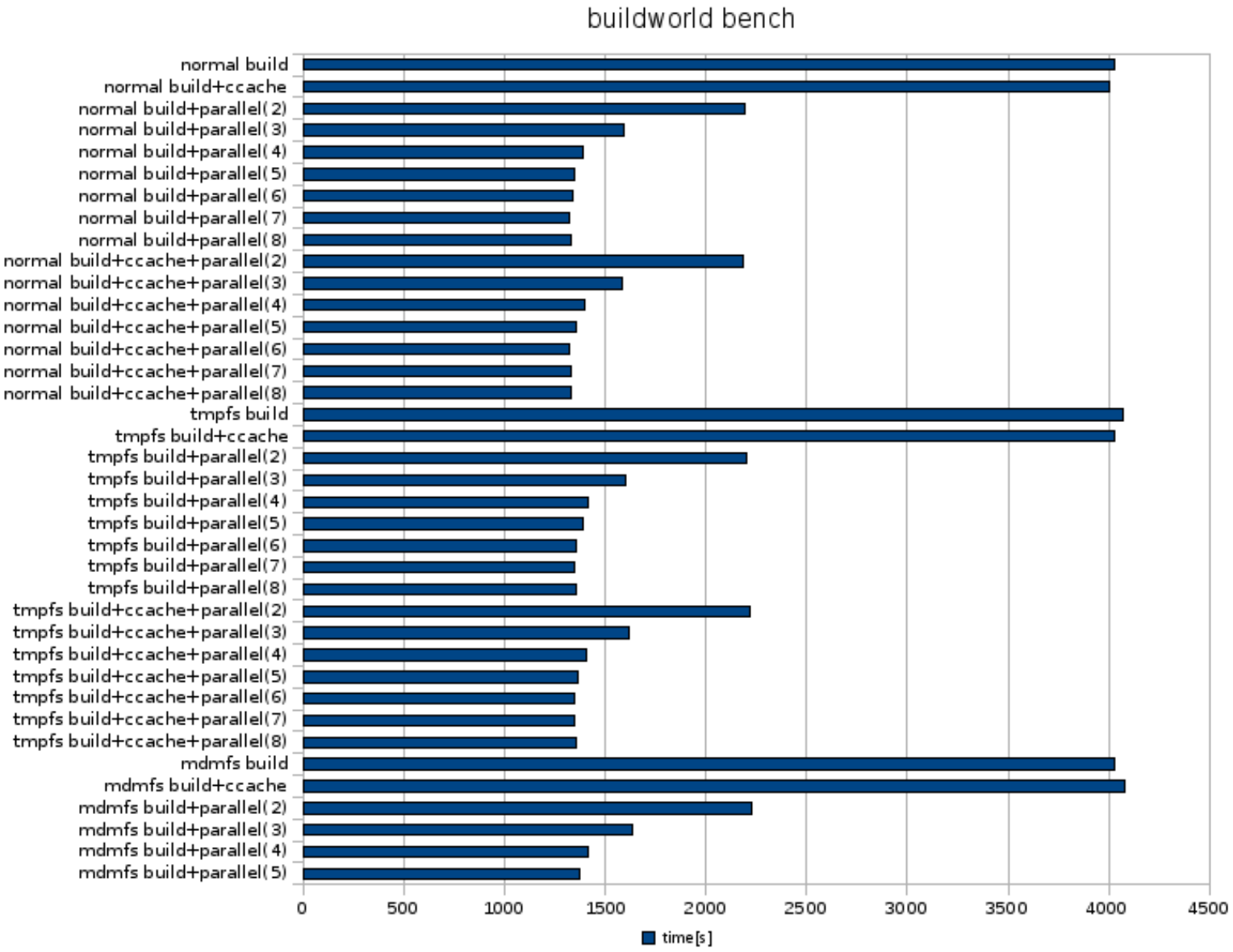
ベンチマークスクリプト

fetch http://www.ongs.net/~daichi/gihyo/001/buildbench.tgz

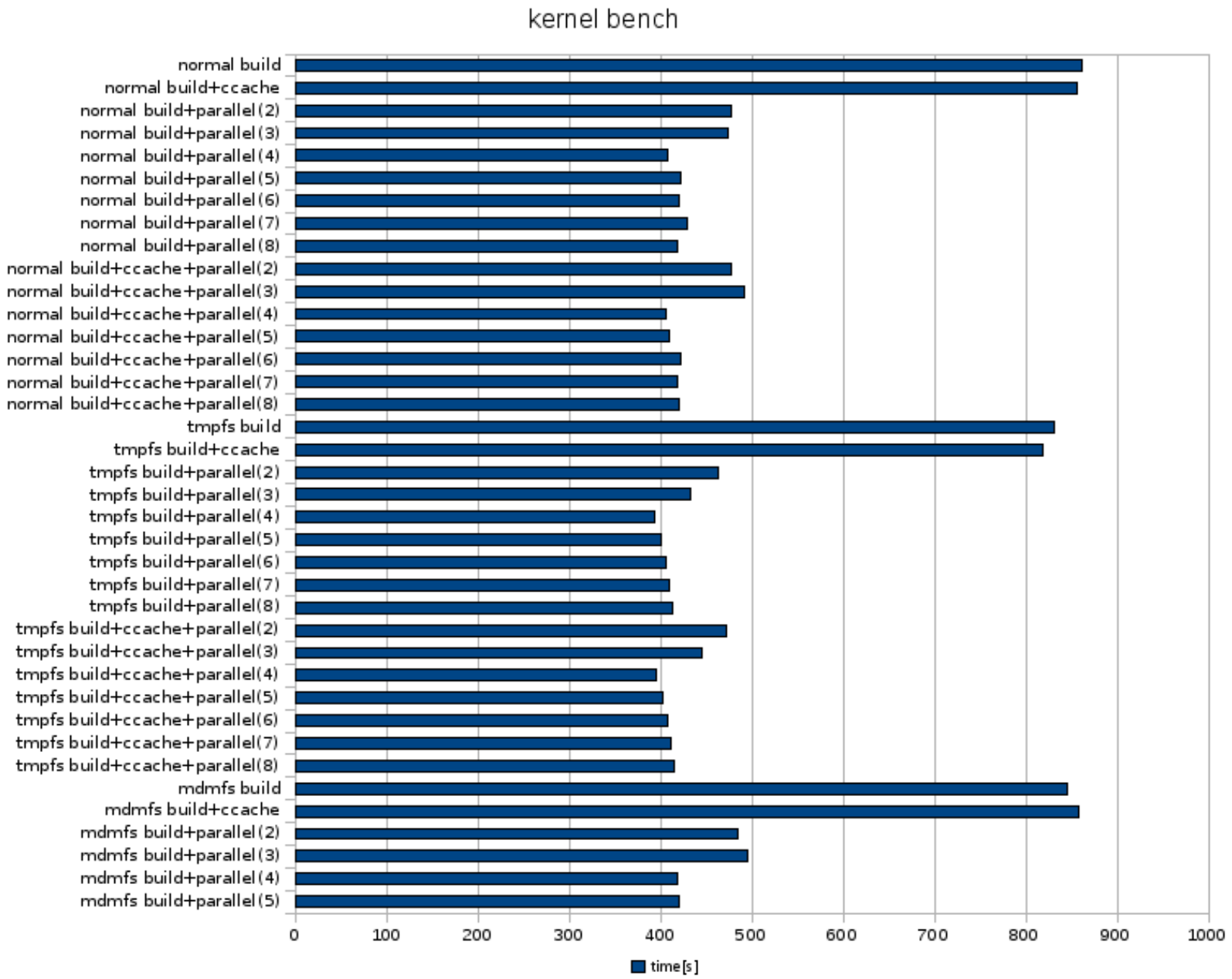
```
% tar zxvf buildbench.tgz x buildbench/
x buildbench/kernelBuildBench/
x buildbench/portsBuildBench/
x buildbench/portsBuildBench/result-firefox3
x buildbench/portsBuildBench/result-thunderbird
x buildbench/portsBuildBench/result-amarok2
x buildbench/portsBuildBench/result-tmpfs-mount
x buildbench/portsBuildBench/result-firefox3-tmpfs
x buildbench/portsBuildBench/result-thunderbird-tmpfs
x buildbench/portsBuildBench/result-amarok2-tmpfs
x buildbench/portsBuildBench/result-mdmfs-mount
x buildbench/portsBuildBench/result-firefox3-mdmfs
x buildbench/portsBuildBench/result-thunderbird-mdmfs
x buildbench/portsBuildBench/result-amarok2-mdmfs
x buildbench/portsBuildBench/bench.sh
x buildbench/portsBuildBench/result-mdmfs-mount-default
x buildbench/portsBuildBench/result-firefox3-mdmfs-default
x buildbench/portsBuildBench/result-thunderbird-mdmfs-default
x buildbench/portsBuildBench/result-amarok2-mdmfs-default
x buildbench/portsBuildBench/test.sh
x buildbench/kernelBuildBench/result-bench-mdmfs
x buildbench/kernelBuildBench/bench-tmpfs.sh
x buildbench/kernelBuildBench/test.sh
x buildbench/kernelBuildBench/bench-mdmfs.sh
x buildbench/kernelBuildBench/result-bench
x buildbench/kernelBuildBench/result-bench-tmpfs
x buildbench/kernelBuildBench/bench.sh
%
```

