WDT 测试性报告

WDT 是 Facebook 开源的一个嵌入式函数库并且也是一个命令行工具，通过多 TCP 通道可以在 2 个系统快速传送数据。

WDT 的目标是通过最小的系统消耗，达到最大的整体传输速率。
WDT 号称自身都以独立的模块切割以方便独立化编译（但是实操中发现编译极其困难，而且在不同的 Linux 系统中存在着各种因 LIB 库版本导致的问题）。

WDT 称在内部的速度测试效果理想，并且在超远距离的传输中有着明显的优势，在没有“瓶颈”的情况下可以达到甚至 4GB/s 的速度 (必须使用万兆网卡)。

由于在 CentOS 上的编译异常困难，所以我们这次测试采用了文档上例子中的 Ubuntu 系统。

编译按照文档的操作：

On Ubuntu 14.04 - to get g++ 4.9

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt-get upgrade
```

If using a vmware image/starting fresh

```
 sudo vmware-config-tools.pl    # to setup shared folders
```

openssl should already be there but may need update to 1.0.x

必须要 apt-get upgrade,并且 gcc g++版本需要满足 4.9 如果高版本的话编译

可能失败(需要做 gcc c++ 等执行文件的 link)。

Build Instructions

Install Cmake 3.2 or greater. See directions below for Mac (Cmake 3.3/head on a Mac is recommended for Xcode support)

```
wget http://www.cmake.org/files/v3.2/cmake-3.2.3.tar.gz
tar xvfz cmake-3.2.3.tar.gz
cd cmake-3.2.3
./bootstrap --prefix=/usr --parallel=16 && make -j && sudo make install
```

Get folly source tree

```
git clone https://github.com/facebook/folly.git
Install glog-dev (includes gflags, libunwind), boost system, double conversion if you can find a
binary distrubution for your variant of linux: libjemalloc-dev dependency is optional

sudo apt-get install libgoogle-glog-dev libboost-system-dev \
libdouble-conversion-dev libjemalloc-dev
Otherwise, Build double-conversion, gflags and glog from source

It's important to build and configure gflags correctly for glog to pick it up and avoid linking errors
later or getting a wdt without flags working

git clone https://github.com/schuhschuh/gflags.git
mkdir gflags/build
cd gflags/build
cmake -DGFLAGS_NAMESPACE=google -DBUILD_SHARED_LIBS=on ..
make -j && sudo make install
git clone https://github.com/google/glog.git
cd glog
./configure # add --with-gflags=whereyouinstalledgflags
# to avoid ERROR: unknown command line flag 'minloglevel' later
make -j && sudo make install
If double-conversion isn't available via apt-get:

git clone https://github.com/floitsch/double-conversion.git
cd double-conversion; cmake . -DBUILD_SHARED_LIBS=on
make -j && sudo make install
```

编译过程相当蛋疼，首先 autoconfig 版本必须为 1.14 这个在之前 upgrade 时可能升到这个版本，但是如果升到 1.15 那也是编译不了，如果 1.13 也是不行必须手动修改 config 文件为 1.14 因为代码被写死了。。貌似。其次 wdt 编译必须跟 folly 在同一目录，并且 folly 的目录结构也很乱，必须要按照很杂乱的结构编译，不然无法通过。

