HTTP BitTorrent tracker performance comparison

Summary

HTTP BitTorrent tracker implementations **aquatic_http**, **opentracker** and **chihaya** were load tested with **aquatic_http_load_test** to measure throughput with different numbers of allotted hyperthreads. Connections were made over TLS with TCP keep alive turned off. Throughput differences were not very large, but aquatic_http came in first (31k responses a second with 16 hyperthreads), opentracker second (29k) and chihaya third (25k).

Setup

Tested tracker implementations

Tracker	URL	Commit
aquatic_http	https://github.com/greatest-ape/aquatic	d8a2e7f
opentracker *	http://erdgeist.org/arts/software/opentracker/	110868e
Chihaya	https://github.com/chihaya/chihaya	7455c2a
hefur †	https://github.com/abique/hefur	a3915fd

* Since opentracker doesn't support TLS, it was configured to run behind high-performance reverse TLS proxy hitch (https://hitch-tls.org/).
† Hefur could not be benchmarked since it only returned error responses with message "torrent not found", even if configured not to check a directory for allowed torrent files.

Settings

- Configuration files are included in appendix
- Connections were made over TLS 1.3
- TCP keep alive was turned off, since it is what public trackers will likely want to do
- I limited processes to virtual CPUs because chihaya doesn't support setting number of workers
- 64 load test workers were used, since this number was observed to cause greater load than other tested amounts
- Load tester was limited to virtual CPUs 16-47 using taskset
- Trackers were limited to virtual CPUs 0-N using taskset, where N was number of allotted hyperthreads minus one. Thread pinning was attempted, but didn't improve performance.
- When benchmarking opentracker, I tried starting hitch with both the same number of workers as number of allotted hyper thread and with one less, and kept the highest throughput data. Using the same number performed better except in the case of 16 hyperthreads, in which case using 15 workers performed better.

Hardware

Hetzner CCX62: 48 dedicated vCPUs (24 cores, AMD Milan Epyc 7003)

Software information

Software	Version
Ubuntu	20.04
Linux	5.15.0
rustc	1.60.0
GCC	9.4.0
golang	1.18
hitch	1.5.2

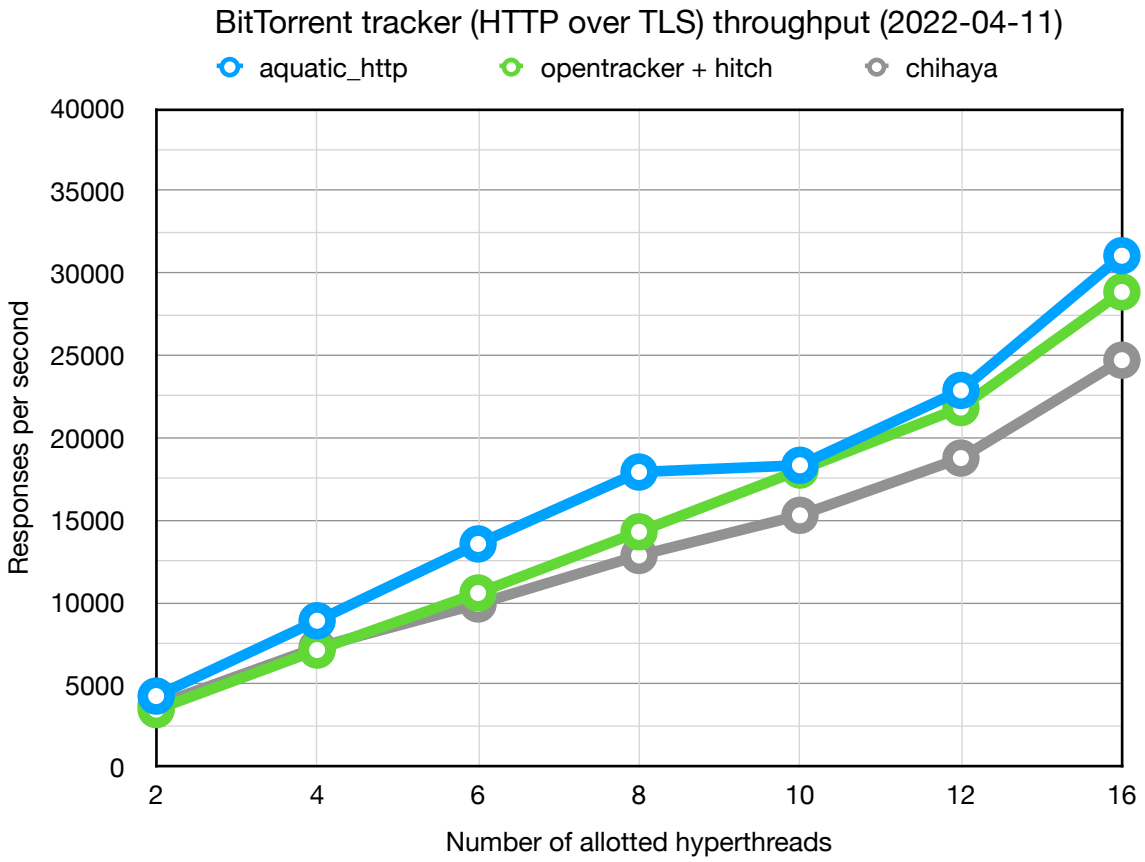
Before building opentracker, enable DWANT_IP_FROM_PROXY. Also, tell compiler to optimise for current CPU by running:

