TASK: VMSTATで得られる値 IOWAITの出身を探索せよ

上川純一
Linuxカーネル初心者

debian

vmstat

- システムの負荷の概要をおしえてくれるツール
- ディスクIOでつらいのか、CPU負荷がつらいのか、メモリ負荷がつらいのか大まかな部分を教えてくれる
- 疑問： さて、このツールでててくる「cpu wa」っていったい何の数字を教えてくれているの？

ユーザランドの追求(1)：vmstatコマンドはどこだ

- Debianでの調査方法
  - dpkg –S コマンドで検索
  - auto-apt searchコマンドで検索
    - どうやらprocpsパッケージにはいっているらしい

```
$ auto-apt search bin/vmstat
usr/bin/vmstat  base/procps
$ dpkg -S bin/vmstat
procps: /usr/bin/vmstat
```


debian

ユーザランドの追求(2)：procpsソースを探求

- パッケージ名がわかったらソースコードを取得
  - apt-get source procps
  - このパッケージはパッチ管理システムを使っているみたいなので、debian/rules patch

```
$ apt-get source procps
Reading Package Lists... Done
Building Dependency Tree... Done
Need to get 303kB of source archives.
Get:1 http://ring.asahi-net.or.jp unstable/main procps 1:3.2.5-1 (dsc) [616B]
Get:2 http://ring.asahi-net.or.jp unstable/main procps 1:3.2.5-1 (tar) [277kB]
Get:3 http://ring.asahi-net.or.jp unstable/main procps 1:3.2.5-1 (diff) [25.0kB]
Fetched 303kB in 0s (1261kB/s)
dpkg-source: extracting procps in procps-3.2.5
$ cd procps-3.2.5/
$ debian/rules patch
test -d debian/patched || install -d debian/patched
dpatch  apply-all
applying patch 10_sysctl.8 to ./ ... ok.
applying patch 20_kill.1 to ./ ... ok.
applying patch 20_top_manpage to ./ ... ok.
applying patch 30_library_map_freeproc to ./ ... ok.
applying patch 30_pgrep_start_time to ./ ... ok.
applying patch 30_readproc_c to ./ ... ok.
applying patch 30_tload_no_optargs to ./ ... ok.
dpatch  cat-all  >>patch-stampT
mv -f patch-stampT patch-stamp
```

