株式会社 ITS MORE
2020年4月設立

Digitally signed by 佐藤 豊
Date: 2020.08.25 21:18:38 +09'00'

2020年8月25日 投稿者: SATOXITS

GShell 0.2.1 – ツイステッドストリーム

開発：機能GShellでリモートコマンドを実装してる時に思ったのですが、やはり一本のストリーム上でソフトにEOFというかEORを送るのは難しいかなと。

社長：なので制御用とは別に毎回TCPを張って使い捨てるという計画ですよね。FTP式。

開発：それもまた野蛮というか、各レイヤに弊害が出るような予感もするするわけです。特にアプリケーション層で中継する場合には、接続の確立は一大イベントですから、認証もするでしょうしログも取る。エンドーエンドにしても、通信をSSLとかで暗号化するなら、ネゴにかなり重い処理というか遅延が生じる。

開発：1KB程度の不揃いで細かいデータを毎秒10,000個送れるのを目標とするとします。これはネットのバンド幅的には10MB/sでたやすいし、データの出どころがディスクであっても、まあHDDの場合ディレクトリ・ファイルのopenに律速されて状況によるでしょうけど、SSDなら楽勝の数値でしょう。

基盤：へのかっぱですね。HDDにしても、RAM上にinodeとblockが残ってれば、問題ないんじゃないですか。頻繁にアクセスするようなファイルなら。ちょっと実測してみます…

```
iMac% ls -l ../go1.15.src.tar.gz
-rw-r--r--@ 1 ysato  staff  23002901 Aug 15 10:19 ../go1.15.src.tar.gz
iMac% time tar xf ../go1.15.src.tar.gz
tar xf ../go1.15.src.tar.gz  0.36s user 2.03s system 98% cpu 2.430 total
iMac% time du -a go | wc
   10150   20300  404055
du -a go  0.01s user 0.03s system 98% cpu 0.039 total
wc  0.00s user 0.00s system 5% cpu 0.039 total
iMac% time tar cf - go > /dev/null
tar cf - go > /dev/null  0.58s user 4.11s system 80% cpu 5.839 total
iMac%
```

基盤：Goのソースディレクトリの構造はまあ一例ですけど、du で見ると、ディレクトをなめてinode情報を収集するだけなら、1秒に200,000ファイルはイケます。10,000は楽勝です。blockまで読み書きするとなると、tarの例では、2,000ファイル/秒くらいに落ちてます。10,000は厳しいです。

社長：tar xf より cf のほうが遅いというのは意外ですね。

開発：大半はOS時間みたいですね。めっちゃstatとかしてるんでしょうか？

社長：いずれシステムコールをトレースして覗きましょう。

開発：でも GShell 版の rsync というか mirror でのファイル転送では、tar は使わない予定です。ただ find 相当の内部コマンドでディレクトリをなめて、ファイルをかたはしからメタ情報とともに投げる。もちろん tar のようなパディングとか無いです。

開発：毎秒10,000個ということは、一個あたり0.1ミリ秒です。これはLANであってもちょっと厳しいレイテンシ。遠隔なら250msかかってしまうことも有るという実例は我社のフランクフルト支所との通信で目にしているところです。TCPを張ってリクエストとレスポンスをするだけで1秒かかる事は必定。毎秒1個しか送れないことになります。

開発：なので GShell版の sync では、まず受け取り側の 10,000（10K）個のファイルのstatを送る。ファイル名が可変長なので読めないですが、一連のファイル名トラバースの中の相対パスで、ベースネームだけにできる。ファイル名とメタ情報含めて32バイト平均くらいとすれば、320Kバイト。フランクフルトのように1MB/sであったとしても、0.3秒です。というかこれは、データの送信と並行して送っても良い。

開発：次に受け取り側は自分のファイルシステムと比較しながら変化のあるファイルだけを送信する。ここはメタ情報だけで済ませるか、中身のダイジェストまで含めるか、思案のしどころです。

社長：中身まで読んじゃうと tar の例のように 2,000個/秒に落ちてしまうかも知れない。

開発：いえ、あれは tar が遅すぎです。GShellでgrep相当内部コマンドを試した時はもっと高速でした。ちょっと、 チェックサムだけ取るコマンドを作って試してみます…

＊　＊　＊

開発：そういうわけで、いくつはチェックサム方式を実装して測ってみました。まず、Unix で昔よく使ってた sum コマンドのアルゴリズム。BSDsum と呼ばれてるらしいです。

```
gsh/0.2.1 (2020-08-25) SatoxITS(^-^)/
!1! cd go1.15-src
!2  @1 [Aug 25 20:21:08] /Users/ysato/go1.15/go1.15-src
!2! -sum -s -ru -bsd
38978 95028731 // 90.626MiB / 10149 files, 9.144KiB/file
total: 10150 files (1015d 0s 0h) 95.418459 MB (120.48 MBK)
--cksum-size: 95028731 (90.626MiB) / 10149 files, 9.144KiB/file
--cksum-time: 428.628ms/total, 42.233us/file, 23677.9 files/s, 221.704MB/s
--cksum-rusg: 506.365ms/sum, 306.440ms/usr, 199.925ms/sys
!2! -sum -s -ru -bsd
38978 95028731 // 90.626MiB / 10149 files, 9.144KiB/file
total: 10150 files (1015d 0s 0h) 95.418459 MB (120.48 MBK)
--cksum-size: 95028731 (90.626MiB) / 10149 files, 9.144KiB/file
--cksum-time: 424.595ms/total, 41.836us/file, 23902.7 files/s, 223.810MB/s
--cksum-rusg: 500.641ms/sum, 305.535ms/usr, 195.106ms/sys
```