ベンチマーク結果

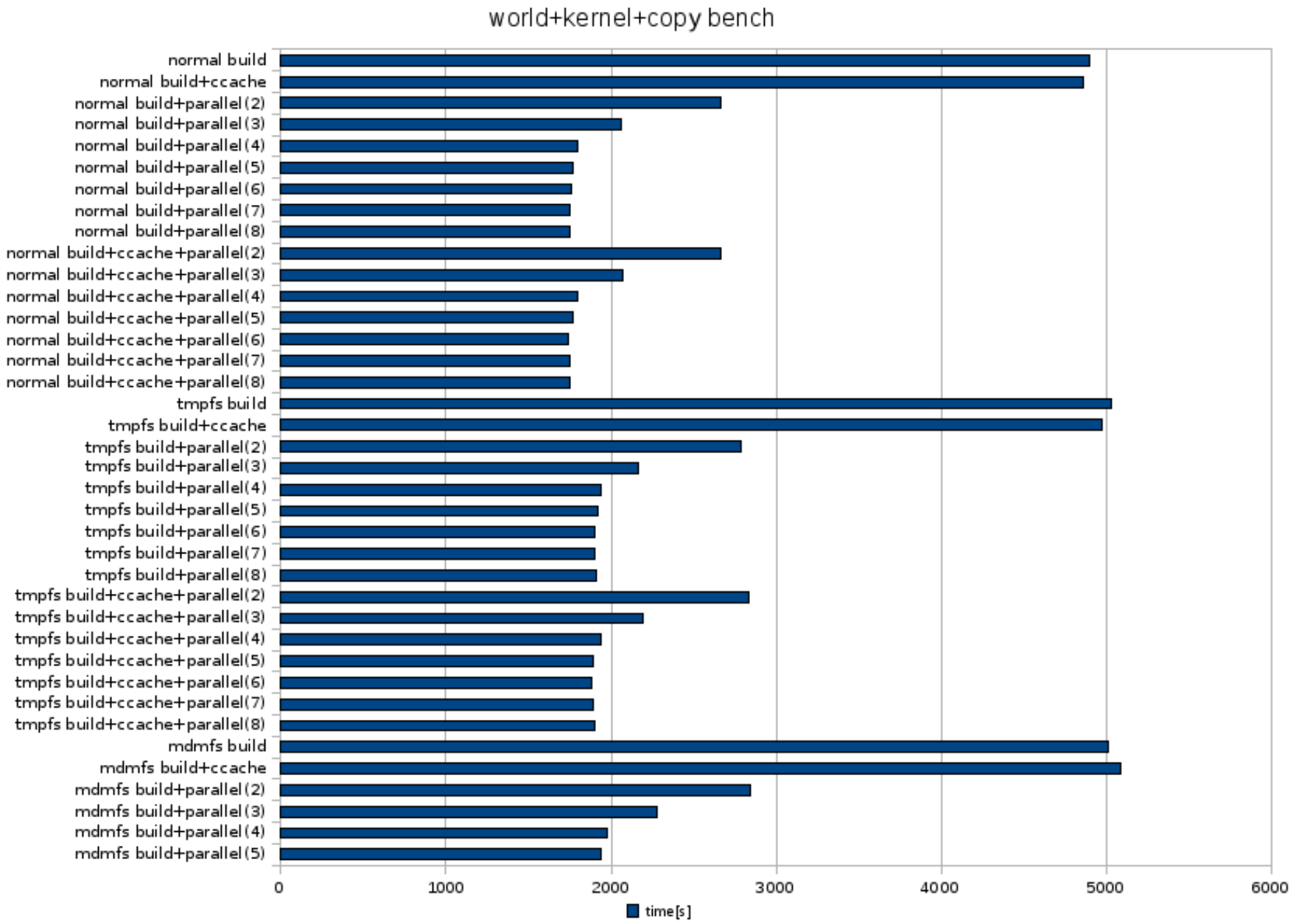
buildworld bench



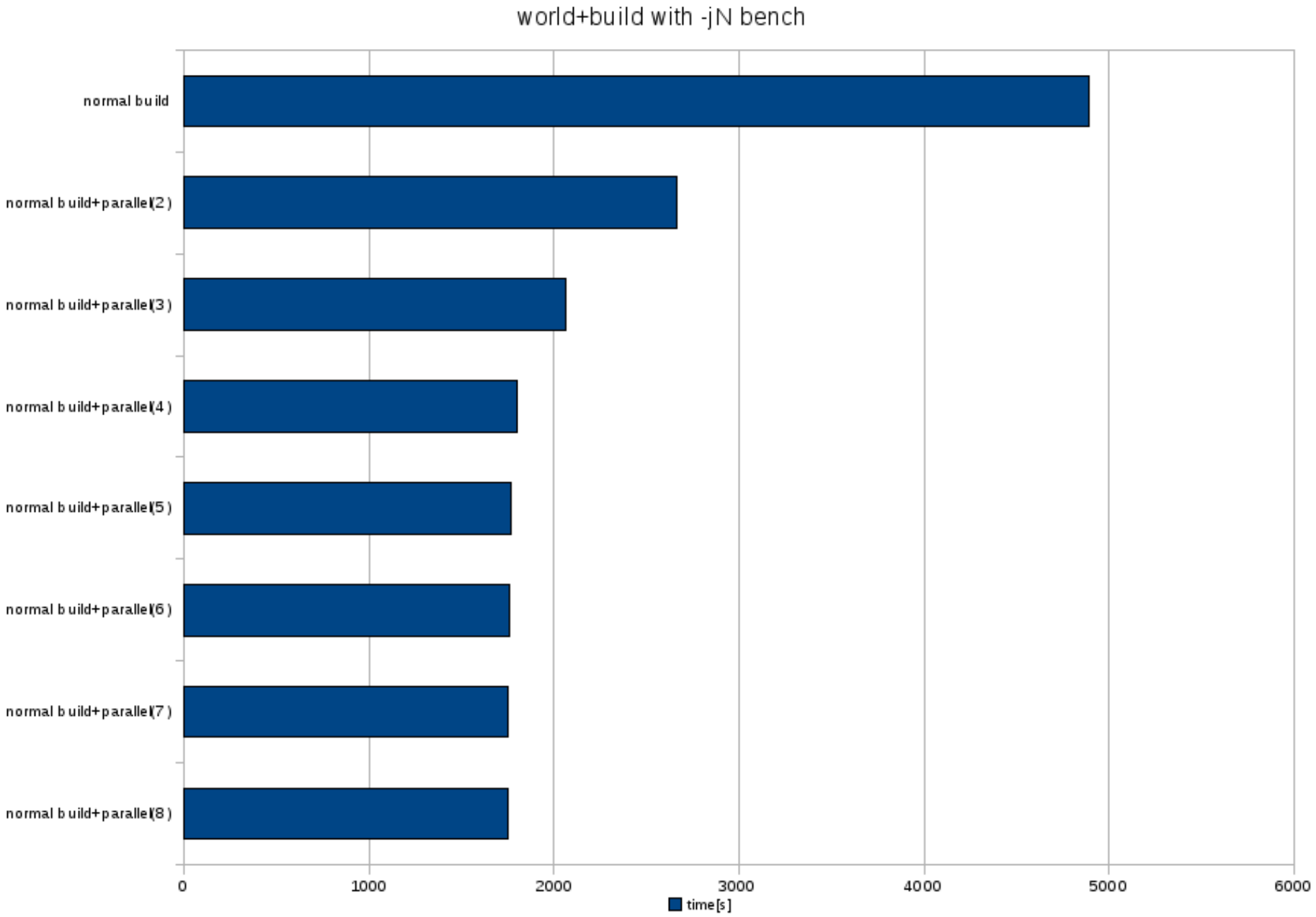
buildkernel bench

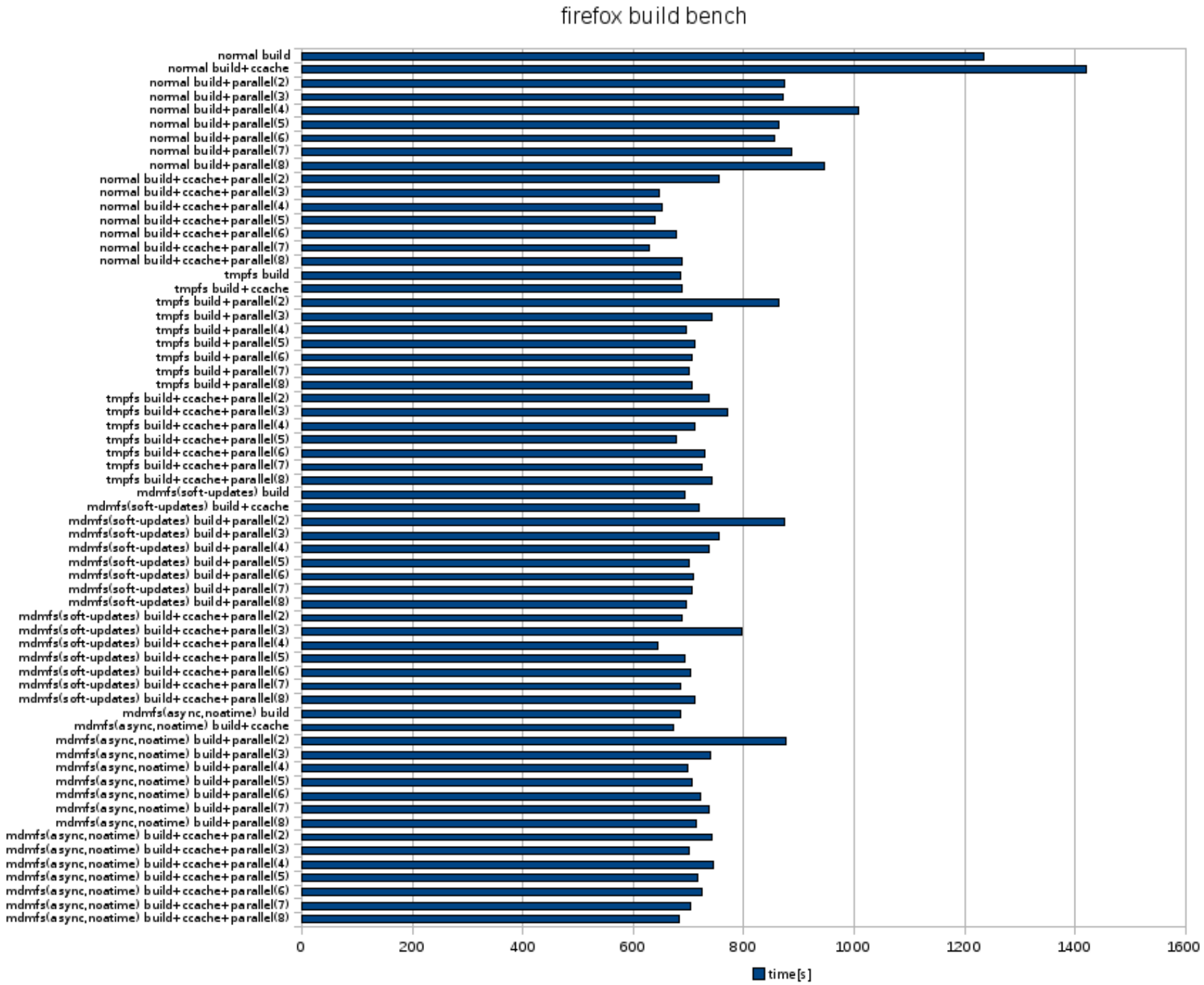


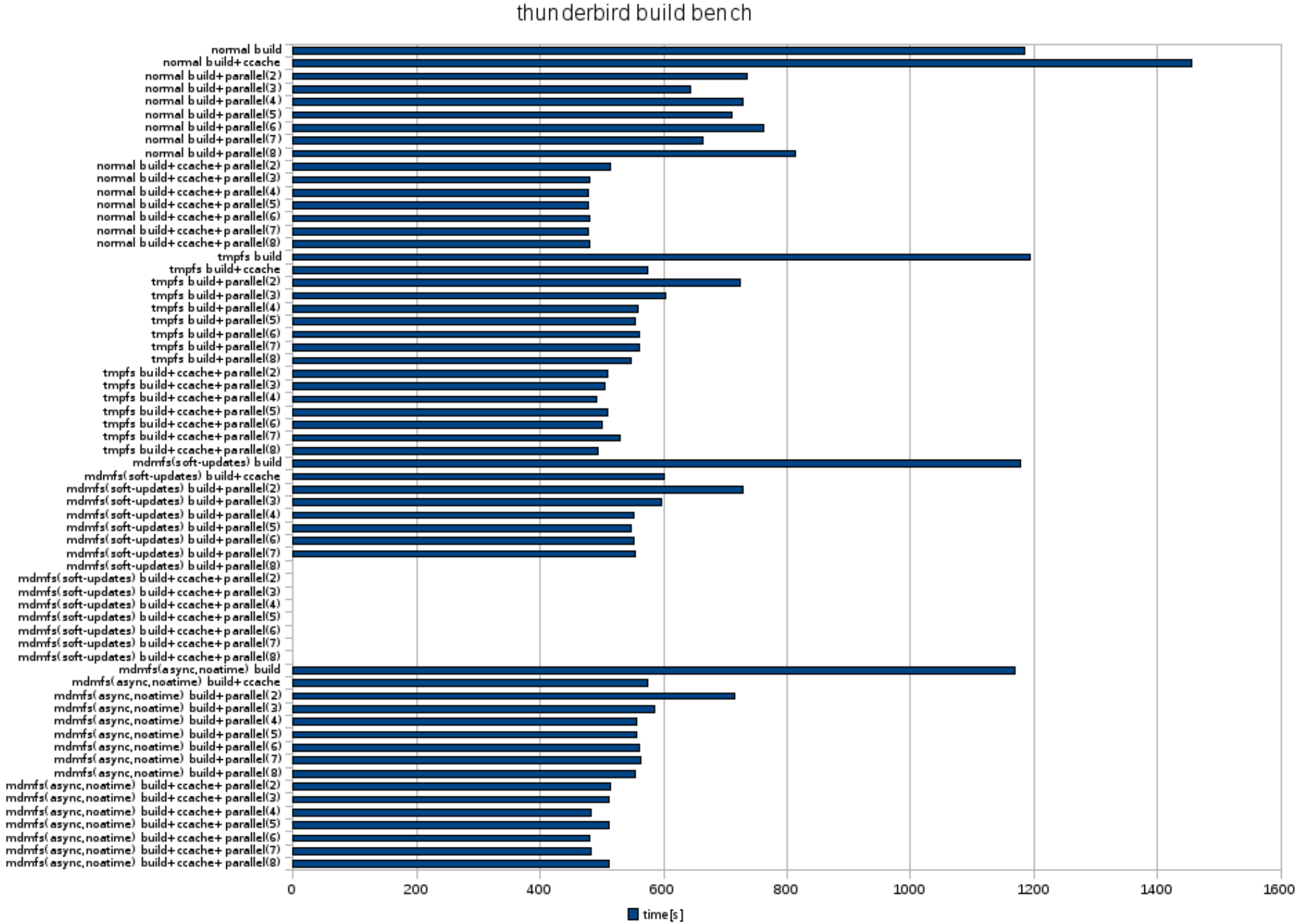
buildworld/buildkernel/copy bench

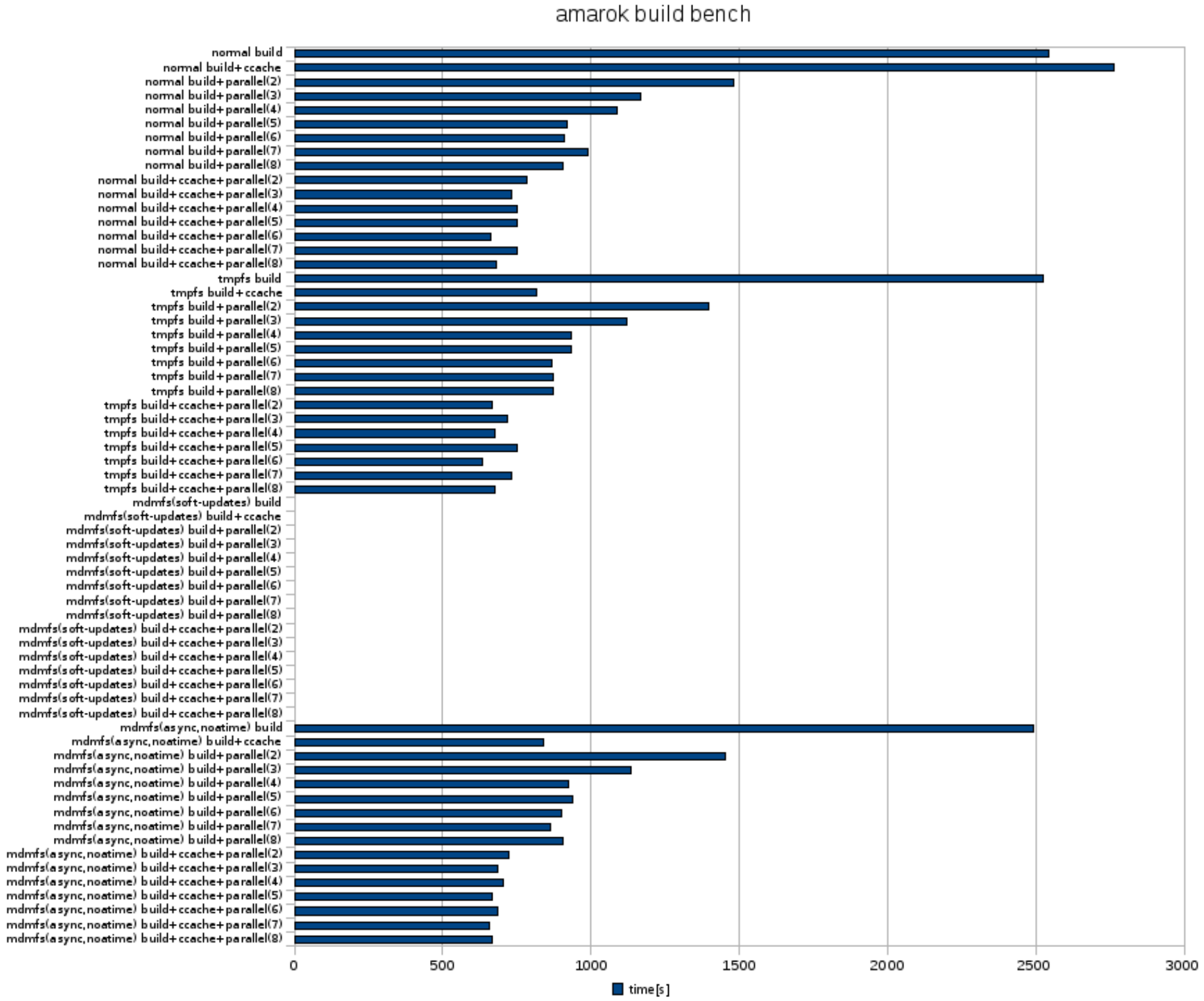


world/kernel paralell build bench




firefox build bench
normal build
normal build+ccache
normal build+parallel(2)
normal build+parallel(3)
normal build+parallel(4)
normal build+parallel(5)
normal build+parallel(6)
normal build+parallel(7)
normal build+parallel(8)
normal build+ccache+parallel(2)
normal build+ccache+parallel(3)