```
Build wdt from source

cmake pathtowdtsrcdir -DBUILD_TESTING=on    # skip -D... if you don't want tests
make -j
CTEST_OUTPUT_ON_FAILURE=1 make test
sudo make install
```

总算编译通过了，开始测试。

我们对比下 SCP 与 WDT 的速度：

接收端：

root@ubuntu:~# wdt -directory /home/ubuntu/test/ -transfer_id 1 -start_port 22356

这里需要吐槽下，WDT 竟然需要指定 start_port 端口，这台不指定的话是从 4xxxx 端口开始启动的（说明中默认是从 22356 启动） 而 Sender 端默认从 22356 开始推送，两边竟然不协商端口导致 ”connection refuse” 错误，不知道 wdt 这么设计的初衷是什么。

I0316 18:42:13.630023 3167 WdtFlags.cpp:63] wdt> Running WDT 1.29.1703140 p 29
I0316 18:42:13.630658 3167 WdtOptions.cpp:56] wdt> start_port is specified, setting static_ports true
I0316 18:42:13.630705 3167 EncryptionUtils.cpp:83] wdt> Openssl library initialized
I0316 18:42:13.630723 3167 WdtFlags.cpp:63] wdt> Running WDT 1.29.1703140 p 29
I0316 18:42:13.631186 3167 WdtOptions.cpp:56] wdt> start_port is specified, setting static_ports true
I0316 18:42:13.631213 3167 WdtResourceController.cpp:29] wdt> Updated max number of senders for _root controller_ to 0
I0316 18:42:13.631233 3167 WdtResourceController.cpp:22] wdt> Updated max number of receivers for _root controller_ to 0
I0316 18:42:13.631252 3167 Throttler.cpp:73] wdt> No average rate specified
I0316 18:42:13.631265 3167 Throttler.cpp:80] wdt> No peak rate specified
I0316 18:42:13.631286 3167 Wdt.cpp:24] wdt>One time initialization of WDT for wdt
I0316 18:42:13.631310 3167 Throttler.cpp:98] wdt> Updating the rates avgRateBytesPerSec : -1048576.00 bucketRateBytesPerSec : 0.00 bytesTokenBucketLimit : 0.00
I0316 18:42:13.631376 3167 Receiver.cpp:45] wdt> WDT Receiver 1.29.1703140 p 29
I0316 18:42:13.631404 3167 WdtBase.cpp:94] wdt> using wdt protocol version 29
I0316 18:42:13.631444 3167 FileCreator.cpp:319] wdt> dir already exists /
I0316 18:42:13.631472 3167 FileCreator.cpp:319] wdt> dir already exists /home/
I0316 18:42:13.631491 3167 FileCreator.cpp:319] wdt> dir already exists /home/ubuntu/
I0316 18:42:13.631513 3167 FileCreator.cpp:319] wdt> dir already exists /home/ubuntu/test/
I0316 18:42:13.631538 3167 Receiver.cpp:169] wdt> aes128gcm encryption is enabled for this transfer
I0316 18:42:13.631556 3167 Receiver.cpp:172] wdt> Receiver generating encryption key for type aes128gcm
I0316 18:42:13.631747 3167 EncryptionUtils.cpp:115] wdt> New encryption params 0x7ffc3a9758e0 2:...13337153740194504297...
I0316 18:42:13.632074 3167 ReceiverThread.cpp:1003] wdt> Thread[0, port: 22356] Listening on port 22356
I0316 18:42:13.632189 3167 ReceiverThread.cpp:1003] wdt> Thread[1, port: 22357] Listening on port 22357

I0316 18:42:13.632290 3167 ReceiverThread.cpp:1003] wdt> Thread[2, port: 22358] Listening on port 22358
I0316 18:42:13.632391 3167 ReceiverThread.cpp:1003] wdt> Thread[3, port: 22359] Listening on port 22359
I0316 18:42:13.632490 3167 ReceiverThread.cpp:1003] wdt> Thread[4, port: 22360] Listening on port 22360
I0316 18:42:13.632587 3167 ReceiverThread.cpp:1003] wdt> Thread[5, port: 22361] Listening on port 22361
I0316 18:42:13.632683 3167 ReceiverThread.cpp:1003] wdt> Thread[6, port: 22362] Listening on port 22362
I0316 18:42:13.632778 3167 ReceiverThread.cpp:1003] wdt> Thread[7, port: 22363] Listening on port 22363
I0316 18:42:13.632796 3167 Receiver.cpp:211] wdt> Registered 8 successful sockets