sed -i "s/^OPTS_production=-O3/OPTS_production=-O3 -march=native -mtune=native/g" Makefile

Results

BitTorrent tracker (HTTP over TLS) throughput (2022-04-11)

	Responses per second, by tracker		
Hyperthreads	**aquatic_http**	**opentracker + hitch**	**chihaya**
2	**4313**	3492	3757
4	**8888**	7107	7271
6	**13554**	10573	9906
8	**17921**	14289	12853
10	**18332**	18040	15293
12	**22879**	21835	18768
16	**31059**	28859	24713



Approximate per-thread CPU utilization with 16 hyperthreads

	aquatic_udp	opentracker + hitch	chihaya
Tracker CPU utilization	85 %	99,5 %	98 %
Load test CPU utilization	65 %	55 %	43 %

Observations

- Performance differences are not that large
- TLS handling dominates CPU time when TCP keepalive is turned off
- *aquatic_http_load_test* should be better optimised in order to achieve same load while using less hyperthreads. This would enable benchmarking trackers up to higher hyper thread counts. It might also enable pushing *aquatic_http* tracker per-thread CPU usage closer to 100%, which could increase its performance advantage.
- *opentracker* performs very well with a single worker, but it also offloads most work to hitch (note that *aquatic_http* is also configured to only use a single request worker). However, being single-threaded might not scale well with more hyperthreads.
- For this use case, using hitch to handle most connection work and writing a fast, single threaded tracker will take you very far

Appendix: configuration files

aquatic_http_load_test

```
server_address = "127.0.0.1:3000"
log_level = "error"
num_workers = 64
num_connections = 1024
connection_creation_interval_ms = 0
duration = 60
keep_alive = false

[torrents]
number_of_torrents = 10000
torrent_selection_pareto_shape = 2.0
peer_seeder_probability = 0.25
weight_announce = 99
weight_scrape = 0
```

aquatic_http

Default settings were used, except that TLS was set up, keep_alive was set to false and cleaning intervals were set to 600 seconds.

Chihaya

```
---
chihaya:
  announce_interval: "30m"
  min_announce_interval: "15m"
  http:
    addr: "127.0.0.1:3001"
    https_addr: "127.0.0.1:3000"
    tls_cert_path: "cert.crt"
    tls_key_path: "key.pem"
    read_timeout: "5s"
    write_timeout: "5s"
    enable_keepalive: false
    idle_timeout: "30s"
    enable_request_timing: false
    announce_routes:
      - "/announce"
    scrape_routes:
      - "/scrape"
    allow_ip_spoofing: false
    real_ip_header: "x-real-ip"
    max_numwant: 100
    default_numwant: 50
    max_scrape_infohashes: 50

  storage:
    name: "memory"
    config:
      gc_interval: "10m"
      peer_lifetime: "31m"
      shard_count: 1024
      prometheus_reporting_interval: "10m"
  prehooks:
```

opentracker

```
listen.tcp 127.0.0.1:3001
```

hitch

```
backend = "[127.0.0.1]:3001"
frontend = "[127.0.0.1]:3000"

keepalive = 0
workers = 16
user = "nobody"
group = "nogroup"

pem-file = {
    cert = "cert.crt"
    private-key = "key.pem"
}
```