HTTP BitTorrent tracker performance comparison

This is a performance comparison of several HTTP BitTorrent tracker implementations, using *aquatic_http_load_test*.

Setup

Tested tracker implementations

Tracker	URL	Commit
aquatic_http	https://github.com/greatest-ape/aquatic	5ac8f37
chihaya	https://github.com/chihaya/chihaya	acf2a5a
opentracker *	http://erdgeist.org/arts/software/opentracker/	110868e

* Since opentracker doesn't support TLS, it was configured to run behind Hitch (https://hitch-tls.org/), a high performance TLS proxy

General information

- Connections were made over TLS 1.3
- TCP keepalive was turned off, since that's what public trackers will likely want to do
- Processes were limited to virtual CPUs corresponding to cores
- 64 load test workers were used, since this caused highest load among tested numbers

Hardware

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

Software versions

Software	Version
Debian	Bullseye
Linux	6.0.0
rustc	1.66.1
Go	1.19.5
GCC	10.2.1
Hitch	1.7.3

Configuration files and build instructions

Generate TLS certificates by going to aquatic source directory and running:

```
./scripts/gen-tls.sh
```

Before running each applications, run:

```
ulimit -n 1024000
```

aquatic_http_load_test

```
num_workers = 64
num_connections = 1024
connection_creation_interval_ms = 1
duration = 60
keep_alive = false
```

Run with:

```
taskset -c 12-23,36-47 ./scripts/run-load-test-http.sh -c tmp/load-test-http.toml
```

aquatic

```
[network]
address = "0.0.0.0:3000"
only_ipv6 = false
tcp_backlog = 1024
tls_certificate_path = "./tmp/tls/cert.crt"
tls_private_key_path = "./tmp/tls/key.pk8"
keep_alive = false

[protocol]
max_peers = 50

[cleaning]
torrent_cleaning_interval = 180
connection_cleaning_interval = 180
```

opentracker

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

Before building, run:

```
sed -i "s/^OPTS_production=-O3/OPTS_production=-O3 -march=native -mtune=native/g"
Makefile
sed -i "s/if( numwant > 200 ) numwant = 200/if( numwant > 50 ) numwant = 50/g" ot_http.c
```

hitch

```
frontend = {
    host = "127.0.0.1"
    port = "3000"
}
backend = "[127.0.0.1]:3001"
workers = 8
daemon = off
pem-file = {
        cert = “./tmp/tls/cert.crt”
        private-key = “./tmp/tls/key.pem"
}
user = "hitch"
group = “hitch"
```

Hitch will by default bind to hyperthreads itself, which circumvents using taskset. To prevent this, before building it, run:

```
sed -i "s/#if defined(CPU_ZERO) && defined(CPU_SET)/#if 0/g" ./src/hitch.c
```

chihaya

```
---
chihaya:
  announce_interval: "30m"
  min_announce_interval: "15m"
  http:
    addr: "0.0.0.0:6969"
    https_addr: "127.0.0.1:3000"
    tls_cert_path: “./tmp/tls/cert.crt"
    tls_key_path: “./tmp/tls/key.pk8"
    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: 50
    default_numwant: 50
    max_scrape_infohashes: 50

  storage:
    name: "memory"
    config:
      gc_interval: "3m"
      peer_lifetime: "31m"
      shard_count: 1024
      prometheus_reporting_interval: "1s"

  prehooks:
```

Measurements

Results are ordered ascendingly by 1) allotted cores and 2) number of responses per second.
Best results within a core number tier are marked in bold.

aquatic_http

CPU cores	taskset	Socket workers	Swarm workers	Responses per second
1	**0,24**	**1**	**1**	**4830**
2	**0-1,24-25**	**2**	**1**	**8882**
4	**0-3,24-27**	**4**	**1**	**17971**
6	0-5,24-29	5	1	21764
6	**0-5,24-29**	**6**	**1**	**26566**
8	**0-7,24-31**	**8**	**1**	**35778**
10	**0-9,24-33**	**10**	**1**	**37671**
12	0-11,24-35	11	1	39330
12	**0-11,24-35**	**12**	**1**	**40351**

Chihaya

CPU cores	taskset	Responses per second
1 *	**0,24**	**5099**
2 *	**0-1,24-25**	**7017**
4	**0-3,24-27**	**13524**
6	**0-5,24-29**	**19605**
8	**0-7,24-31**	**24788**
10	**0-9,24-33**	**31767**
12	**0-11,24-35**	**35926**