基盤：23,000ファイル/秒、223MB/s 出てますね。HDDでは厳しいですが、SSDでならイケる可能性のあるスピード。

開発：次は、Go言語のCRC32パッケージによるCRC32。

```
!4! -sum -s -ru -ieee
7609718793 95028731 // 90.626MiB / 10149 files, 9.144KiB/file
total: 10150 files (1015d 0s 0h) 95.418459 MB (120.48 MBK)
--cksum-size: 95028731 (90.626MiB) / 10149 files, 9.144KiB/file
--cksum-time: 525.057ms/total, 51.734us/file, 19329.3 files/s, 180.987MB/s
--cksum-rusg: 610.674ms/sum, 412.260ms/usr, 198.414ms/sys
```

社長：さすがにCRCの計算でユーザCPU時間が多いですが、それがネックにはならなそうですね。というか、BSD sum より速いんですか…

開発：ちなみに社長が昔自作したCRC32だと、20MB/s でした。tar と同格です。

```
!2! -sum -s -ru -unix
Aug 25 20:43:52: 4719 files (494d 0s 0h) 39.975012 MB (53.10 MBK)
Aug 25 20:43:54: 9024 files (922d 0s 0h) 80.351294 MB (102.98 MBK)
1019810010 95028731 // 90.626MiB / 10149 files, 9.144KiB/file
total: 10150 files (1015d 0s 0h) 95.418459 MB (120.48 MBK)
--cksum-size: 95028731 (90.626MiB) / 10149 files, 9.144KiB/file
--cksum-time: 4.70s/total, 464.022us/file, 2155.1 files/s, 20.179MB/s
--cksum-rusg: 4.88s/sum, 4.61s/usr, 268.906ms/sys
```

社長：(^-^;

開発：次に、単純な足し算。まさにsum。SysV sum と同系です。ただし、64ビット。

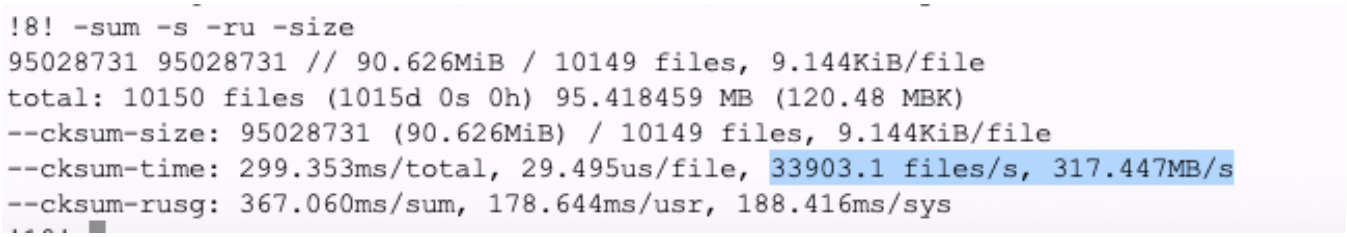
```
!6! -sum -s -ru -sum
7609718793 95028731 // 90.626MiB / 10149 files, 9.144KiB/file
total: 10150 files (1015d 0s 0h) 95.418459 MB (120.48 MBK)
--cksum-size: 95028731 (90.626MiB) / 10149 files, 9.144KiB/file
--cksum-time: 342.728ms/total, 33.769us/file, 29612.3 files/s, 277.271MB/s
--cksum-rusg: 410.421ms/sum, 217.609ms/usr, 192.812ms/sys
```

基盤：流石に速いですね。とはいえ、CRC32の半分程度は食うと…

開発：多数の細かいファイルを読むところで律速されていますが、でっかいファイルのsumなら、2GB/s 以上出ます。もちろん、ファイルがメモリ上にバッファされてた場合ですが。

開発：そして最後に、中身を読まないで外側だけなめる式。ここではサイズの和だけ計算しています。各ファイルの日付とサイズの列をCRCにでもすれば、変更の検出には十

分かも知れません。



基盤：SSDならまだ追随するかも知れませんね。

社長：結論としては、tar は GShell版の sync に使うには遅すぎるということですね。

開発：少なくともこのmacOS備え付けのtarでは。ギガビットのインターネット、100MB/s のネットを生かせないですね。20MB/s に律速してしまうかも知れない。それにどの道、大きなファイルの中をブロック化して差分だけsyncする際には、tarの方法は使えないです。

