Joakim Frostegård, 2021-11-28

BitTorrent (UDP) tracker comparison

This is a performance comparison of two programs implementing the UDP BitTorrent tracker protocol. aquatic_udp_load_test (https://github.com/greatest-ape/aquatic) was used to generate requests and count responses.

Setup

Tested trackers

Tracker	URL	Commit
aquatic_udp	https://github.com/greatest-ape/aquatic	14474fb
opentracker	http://erdgeist.org/arts/software/opentracker/	110868e

Hardware

Hetzner CCX62: 48 vCPUs (AMD Milan Epyc 7003)

Software

Software	Version
Debian	Bullseye
Linux	5.14.0 sysctl -w net.core.rmem_max=104857600 sysctl -w net.core.rmem_default=104857600
rustc	1.56.1
gcc	10.2.1
aquatic	Run with ./scripts/run-aquatic-udp.sh
opentracker	Before building, run: sed -i "s/^OPTS_production=-O3/OPTS_production=-O3 -march=native -mtune=native/g" Makefile

The default load test configuration was used, except that *workers* was set to 14, *duration* to 60 and *additional_request_probability* to 0.5.

Default tracker settings were used, expect that aquatic_udp *recv_buffer* was set to 104857600 and opentracker was configured only to listen to udp.

Joakim Frostegård, 2021-11-28

Measurements

Best results per total worker tier are marked in bold.

aquatic_udp

aquatic_udp throughput with additional_request_probability = 0.5

aquatic_udp total workers	aquatic_udp socket workers	aquatic_udp request workers	responses per second (all types)	responses per second (announce)
2	1	1	**255k**	**145k**
3	2	1	**503k**	**293k**
4	3	1	**697k**	**410k**
5	3	2	728k	442k
5	4	1	**792k**	**448k**
6	5	1	742k	428k
6	4	2	**951k**	**597k**
7	5	2	**1151k**	**738k**
8	5	3	1119k	718k
8	6	2	**1124k**	**746k**
10	6	4	1091k	726k
10	8	2	1191k	791k
10	7	3	**1335k**	**906k**
12	8	4	1306k	881k
12	9	3	**1482k**	**1026k**
14	10	4	1314k	886k
14	11	3	**1590k**	**1113k**
16	13	3	1567k	1097k
16	12	4	**1588k**	**1111k**

opentracker

opentracker throughput with additional_request_probability = 0.5

opentracker workers	responses per second (total)	responses per second (announce)	notes
1	187k	117k	event mode
1	**241k**	**154k**	
2	**259k**	**173k**	
3	**396k**	**268k**	
4	**460k**	**314k**	
5	**517k**	**353k**	
6	**598k**	**409k**	
7	**723k**	**503k**	
8	**745k**	**510k**	
10	**826k**	**562k**	
12	**926k**	**628k**	
14	**875k**	**584k**	
16	**972k**	**655k**	

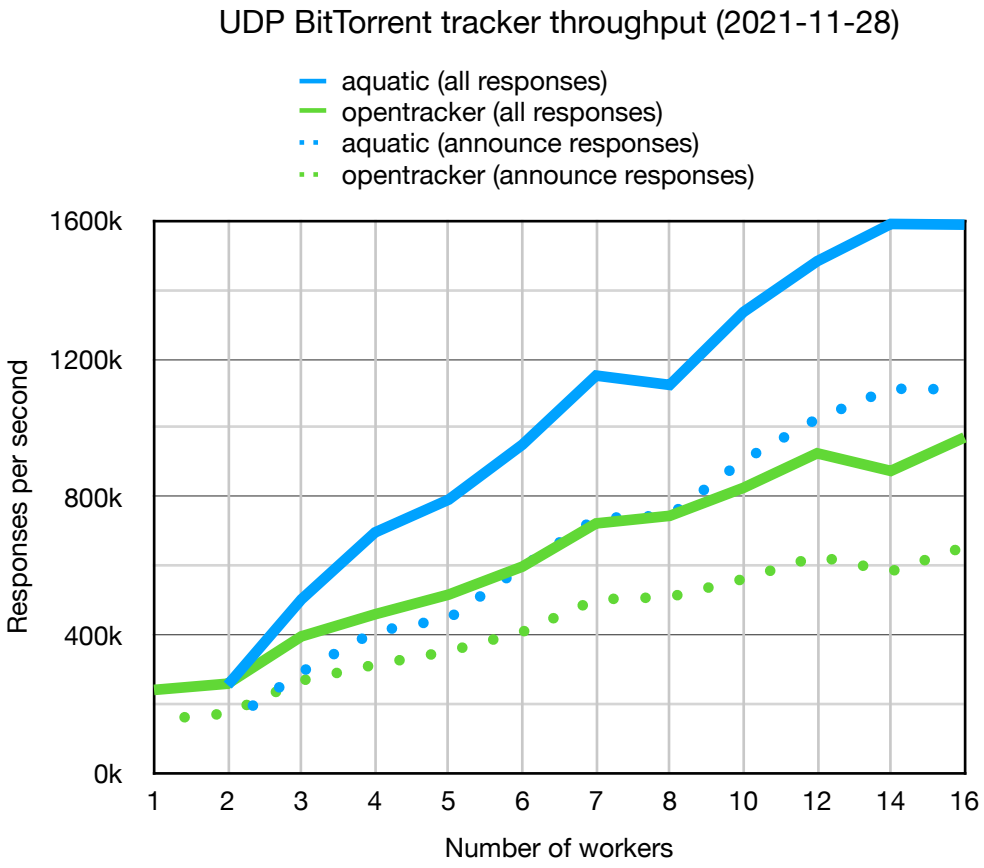
I saw that with aquatic_load_test setting *additional_request_probabilty* set to to 0.5, opentracker tended to process a high proportion of typically more expensive announce requests compared to connect requests. For this reason, I tried changing the setting to 1.0 to achieve a proportion more similar to with aquatic. This didn't significantly help performance and I didn't use the data for the final comparison, but I include it below for reference.

opentracker throughput with additional_request_probability = 1.0

opentracker workers	responses per second (all types)	responses per second (announce)	notes
1	**230k**	**114k**	event mode
1	228k	140k	
2	**259k**	**150k**	
3	**366k**	**216k**	
4	**487k**	**294k**	
5	**539k**	**320k**	
6	**596k**	**360k**	

opentracker workers	responses per second (all types)	responses per second (announce)	notes
7	676k	407k	
8	822k	503k	
10	859k	518k	
12	875k	515k	
14	946k	558k	
16	791k	444k	far from full CPU utilisation

Results



- when using 3 or more workers, aquatic outperforms opentracker in throughput
- aquatic throughput peaks at around 1.6 million responses per second with 14 workers
- opentracker throughput peaks at around 1 million responses per second with 16 workers

Joakim Frostegård, 2021-11-28

UDP BitTorrent tracker throughput (2021-11-28)

Number of workers	aquatic_udp (all responses)	opentracker (all responses)	aquatic_udp (announce responses)	opentracker (announce responses)
1	n/a	**241**	n/a	**154**
2	255	**259**	145	**173**
3	**503**	**396**	**293**	**268**
4	**697**	**460**	**410**	**314**
5	**792**	**517**	**448**	**353**
6	**951**	**598**	**597**	**409**
7	**1151**	**723**	**738**	**503**
8	**1124**	**745**	**746**	**510**
10	**1335**	**826**	**906**	**562**
12	**1482**	**926**	**1026**	**628**
14	**1590**	**875**	**1113**	**584**
16	**1588**	**972**	**1111**	**655**

Results are shown in thousands of responses per second. The best result for each number of workers is marked in bold.