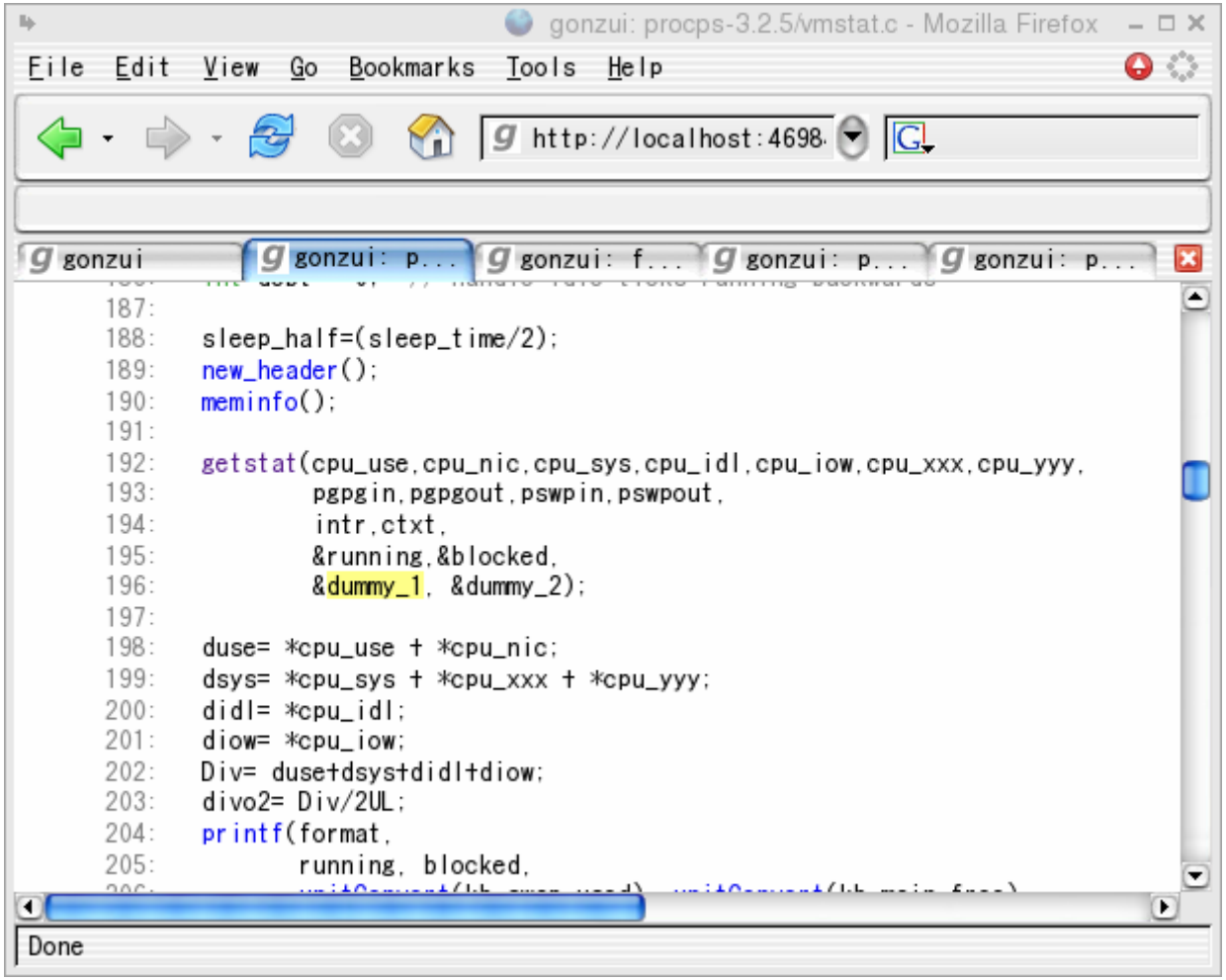

debian

脱線：GONZUIを導入

- WEBサーバになっていてWEBインタフェースがある
  - ./gonzui-server
- ソースコードを検索するツール
  - ./gonzui-import –apt procps
    - パッチをapplyしてくれないのでパッチを書いてみた
  - ./gonzui-import procps*.dsc
    - なんかできないのでパッチを書いてみた