normal build+ccache+parallel(4)
normal build+ccache+parallel(5)
normal build+ccache+parallel(6)
normal build+ccache+parallel(7)
normal build+ccache+parallel(8)
tmpfs build
tmpfs build+ccache
tmpfs build+parallel(2)
tmpfs build+parallel(3)
tmpfs build+parallel(4)
tmpfs build+parallel(5)
tmpfs build+parallel(6)
tmpfs build+parallel(7)
tmpfs build+parallel(8)
tmpfs build+ccache+parallel(2)
tmpfs build+ccache+parallel(3)
tmpfs build+ccache+parallel(4)
tmpfs build+ccache+parallel(5)
tmpfs build+ccache+parallel(6)
tmpfs build+ccache+parallel(7)
tmpfs build+ccache+parallel(8)
mdmfs(soft-updates) build
mdmfs(soft-updates) build+ccache
mdmfs(soft-updates) build+parallel(2)
mdmfs(soft-updates) build+parallel(3)
mdmfs(soft-updates) build+parallel(4)
mdmfs(soft-updates) build+parallel(5)
mdmfs(soft-updates) build+parallel(6)
mdmfs(soft-updates) build+parallel(7)
mdmfs(soft-updates) build+parallel(8)
mdmfs(soft-updates) build+ccache+parallel(2)
mdmfs(soft-updates) build+ccache+parallel(3)
mdmfs(soft-updates) build+ccache+parallel(4)
mdmfs(soft-updates) build+ccache+parallel(5)
mdmfs(soft-updates) build+ccache+parallel(6)
mdmfs(soft-updates) build+ccache+parallel(7)
mdmfs(soft-updates) build+ccache+parallel(8)
mdmfs(async,noatime) build
mdmfs(async,noatime) build+ccache
mdmfs(async,noatime) build+parallel(2)
mdmfs(async,noatime) build+parallel(3)
mdmfs(async,noatime) build+parallel(4)
mdmfs(async,noatime) build+parallel(5)
mdmfs(async,noatime) build+parallel(6)
mdmfs(async,noatime) build+parallel(7)
mdmfs(async,noatime) build+parallel(8)
mdmfs(async,noatime) build+ccache+parallel(2)
mdmfs(async,noatime) build+ccache+parallel(3)
mdmfs(async,noatime) build+ccache+parallel(4)
mdmfs(async,noatime) build+ccache+parallel(5)
mdmfs(async,noatime) build+ccache+parallel(6)
mdmfs(async,noatime) build+ccache+parallel(7)
mdmfs(async,noatime) build+ccache+parallel(8)
0
200
400
600
800
1000
1200
1400
1600
time[s]