I0316 18:42:13.632871 3167 wdtCmdLine.cpp:321] wdt> Starting receiver with connection url wdt://ubuntu:22356?Enc=2:...13337153740194504297...&dir=%2fhome%2fubuntu%2ftest%2f&id=1&num_ports=8&recpv=29
wdt://ubuntu:22356?Enc=2:c9574dde63c61f2f9537f1e3ea0a57d5&id=1&num_ports=8&recpv=29
I0316 18:42:13.632951 3167 Receiver.cpp:454] wdt> Starting (receiving) server on ports [22356 22357 22358 22359 22360 22361 22362 22363] Target dir : /home/ubuntu/test/
I0316 18:42:13.632977 3167 Throttler.cpp:73] wdt> No average rate specified
I0316 18:42:13.632992 3167 Throttler.cpp:80] wdt> No peak rate specified
I0316 18:42:13.633005 3167 WdtBase.cpp:183] wdt> Enabling throttling avgRate: -1.00 Mbytes/sec, peakRate: 0.00 Mbytes/sec, bucketLimit: 0.00 Mbytes, throttlerLogTimeMillis: 0
I0316 18:42:13.633317 3176 Receiver.cpp:395] wdt> Progress reporter updating every 20 ms
I0316 18:42:13.633317 3176 Receiver.cpp:395] wdt> Progress reporter updating every 20 ms
[ ] 0% 0.0 0.0 Mbytes/s

传送端：

root@ubuntu:~# time wdt -directory /root/test -destination 10.129.83.191 -transfer_id 1 --ipv6=false

开始发送一个 5GB 的目录传输

I0316 18:44:24.157038 3480 WdtFlags.cpp:63] wdt> Running WDT 1.29.1703140 p 29
I0316 18:44:24.157658 3480 EncryptionUtils.cpp:83] wdt> Openssl library initialized
I0316 18:44:24.157686 3480 WdtFlags.cpp:63] wdt> Running WDT 1.29.1703140 p 29
I0316 18:44:24.158160 3480 WdtResourceController.cpp:29] wdt> Updated max number of senders for _root controller_ to 0
I0316 18:44:24.158180 3480 WdtResourceController.cpp:22] wdt> Updated max number of

receivers for _root controller_ to 0
I0316 18:44:24.158208 3480 Throttler.cpp:73] wdt> No average rate specified
I0316 18:44:24.158222 3480 Throttler.cpp:80] wdt> No peak rate specified
I0316 18:44:24.158241 3480 Wdt.cpp:24] wdt>One time initialization of WDT for wdt
I0316 18:44:24.158258 3480 Throttler.cpp:98] wdt> Updating the rates avgRateBytesPerSec : -1048576.00 bucketRateBytesPerSec : 0.00 bytesTokenBucketLimit : 0.00
I0316 18:44:24.158301 3480 wdtCmdLine.cpp:365] wdt> Making Sender with encryption set = 0
I0316 18:44:24.158323 3480 WdtResourceController.cpp:357] wdt>First time (default) is seen, creating.
I0316 18:44:24.158339 3480 WdtResourceController.cpp:29] wdt> Updated max number of senders for to 0
I0316 18:44:24.158351 3480 WdtResourceController.cpp:22] wdt> Updated max number of receivers for to 1
I0316 18:44:24.158394 3480 Throttler.cpp:98] wdt> Updating the rates avgRateBytesPerSec : -1048576.00 bucketRateBytesPerSec : 0.00 bytesTokenBucketLimit : 0.00
I0316 18:44:24.158429 3480 Sender.cpp:47] wdt> WDT Sender 1.29.1703140 p 29
I0316 18:44:24.158458 3480 WdtResourceController.cpp:379] wdt>Successfully added a sender for identifier 10.129.83.191
I0316 18:44:24.158479 3480 WdtBase.cpp:94] wdt> using wdt protocol version 29
I0316 18:44:24.158512 3480 DirectorySourceQueue.cpp:127] wdt> Root dir now /root/test/
I0316 18:44:24.158530 3480 Sender.cpp:311] wdt> Client (sending) to 10.129.83.191, Using ports [22356 22357 22358 22359 22360 22361 22362 22363]
I0316 18:44:24.158552 3480 Sender.cpp:325] wdt> Skipping throttler setup. External throttler set.Throttler details : avgRate: -1.00 Mbytes/sec, peakRate: 0.00 Mbytes/sec, bucketLimit: 0.00 Mbytes, throttlerLogTimeMillis: 0