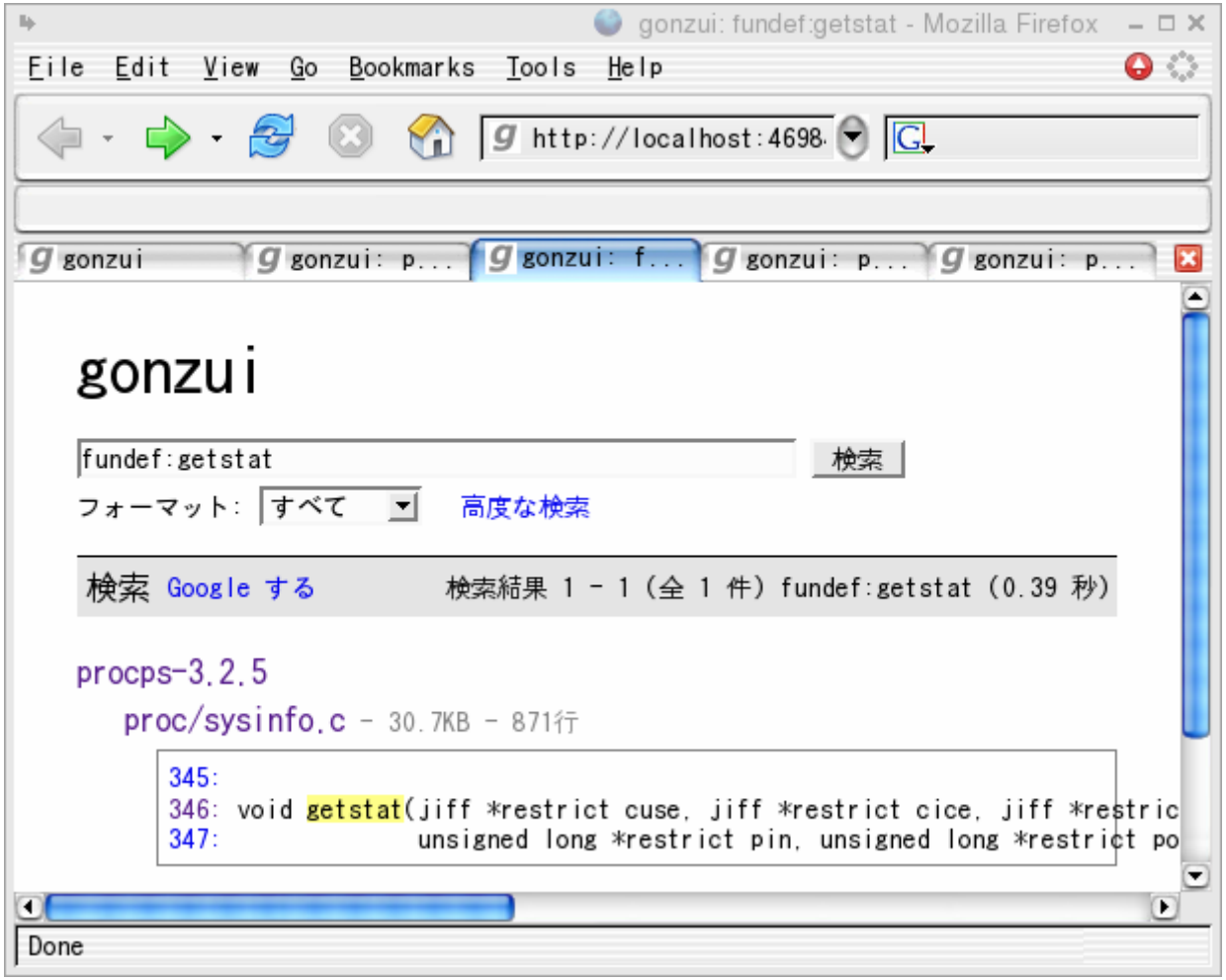
debian

ユーザランドの追求(3) :vmstat.cの該当部分をみつける



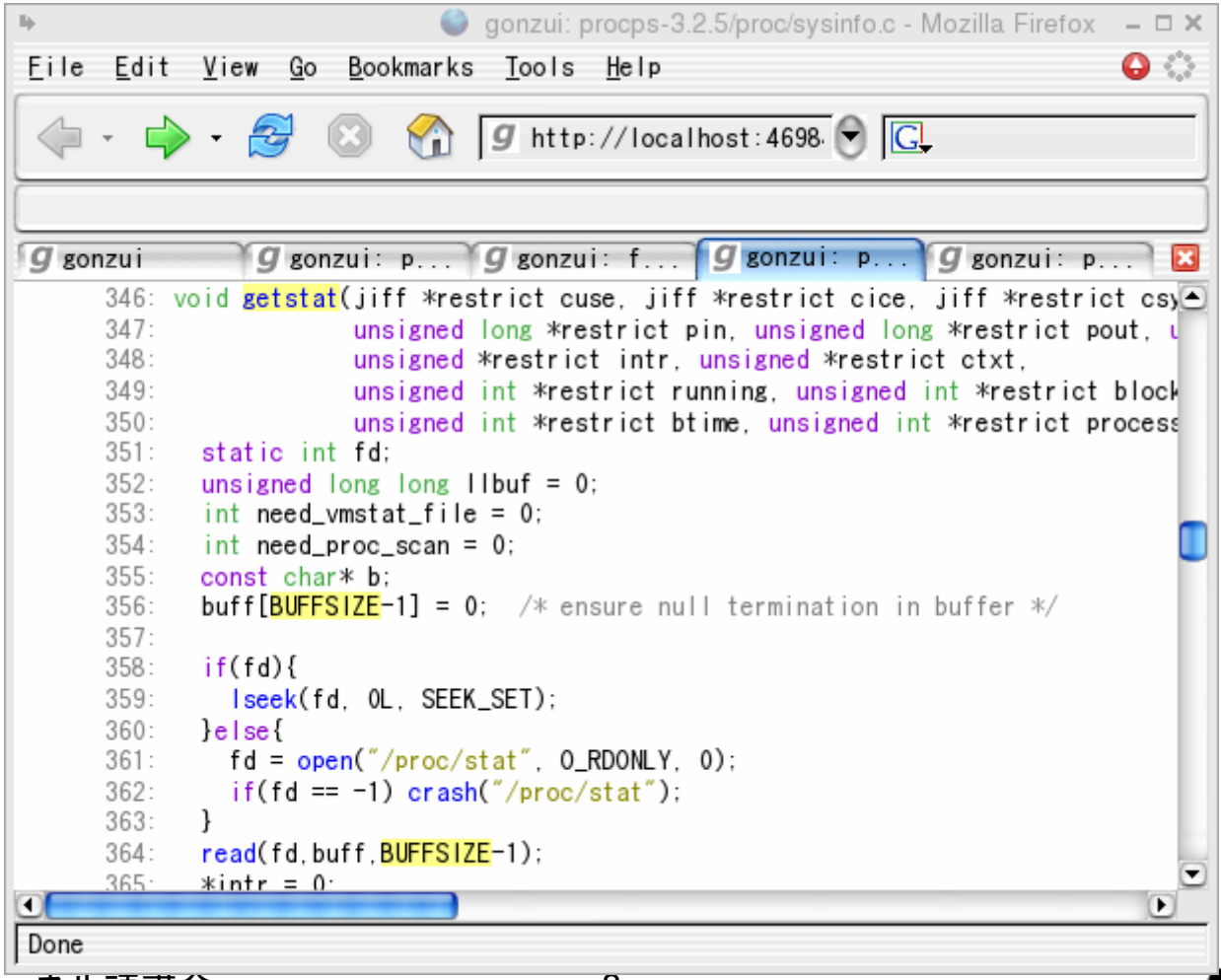
debian

ユーザランドの追求(4)：getstatの関数定義元を探す



debian

ユーザランドの追求(5)：getstatは/proc/statを見ているだけだという事実に驚愕する



debian

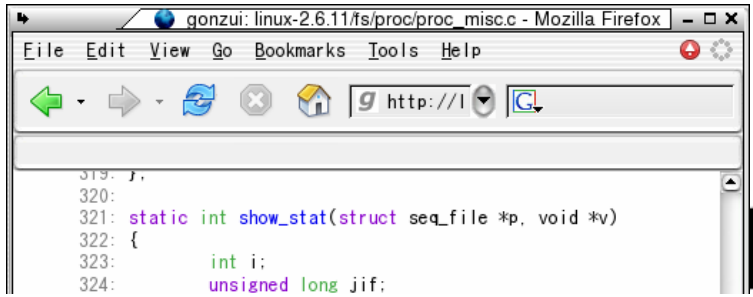
ユーザランドの追求：vmstatのソースコードの結論

- どうやらvmstatはvmstat.c から生成されているらしい
- そこでは、値を得るためにgetstat()という関数を活用しているようだ
- proc/sysstat.cでgetstat()関数が定義されているようだ
- そこを見ると、getstat ()関数は/proc/statの内容を見ているだけのようだ
  - カーネルソース読め

debian

カーネル側の追求：/proc/statはどこで提供しているのか

- 過去の経験でfs/proc/proc_misc.cあたりだろうと見る
- proc/statファイルを開くとproc/stat ファイルはproc_misc_init()のcreate_seq_entry(“stat”, 0, &proc_stat_operations);で作成されているようだ
- proc_stat_operations構造体で指定してあるstat_open()関数がどうやら呼ばれるらしい
- stat_openはshow_stat()を呼び出し、そこでcpustat情報を返しているらしい



カーネル側の追求：cpustatとは何者か

- include/linux/kernel_stat.h: kstat_cpu = per_cpu で、kernel_stat構造体がCPU毎にあるらしい。
- struct cpu_usage_stat cpustat; を参照しているらしい。
  - kstat_cpu().cpustat.user/nice/などを更新しているのはどこだろう。