thunderbird build bench
normal build
normal build+ccache
normal build+parallel(2)
normal build+parallel(3)
normal build+parallel(4)
normal build+parallel(5)
normal build+parallel(6)
normal build+parallel(7)
normal build+parallel(8)
normal build+ccache+parallel(2)
normal build+ccache+parallel(3)
normal build+ccache+parallel(4)
normal build+ccache+parallel(5)
normal build+ccache+parallel(6)
normal build+ccache+parallel(7)
normal build+ccache+parallel(8)
tmpfs build
tmpfs build+ccache
tmpfs build+parallel(2)
tmpfs build+parallel(3)
tmpfs build+parallel(4)
tmpfs build+parallel(5)
tmpfs build+parallel(6)
tmpfs build+parallel(7)
tmpfs build+parallel(8)
tmpfs build+ccache+parallel(2)
tmpfs build+ccache+parallel(3)
tmpfs build+ccache+parallel(4)
tmpfs build+ccache+parallel(5)
tmpfs build+ccache+parallel(6)
tmpfs build+ccache+parallel(7)
tmpfs build+ccache+parallel(8)
mdmfs(soft-updates) build
mdmfs(soft-updates) build+ccache
mdmfs(soft-updates) build+parallel(2)
mdmfs(soft-updates) build+parallel(3)
mdmfs(soft-updates) build+parallel(4)
mdmfs(soft-updates) build+parallel(5)
mdmfs(soft-updates) build+parallel(6)
mdmfs(soft-updates) build+parallel(7)
mdmfs(soft-updates) build+parallel(8)
mdmfs(soft-updates) build+ccache+parallel(2)
mdmfs(soft-updates) build+ccache+parallel(3)
mdmfs(soft-updates) build+ccache+parallel(4)
mdmfs(soft-updates) build+ccache+parallel(5)
mdmfs(soft-updates) build+ccache+parallel(6)
mdmfs(soft-updates) build+ccache+parallel(7)
mdmfs(soft-updates) build+ccache+parallel(8)
mdmfs(async,noatime) build
mdmfs(async,noatime) build+ccache
mdmfs(async,noatime) build+parallel(2)
mdmfs(async,noatime) build+parallel(3)
mdmfs(async,noatime) build+parallel(4)
mdmfs(async,noatime) build+parallel(5)
mdmfs(async,noatime) build+parallel(6)
mdmfs(async,noatime) build+parallel(7)
mdmfs(async,noatime) build+parallel(8)
mdmfs(async,noatime) build+ccache+parallel(2)
mdmfs(async,noatime) build+ccache+parallel(3)
mdmfs(async,noatime) build+ccache+parallel(4)
mdmfs(async,noatime) build+ccache+parallel(5)
mdmfs(async,noatime) build+ccache+parallel(6)
mdmfs(async,noatime) build+ccache+parallel(7)
mdmfs(async,noatime) build+ccache+parallel(8)
0
200
400
600
800
1000
1200
1400
1600
time[s]


amarok build bench
normal build
normal build+ccache
normal build+parallel(2)
normal build+parallel(3)
normal build+parallel(4)
normal build+parallel(5)
normal build+parallel(6)
normal build+parallel(7)
normal build+parallel(8)
normal build+ccache+parallel(2)
normal build+ccache+parallel(3)
normal build+ccache+parallel(4)
normal build+ccache+parallel(5)
normal build+ccache+parallel(6)
normal build+ccache+parallel(7)
normal build+ccache+parallel(8)
tmpfs build
tmpfs build+ccache
tmpfs build+parallel(2)
tmpfs build+parallel(3)
tmpfs build+parallel(4)
tmpfs build+parallel(5)
tmpfs build+parallel(6)
tmpfs build+parallel(7)
tmpfs build+parallel(8)
tmpfs build+ccache+parallel(2)
tmpfs build+ccache+parallel(3)
tmpfs build+ccache+parallel(4)
tmpfs build+ccache+parallel(5)
tmpfs build+ccache+parallel(6)
tmpfs build+ccache+parallel(7)
tmpfs build+ccache+parallel(8)
mdmfs(soft-updates) build
mdmfs(soft-updates) build+ccache
mdmfs(soft-updates) build+parallel(2)
mdmfs(soft-updates) build+parallel(3)
mdmfs(soft-updates) build+parallel(4)
mdmfs(soft-updates) build+parallel(5)
mdmfs(soft-updates) build+parallel(6)
mdmfs(soft-updates) build+parallel(7)
mdmfs(soft-updates) build+parallel(8)
mdmfs(soft-updates) build+ccache+parallel(2)
mdmfs(soft-updates) build+ccache+parallel(3)
mdmfs(soft-updates) build+ccache+parallel(4)
mdmfs(soft-updates) build+ccache+parallel(5)
mdmfs(soft-updates) build+ccache+parallel(6)
mdmfs(soft-updates) build+ccache+parallel(7)
mdmfs(soft-updates) build+ccache+parallel(8)
mdmfs(async,noatime) build
mdmfs(async,noatime) build+ccache
mdmfs(async,noatime) build+parallel(2)
mdmfs(async,noatime) build+parallel(3)
mdmfs(async,noatime) build+parallel(4)
mdmfs(async,noatime) build+parallel(5)
mdmfs(async,noatime) build+parallel(6)
mdmfs(async,noatime) build+parallel(7)
mdmfs(async,noatime) build+parallel(8)
mdmfs(async,noatime) build+ccache+parallel(2)
mdmfs(async,noatime) build+ccache+parallel(3)
mdmfs(async,noatime) build+ccache+parallel(4)
mdmfs(async,noatime) build+ccache+parallel(5)
mdmfs(async,noatime) build+ccache+parallel(6)
mdmfs(async,noatime) build+ccache+parallel(7)
mdmfs(async,noatime) build+ccache+parallel(8)
0
500
1000
1500
2000
2500
3000
time[s]