I0316 18:44:24.158820 3481 DirectorySourceQueue.cpp:240] wdt> Exploring root dir /root/test/ include_pattern : exclude_pattern : prune_dir_pattern :
I0316 18:44:24.158884 3482 Sender.cpp:41] wdt> Starting a new transfer 1 to 10.129.83.191
I0316 18:44:24.159659 3482 SenderThread.cpp:84] wdt> Thread[0, port: 22356] Connection took 1 attempt(s) and 0.00 seconds. port 22356
I0316 18:44:24.159711 3484 SenderThread.cpp:84] wdt> Thread[2, port: 22358] Connection took 1 attempt(s) and 0.00 seconds. port 22358
I0316 18:44:24.159713 3483 SenderThread.cpp:84] wdt> Thread[1, port: 22357] Connection took 1 attempt(s) and 0.00 seconds. port 22357
I0316 18:44:24.159941 3481 DirectorySourceQueue.cpp:389] wdt> Number of files explored: 8 opened 0 with direct 0 errors false
I0316 18:44:24.159942 3485 SenderThread.cpp:84] wdt> Thread[3, port: 22359] Connection took 1 attempt(s) and 0.00 seconds. port 22359
I0316 18:44:24.159997 3487 SenderThread.cpp:84] wdt> Thread[5, port: 22361] Connection took 1 attempt(s) and 0.00 seconds. port 22361
I0316 18:44:24.160032 3490 Sender.cpp:418] wdt> Progress reporter tracking every 20 ms
I0316 18:44:24.159999 3486 SenderThread.cpp:84] wdt> Thread[4, port: 22360] Connection took 1 attempt(s) and 0.00 seconds. port 22360
I0316 18:44:24.160115 3488 SenderThread.cpp:84] wdt> Thread[6, port: 22362] Connection

took 1 attempt(s) and 0.00 seconds. port 22362

I0316 18:44:24.160159 3489 SenderThread.cpp:84] wdt> Thread[7, port: 22363] Connection took 1 attempt(s) and 0.00 seconds. port 22363

[==========] 100% 187.4 Mbytes/s

I0316 18:44:50.581629 3482 SenderThread.cpp:968] wdt> Thread[0, port: 22356] Port 22356 done. Transfer status = OK. Number of blocks transferred = 62. Data Mbytes = 987.00. Header Kbytes = 3.44 (0.00% overhead). Total bytes = 1034948037. Wasted bytes due to failure = 0 (0.00% overhead). Encryption type = none. Total throughput = 37.35 Mbytes/sec

I0316 18:44:50.581710 3483 SenderThread.cpp:968] wdt> Thread[1, port: 22357] Port 22357 done. Transfer status = OK. Number of blocks transferred = 42. Data Mbytes = 667.00. Header Kbytes = 2.42 (0.00% overhead). Total bytes = 699402669. Wasted bytes due to failure = 0 (0.00% overhead). Encryption type = none. Total throughput = 25.24 Mbytes/sec

I0316 18:44:50.581640 3487 SenderThread.cpp:968] wdt> Thread[5, port: 22361] Port 22361 done. Transfer status = OK. Number of blocks transferred = 34. Data Mbytes = 544.00. Header Kbytes = 2.00 (0.00% overhead). Total bytes = 570427393. Wasted bytes due to failure = 0 (0.00% overhead). Encryption type = none. Total throughput = 20.59 Mbytes/sec

I0316 18:44:50.581730 3488 SenderThread.cpp:968] wdt> Thread[6, port: 22362] Port 22362 done. Transfer status = OK. Number of blocks transferred = 31. Data Mbytes = 491.00. Header Kbytes = 1.85 (0.00% overhead). Total bytes = 514852712. Wasted bytes due to failure = 0 (0.00% overhead). Encryption type = none. Total throughput = 18.58 Mbytes/sec

I0316 18:44:50.581732 3484 SenderThread.cpp:968] wdt> Thread[2, port: 22358] Port 22358 done. Transfer status = OK. Number of blocks transferred = 29. Data Mbytes = 459.00. Header Kbytes = 1.75 (0.00% overhead). Total bytes = 481298175. Wasted bytes due to failure = 0 (0.00% overhead). Encryption type = none. Total throughput = 17.37 Mbytes/sec