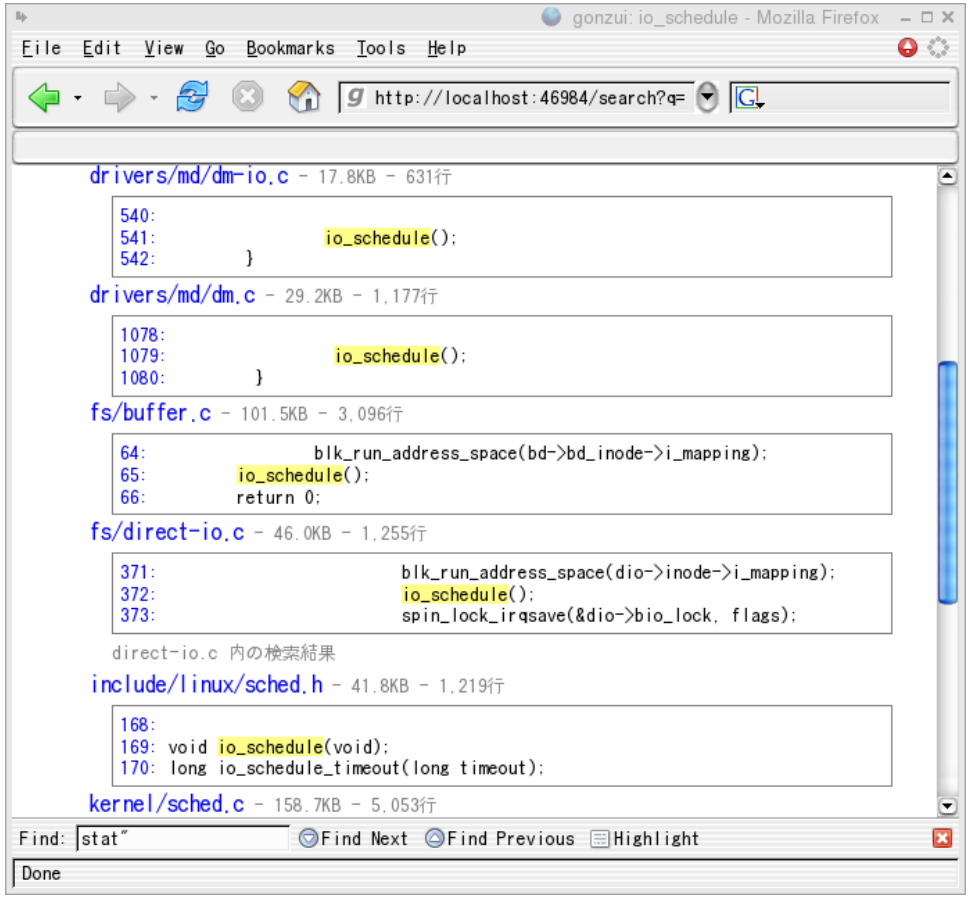
カーネル側の追求：cpustatの更新している場所

- kernel/sched.cのscheduler_tickがそれっぽい
  - パラメータのuser_ticksとsys_ticksが与えられている。
  - kernel/timer.c: update_process_timesから、user_ticksかsys_ticksのどちらかが1になるように呼び出されている。forkからは両方0で呼び出されている
- ロジックを見ると現在のプロセスがidleの場合にIO待ちの状態であるプロセスが一つでもあれば（nr_iowait>0）iowait の値が増加するらしい、そうでないならidleとみなす
- nr_iowaitはio_schedule_timeout/io_scheduleでのみ変更されているようだ