* reduced number of load test workers to 12 due to load tester error messages about opening connections

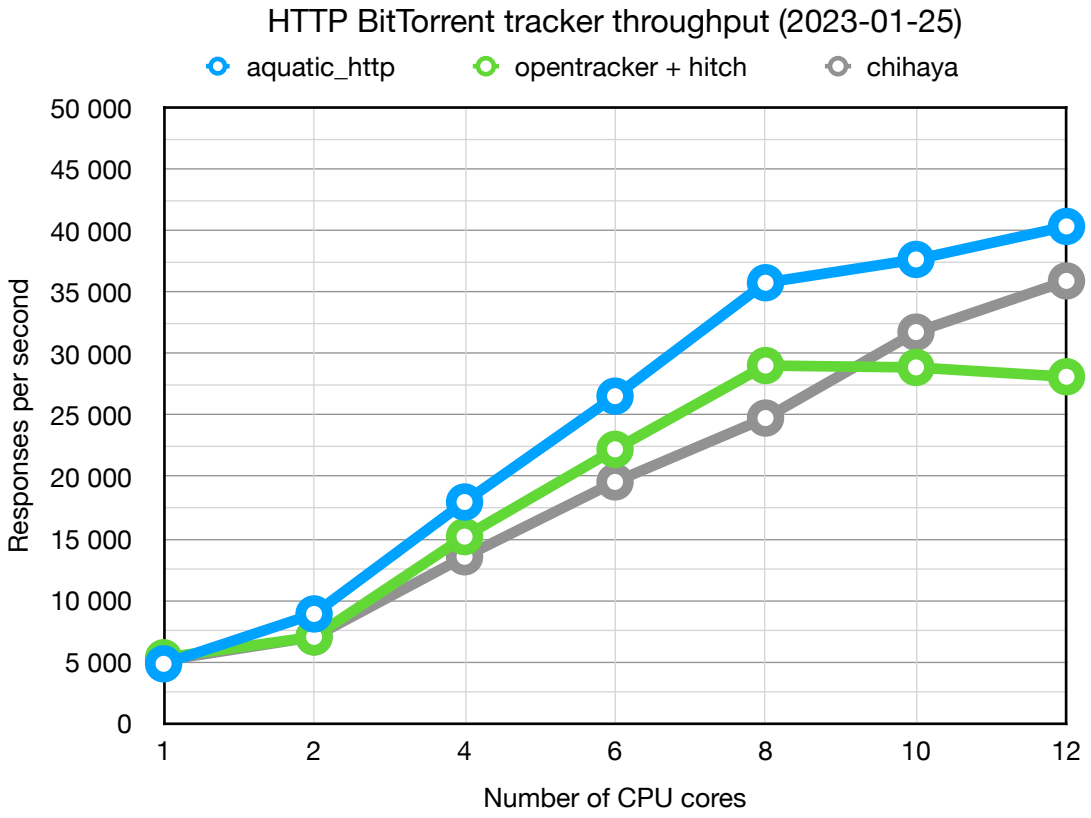
opentracker

CPU cores	taskset	hitch workers	Responses per second
1 *	0,24	1	3047
1 *	**0,24**	**2**	**5323**
2 *	0-1,24-25	2	4027
2 *	**0-1,24-25**	**4**	**7020**
4	0-3,24-27	4	7890
4	**0-3,24-27**	**8**	**15160**
6	0,24; 1-5,25-29 †	10	21331
6	**0-5,24-29**	**12**	**22247**
8	0-7,24-31	16	28034
8	**0,24; 1-7,25-31 †**	**14**	**29052**
10	0-9,24-33	20	26800
10	**0,24; 1-9,25-33 †**	**18**	**28901**
12	0-11,24-35	24	27397
12	**0,24; 1-11,25-35 †**	**22**	**28142**

* reduced number of load test workers to 12 due to load tester error messages about opening connections

† separate hyperthread pinning for opentracker and hitch

Results



HTTP BitTorrent tracker throughput (2023-01-25)

	Responses per second		
CPU cores	aquatic_udp	opentracker + hitch	chihaya
1	4830	**5323**	5099
2	**8882**	7020	7017
4	**17971**	15160	13524
6	**26566**	22247	19605
8	**35778**	29052	24788
10	**37671**	28901	31767
12	**40351**	28142	35926

Observations

- TLS handling dominates CPU time when TCP keep alive is turned off
- Writing a fast single-threaded plain HTTP tracker (such as opentracker) and offloading TLS handling to a fast proxy will take you very far