ARDOR Load Test 2019

Jelurida Swiss SA
www.jelurida.com

Abstract: We have conducted load testing of the Ardor blockchain software in a test environment in order to evaluate the maximum sustainable transaction throughput of the system and the performance of the network and fork resolution algorithm under high transaction load. A steady throughput of above 100 TPS for more than two weeks has been achieved, and 2 second block times are feasible although not really necessary.

1. Hardware setup

We used Shuttle DH310 servers, configured with Intel i7-8700 3.2 GHz 6-core CPUs, 16 GB SO-DDR4 2666MHz RAM, and Samsung 970 PRO solid state drives, costing around 1,000 CHF total each.

All servers were running FreeBSD 12.0, using UFS filesystem, with OpenJDK 1.8.0_202 compiled from source. The servers were connected to a gigabit router, and the network was tested to sustain more than 932 Mbits/s and 0.220 ms ping times.

2. Software modifications

A modified version of the Ardor software, as of release 2.2.2 with some optimizations and bugfixes (all of which are now included in 2.2.3) was prepared for the loadtest.

The following changes were made, compared to the public Ardor blockchain version:

- A new genesis block was created, with 10 hardcoded test accounts, having the numbers 0 to 9 as passwords, and an initial balance of 100,000,000 on each chain. Four child chains were used, as in the public Ardor blockchain.
- The desktop application was disabled, but the browser-based UI was regularly used to monitor the current blockchain state on each server.
- The Bundler code was modified to perform bundling only before a block is generated, instead of every time a new unconfirmed transaction is received since this was identified as a considerable performance bottleneck.

The following code constants and parameters were modified in order to increase (and practically remove) the hardcoded limits on transaction counts, sizes, and network limits:

Parameter	Used in test	Production default
Constants.MAX_NUMBER_OF_FXT_TRANSACTIONS	10000	10
Constants.MAX_NUMBER_OF_CHILD_TRANSACTIONS	10000	100
Constants.MAX_CHILDBLOCK_PAYLOAD_LENGTH	10000*1024	128*1024
NetworkMessage.MAX_ARRAY_LENGTH	Integer.MAX_VALUE	48*1024
NetworkMessage.MAX_LIST_SIZE	Integer.MAX_VALUE	1500
NetworkHandler.MAX_PENDING_MESSAGES	250	25
nxt.forgingDelay	0	10
nxt.batchCommitSize	10	100
nxt.blacklistingPeriod	60	600
nxt.maxUnconfirmedTransactions	36000	2000
nxt.testnetNumberOfForkConfirmations	0	5
getMoreBlocksThread ran every (sec.)	2	5

- The Constants.TESTNET_ACCELERATION parameter was varied as needed to achieve block times of 2, 5, or 10 seconds.
- The H2 database was started with MVCC=TRUE;MV_STORE=TRUE (now the default since Ardor 2.2.3).

3. Loadtest script parameters

The modified Ardor software was installed on four identical servers, starting with an empty blockchain. Forging and bundling was configured to run with several accounts as follows:

Server	Forging accounts	Bundling accounts (each bundling on all chains)
Voyager	0, 1, 2	0, 1, 2
Enterprise	3, 4, 5	3, 4, 5
Discovery	6, 7	6, 7
Defiant	8, 9	8, 9

The JMeter script was run from a separate server with similar hardware configuration. The script was submitting ordinary payment transactions on all child chains as well as on the parent chain, randomly from all test accounts to all servers, using a predefined number of threads for each run. The following configuration files were used by the script:

```
servers.csv – lists the remote server addresses.
chains.csv – lists the chain properties [chain id, default transaction fee].
accounts.csv – lists sender [secret phrase, recipient address].
```

The JMeter script iterated over these CSV files sequentially wrapping around when reaching the end of a file. This created an even distribution of requests between servers, across chains and between sender and recipient account pairs. Distributing the transactions to different sender and recipient accounts is important for maintaining a realistic scenario and for not overloading the transaction table indexes.

To prevent duplicate transactions, a counter was maintained by the script, the transaction amount field was set to the counter value thus preventing duplicate transactions. To prevent accumulation of waiting threads on the script side when a server is busy, sleep times were randomized between 0 and 200 milliseconds.

4. Initial runs

A number of 1-2 hour test runs were conducted to evaluate the system behavior with different block times and number of JMeter threads:

Threads	Blocktime	Time	Tx Count	TPS	Blockheight
40	5	3600	432,200	120	655
40	2	3600	465,400	129	1570
50	10	5200	717,500	137	470
50	5	3600	520,100	144	670
50	2	3600	540,700	150	1524
60	5	3600	421,000	117	680
60	2	2800	418,200	149	1180

Increasing the number of threads for the loadtest beyond 60 started to only reduce the TPS achieved. While reducing block times from 10s to 5s and 2s resulted in some increase of the transaction throughput, that increase was certainly not proportional to the increased number of blocks. Setting the blocktime to 20 s lead to an accumulation of too many unconfirmed transactions in memory which started to be automatically discarded by the system as designed, therefore blocktimes were kept at 10 s or below in all test runs.

5. Extended duration loadtest

Based on the above results, we embarked on a long duration loadtest, to probe how sustainable this performance is as the total number of transactions and database size keeps growing. We used a setting of 60 threads in the JMeter script, and a blocktime of 2 seconds. The total transaction count and average TPS in the first hours were as follows:

Time	Tx count	TPS	Blockheight	db size
3780	590 600	156	1626	1 GB
38000	5,162,000	136	16100	8 GB
43000	5,663,700	132	18220	9 GB

The servers were left running. The transaction throughput was checked daily and found to slowly drop further below 130 TPS, but consistently exceeding 100 TPS for the first 10 days.