debian

カーネル側の追求：io_scheduleを呼ぶ場合って？

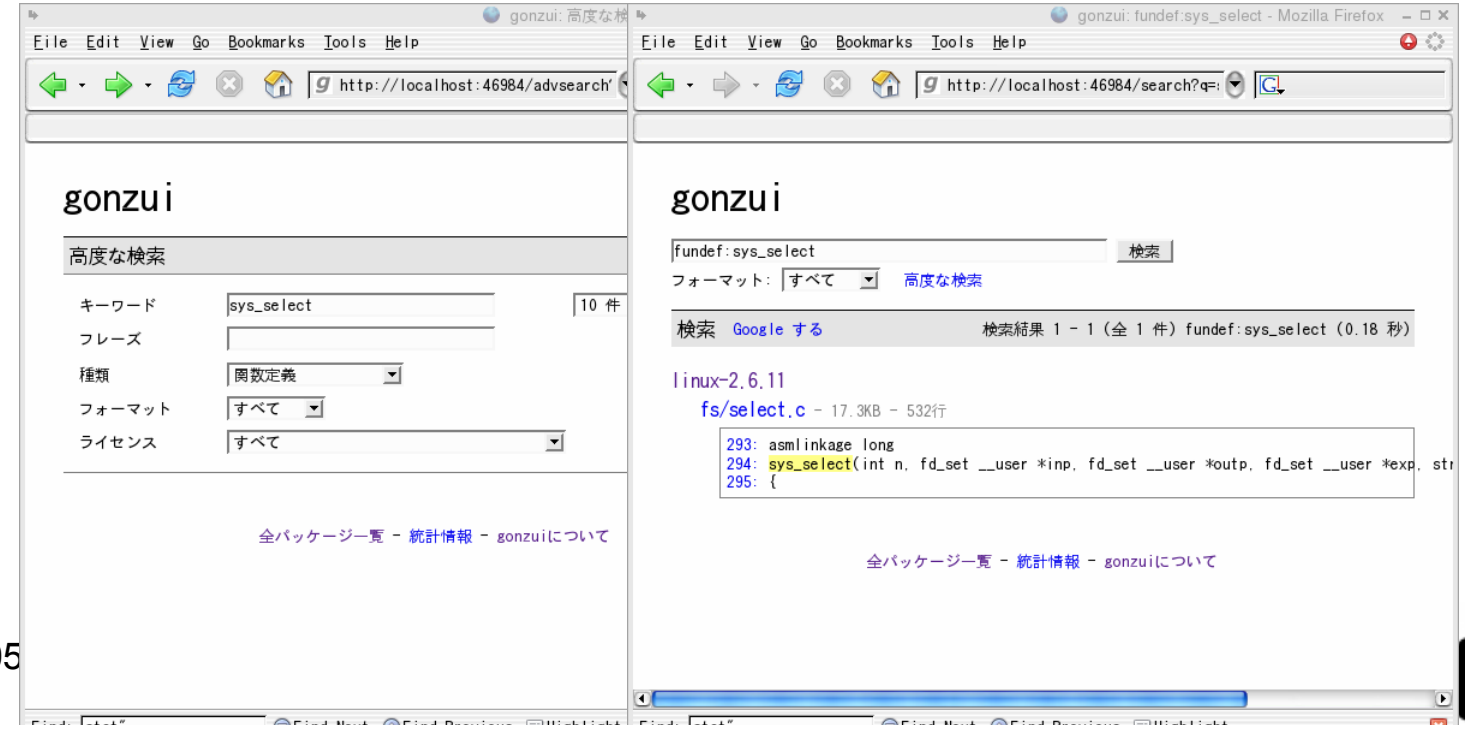
- gonzuiで検索
  - どうやらio_scheduleはファイル名から眺めるとFS関係、ディスク書き込み関連っぽいところでのみ利用されているようだ
  - io_schedule_timeoutはblk_congestion_wait(int rw, long timeout)のみから呼ばれている
- どうやらディスクアクセスを中心に呼ばれており、ネットワーク関連では呼ばれないようだ



debian

念のためにネットワーク関連の待ちでは呼ばれていないことを確認

- なんとなくsys_selectあたりを調べてみる
- sys_selectはdo_selectを呼ぶだけのようだ
- fs/select.cのdo_selectは待ちのためにschedule_timeoutを呼んでいる



まとめ

- Debian GNU/Linuxでアプリケーションからカーネルの部分まで追跡する方法について説明しました
- 追跡にgonzuiを活用してみました
- ディスク関連の操作で”cpu wa”の値が増加するらしいということがわかりました。ネットワークの通信ではどうやら増加しないっぽい雰囲気です。

debian