＊　＊　＊

社長：GShell版のsyncコマンドは jsync とかになるんですか？

開発：いえもういちいちトップレベルのコマンド名を付けるのは面倒なので、x sync とか x put とかにしようと思います。OpenSSL式です。

社長：j じゃなくて x なんですか。

開発：j より x のほうがカッコいいかなみたいな。j の弱点は、特に小文字では見栄えが悪いというか、iと判別がしにくいことです。

基盤：フォントにもよりますが、大文字 X と小文字 x が単体では区別しにくいという問題があると思います。

社長：個人的には y がいいですね。y と Y はふつう明らかに違う。だもんでわたしはDeleGate版のリモートシェルをyyshにしました。b と B でも良いですが。

開発：r と R という線もあると思いますが、r のフォントが一般にしょぼいですね。q

も p も t もしょぼい。l は 1と区別がつかないことが多い。k はトポロジー的には K と同じ…　そういう観点からは、y と Y も落第ではありますが。候補は、a/A、b/B、d/D、e/E、f/F、g/Gと、 h /H、i/I、j/J、n/N、q/Q、r/R、t/T、y/Y あたりですかね。

社長：母音字はIMEと干渉する可能性があっていやですけど、e という小文字は好きです。g はカッコいいけど、下にはみ出てるのがちょっと嫌。b と dとか、h と n とか、q と p とか、t と f とか、似た形状のがあるのも嫌。

基盤：候補が a/A と e/E に絞られましたw

社長：eは大文字のEの形状ががさつなのが嫌。Fに似てるし。a にしましょう。

基盤：アルファベット中のアルファベットですね。

社長：では a にしましょう。当社としての正式決定です。GShellをやめてAlphaShellに改名すると良いかもですね。略記およびコマンド名は ash。

開発：永遠のアルファ版であると。あ、フォントによってαはいまいち見栄えしないという弱点が…　βのほうがカッコいい。

開発：ビルトインコマンドの正式名称も alpha コマンドですかね。あ、そうすると async は async を暗示しているみたいで面白いですねw

基盤：なんにしろ内部コマンドは1文字というのは気分が良いです。多分そんな外部コマンドは無いでしょう…

```
iMac% which {a..z}
a not found
b not found
c not found
d not found
e not found
f not found
g not found
h not found
i not found
j not found
k not found
l not found
m not found
n not found
o not found
p not found
q not found
r: shell built-in command
s not found
t not found
u not found
v not found
/usr/bin/w
x not found
y not found
z not found
```

社長：そういえば w がありました。最近はあまり使う必要がなくなりましたが。

開発：zsh は r をビルトインコマンドにしてるんですね。なんだろう・・・　man zshbuiltins。

```
     r       Same as fc -e -.
```

基盤：それ、alias で良くね？

社長：alias で表現できない事情があるんでしょうね。

開発：しかし今どき、ハイパーテキストでないマニュアルとか、つらすぎますね。
GShellではきっちり、HTML/HTTPでビルトインマニュアルをサービスしたいと思います。

＊　＊　＊

開発：今日はTCPを束ねて使う方法をやってみようと思ったのですが、チェックサムにハマってしまいました。

社長：でもCRCは意外と速いというか。まあ実際ディスクアクセスに使われてるんだから当たり前でしょうけど。

開発：ハードで実装すれば、メモリ転送と同等の速度でイケるでしょうね。ていうか、実際そうなってるんだと思いますが。組み込みのチップにはCRCユニットがふつうに付いてますから、インテルのチップにも当然内蔵していると思うんです。それをユーザに開放しているかですね。

社頭：CRC付きのDMAをユーザが使えると良いですね。というか、GoのReadあたりとか、そもそもOSのread/write, send/recv システムコールにそういうフラグがあればユーザがいちいちか書かなくて良いと思いますが。単なるオクテットストリームの汎用CRC。

開発：ioctl とかにありますかね？

— 2020-0825 SatoxITS

- メモ
  1. NNTPにおけるIHAVE/SENDME的プロトコル。双方向にIHAVE。DOYOUHAVE
  2. 常時自分の全ファイルのハッシュを維持して置く。ファイル変更監視。inodeテーブル監視
  3. OOBはSSLライブラリ経由では使えない
  4. TCPを2本以上束ねて使う。緊急チャネルとバックグラウンド的チャネル。無限長データストリームチャネル

http-im3-gsh-gsh-0.2.1.go_ ダウンロード

```
/*
                                GShell version 0.2.1 // 2020-08-25 // SatoxITS
```

GShell // a General purpose Shell built on the top of Golang

It is a shell for myself, by myself, of myself. –SatoxITS(^–^)

| NewWindow | Unfold | Fold | Stop | Close | */ /*

▶ Total Source of GShell

/ /

▶ Overview

/ /

▶ Go Source Code Index

*/ //

▶ Go Source Code

//

▶ Consideration

/*

▶ References

Fin.

–> */ //