ベンチマーク結果からの考察

並列処理は効果的

- システム/カーネルのビルドは並列化で高速化が可能
- ただし、リニアにはスケールしない。3並列以上ではスケールが鈍っている。コアの数が8や16になった場合、どこまでスケールするか不明瞭だが、あまり期待できないと推測される
- Ports Collectionからインストールするアプリケーションでも並列ビルドに対応しているものは、並列化で高速化が可能だが、こちらもスケールにはすぐに限界がやってきそう

キャッシュの効果

- 特定の組み合わせでキャッシュが効くことはあるが、ほかのケースでは性能はあがっていない
- キャッシュが効きやすい状況を作ったつもりで、この結果なので、日常の利用でキャッシュから十分な効果が得られるか不透明。あまりキャッシュによる効果は得られないのではないかと推測される
- ほかのキャッシュシステムやキャッシュ容量の調整で改善される可能性はある

メモリディスク

- データの書き込み先をメモリディスクにしても、性能の向上は確認できない
- メモリ上限を越えるとビルドできなくなることや、スワップが動作することを考えると、メモリディスクから得られる効果は限定的といえるかもしれない
- 特定の用途、たとえばPorts Collectionにおけるmake cleanなどの操作が一瞬で終わるといったような効果は得られる
- HDDとのIOが衝突しなくなることをなどを考えると、大容量のメモリを積んでおいて、ビルド先にメモリディスクを指定する方法はPorts Collectionに対しては負荷分散という点で一定の効果はありそう

今後さらにビルド時間を短縮するには

コア性能向上と細粒度並列化が鍵

- メモリディスクの活用結果から推測するに、HDDをSSDに換装してもシステムやカーネル、アプリケーションビルド時間の短縮化は見込めない
- もっとも効果的なのはコア単体の性能が向上することと、ビルドの並列の細粒度を向上させること、ということになる

make++ by John Birrell

- BSDCan2009, WIPs SessionにおいてJohn Birrell氏が現在取り組んでいるプロジェクト jbuild (make++)について発表
- BSDmakeを改善し、各種便利機能の追加や、ビルドの並列度を高めることを目指す。取り組みが成功した場合、システム/カーネルビルドの並列ビルドに対するスケーラビリティの向上や、Ports Collectionの並列セーフ化が期待できるかもしれない
- 今後の発表や論文の公開に注目しておきたい

FreeBSD情報収集方法

メーリングリストに参加する

- 次のURLから気になるメーリングリストに参加して情報を収集する

  http://www.freebsd.org/community/mailinglists.html

- メーリングリスト検索

  http://www.mavetju.org/mail/view_all.php

フォーラムをチェックする

- 公式のフォーラムとして The FreeBSD Forumがローンチしている。FreeBSD関連の話題が活発にあがっているため、RSSに登録するなどして積極的に情報を収集する。

The FreeBSD Forum
http://forums.freebsd.org/

Web 2.0 ツールを活用する

- 情報源はメーリングリストやIRCといったメディアから、ブログ、mixi、Twitter、RSS、Facebook、MySpace、IMなど多様化している。特に若者ほどその傾向が顕著
- FreeBSD関連の情報を収集するためにこうしたWeb 2.0のツールを積極的に活用する。これまでの情報収集の発想から、情報を得るために実に柔軟な頭に切り替える

AsiaBSDCon at Tokyo

- 東京で開催される*BSD国際会議
- AsiaBSDCon2009 でアジアでの開催は4回目、日本での開催は3年連続の3回目
- 国際会議として*BSD関係者の間でも認知を得てきた
- 開催にはスポンサーによる支援が欠かせない状況。ちょっとでも興味がある企業の方はぜひともsecretary@asiabsdcon.org にまでご連絡を!

情報の多くはEnglish

- 新しい情報のほとんどは英語で流通する。英語の学習に近道はなく、ともかく大量の英語を読み、大量の英語を書き、大量に英語を聴き、大量に英語を喋るしない

Study! Study! Study!

とは言っても、ね ☆ヽ(＞ワ＜*)

- ただでさえ忙しい毎日。英語の勉強にまで手は回らないというのが現実

- そう、そんなあなたのために尽くす漢、それが後藤です。

FreeBSD Daily Topics

- 実にいいメディアがあります

  技術評論社 Gihyo.jp
  　FreeBSD Daily Topics
  　http://gihyo.jp/admin/clip/01/fdt

  後藤が毎日フレッシュなFreeBSD情報を世界中から探してきてまとめています。まさにあなたのためのメディアです。今すぐRSSの登録をよろしくお願いいたします。

FreeBSD勉強会

- FreeBSDを活用している企業の方や、FreeBSDに興味を抱いている若者が交流を持てる場所がないという現状。
- 実験的に新しい試みを：

  技術評論社にて開催
  月1での開催予定「FreeBSD勉強会 (仮)」
  講師がFreeBSD関連技術を講義するスタイル
  講義内容は後日動画で配信・同時配信も検討
  4月半ばにテストスタート
  参加は登録からの抽選方式。無料か少々の参加費用。
  講義の後には懇親会
- 定期的に続けていくことで認知を広め、一定の効果を実現していきたい

ご聴講ありがとうございます

be FreeBSD with you!

Any Question?