I0316 18:44:50.581760 3489 SenderThread.cpp:968] wdt> Thread[7, port: 22363] Port 22363 done. Transfer status = OK. Number of blocks transferred = 29. Data Mbytes = 454.00. Header Kbytes = 1.74 (0.00% overhead). Total bytes = 476055288. Wasted bytes due to failure = 0 (0.00% overhead). Encryption type = none. Total throughput = 17.18 Mbytes/sec

I0316 18:44:50.581761 3485 SenderThread.cpp:968] wdt> Thread[3, port: 22359] Port 22359 done. Transfer status = OK. Number of blocks transferred = 49. Data Mbytes = 779.00. Header Kbytes = 2.77 (0.00% overhead). Total bytes = 816843537. Wasted bytes due to failure = 0 (0.00% overhead). Encryption type = none. Total throughput = 29.48 Mbytes/sec

I0316 18:44:50.581737 3486 SenderThread.cpp:968] wdt> Thread[4, port: 22360] Port 22360 done. Transfer status = OK. Number of blocks transferred = 36. Data Mbytes = 571.00. Header Kbytes = 2.11 (0.00% overhead). Total bytes = 598739054. Wasted bytes due to failure = 0 (0.00% overhead). Encryption type = none. Total throughput = 21.61 Mbytes/sec

I0316 18:44:50.581972 3486 Sender.cpp:28] wdt> Last thread finished 26.42 for transfer id 1

I0316 18:44:50.582399 3480 Reporting.cpp:129] wdt> Error code summary OK

[==========] 100% 187.4 Mbytes/s

I0316 18:44:50.582440 3480 Sender.cpp:260] wdt> Total sender time = 26.42 seconds (0.00 dirTime). Transfer summary : Transfer status = OK. Number of files transferred = 8. Data Mbytes = 4952.00. Header Kbytes = 18.08 (0.00% overhead). Total bytes = 5192566865. Wasted bytes due to failure = 0 (0.00% overhead). Encryption type = none. Previously sent bytes : 0.

wdt> ***Total sender throughput = 187.41 Mbytes/sec (187.42 Mbytes/sec pure***

transfer rate)

I0316 18:44:50.582490 3480 WdtResourceController.cpp:154] wdt>Released the sender with id 1
I0316 18:44:50.582502 3480 Wdt.cpp:125] wdt> wdtSend for 10.129.83.191 ended with OK
I0316 18:44:50.582870 3480 wdtCmdLine.cpp:372] wdt> Returning with OK exit code
I0316 18:44:50.582890 3480 WdtResourceController.cpp:294] wdt>Shutting down the controller (0 senders 0 receivers)

real 0m26.431s
user 0m0.148s
sys 0m4.016s

传输速度达到了 187MB/s

同样环境 SCP 操作：

root@ubuntu:~# time scp /root/test/ 10.129.83.191:/home/ubuntu/test/*
root@10.129.83.191's password:
ubuntu-14.04.5-server-amd64.iso
100% 619MB 103.2MB/s 00:06
ubuntu-14.04.5-server-amd64.iso1
100% 619MB 103.2MB/s 00:06
ubuntu-14.04.5-server-amd64.iso2
100% 619MB 88.4MB/s 00:07
ubuntu-14.04.5-server-amd64.iso3
100% 619MB 88.4MB/s 00:07
ubuntu-14.04.5-server-amd64.iso4
100% 619MB 103.2MB/s 00:06
ubuntu-14.04.5-server-amd64.iso5
100% 619MB 88.4MB/s 00:07
ubuntu-14.04.5-server-amd64.iso6
100% 619MB 88.4MB/s 00:07
ubuntu-14.04.5-server-amd64.iso7
100% 619MB 88.4MB/s 00:07

real 0m57.561s
user 0m23.464s
sys 0m27.048s

平均速度在 100MB/s 左右，时间差了接近两倍。

但是在多次测试的过程中发现，SCP 总是可以保持一个稳定的传输速度，而 WDT

在多次中出现了速度的较大波动，总体来说 SCP 的稳定性远远高于 WDT。

WDT 在数据传输中必须在接收端开启接收进程，这在一个整体传输很频繁的系统中无疑增加了维护的难度（例如备份系统），而且 WDT 的稳定性目前存在问题，并且最严重的是其对系统本身依赖过大，目前只在 Ubuntu 系统编译成功，而对于目前使用很普及的 CentOS、RedHat 系统，很多 LIB 库需要升级才能够编译成功（官方本身的最高版本并不符合编译要求），所以对于大规模部署 WDT 来说是不现实的。

WDT 在超远距离的传输中可能会发挥比较大的作用，对于单次传输一个超大的文件来说，追求极限速度是毫无疑问的，但在目前整体 IT 大环境中，万兆网卡还远远没有普及，所以在多块千兆网卡做 bond 的情况下，单块网卡的最高带宽可能是一个潜在的瓶颈。

结论：

目前的系统环境不存在大量部署 WDT 的条件（备份系统的端对端的传输瓶颈可能出现在单块网卡上），同时 WDT 繁琐的编译步骤也是让人崩溃的。