During all loadtest runs, we observed the regular formation of forks of up to 20-30 blocks, especially with the short 2 s block times. Forks, however, always resolved and all four servers remained in eventual consensus.

The JMeter log and the server logs were regularly monitored to ensure the number of errors remained insignificant and all transactions submitted were being processed without a transaction loss.

After **17 days** (1,510,000 s), the database size had reached more than **170 GB**, and the test had to be stopped as the servers were about to run out of disk space. The total transaction count from all chains was 150,760,898, resulting in an overall average of **99.84 transactions per second**. The total number of blocks was 641,270, for an average of **2.35 s block time**. For comparison, as of 21.05.2019 the Bitcoin blockchain for its 10 years, 5 months of existence has processed 415 M transactions, is currently at block 576942, and its full database requires 220 GB.

It should be noted that shutting down and restarting the servers resulted in a significant reduction of database size, from 170 GB down to 30 GB. This can be attributed to the H2 database only being able to perform a full "compact" (an internal database operation which discards obsolete records) on shutdown. For a production system running at such transaction volume, an investigation of whether a more efficient database compaction can be performed at runtime without requiring server restarts should be undertaken. We should also mention that applying our existing compact.sh script (which dumps and reloads the database content from an sql backup file) to the already compacted database actually resulted in an increase of the database size to 50 GB, only to drop back to 30 GB after another server restart cycle.

6. Performance analysis

After the end of the loadtest run, the following numbers were obtained by querying the blockchain database directly:

```
Total number of transactions on all chains:   150,760,898
Total runtime:                                1,509,958 s (17.5 days)
Total number of blocks:                       641,270
Total number of parent chain transactions[1]: 35,296,971
Total number of child chain blocks:           1,950,914
Total number of child chain transactions:     113,513,013
```

From those, the following averages can be calculated:

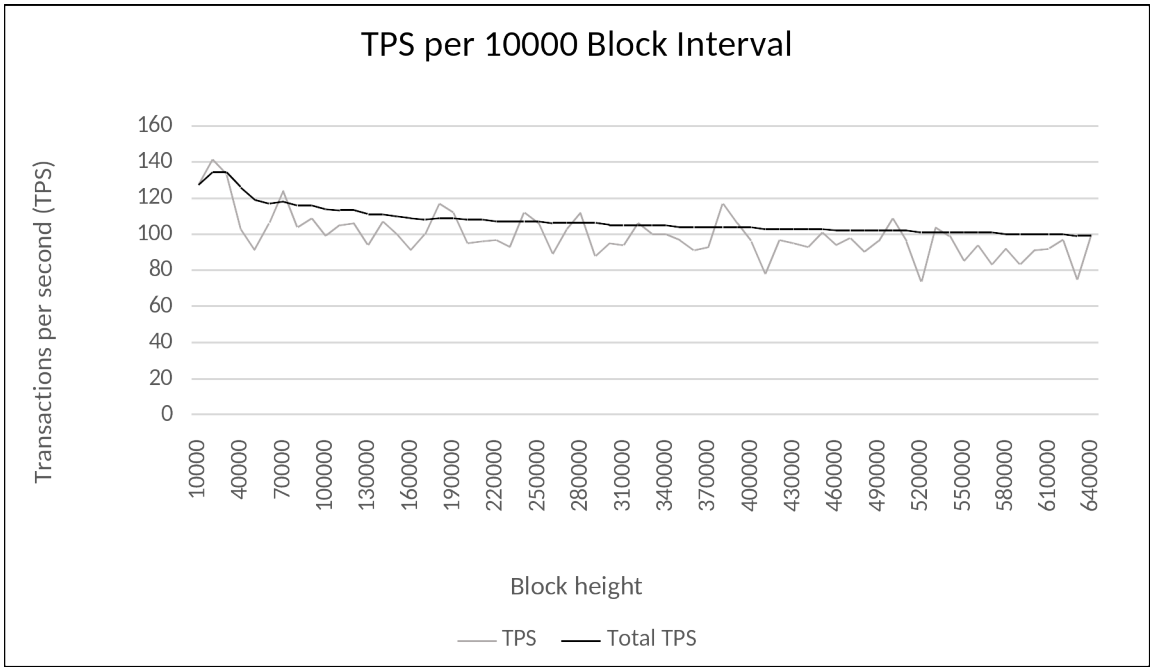
```
Average block time:                           1,509,958 / 641,270 =      2.35 s
Average total transaction throughput:         150,760,898 / 1,509,958 =  99.84 TPS
Average total transaction count per block:    150,760,898 / 641,270 =    235
Average parent chain transactions per block[1]: 35,296,971 / 641,270 =   55
Average "child chain block" tx per block:     1,950,914 / 641,270 =      3
Average child chain transactions per block:   113,513,013 / 641,270 =    177
```

[1]excluding "child chain block" transactions

The variation of transaction throughput with time as the blockchain size increases can be seen from the following table and graph, which presents the total transaction counts for each subsequent range of 10,000 blocks, the TPS (transactions per second) for that 10,000 blocks interval, and the cumulative TPS for the run up to that height:

HEIGHT RANGE	TRANSACTION COUNT	TIME START	TIME END	TIME INTERVAL	TPS	TOTAL TPS
10000	2999662	9158037	9181513	23476	127	127
20000	3359640	9181519	9205210	23691	141	134
30000	3131136	9205212	9228709	23497	133	134
40000	2440533	9228713	9252202	23489	103	126
50000	2158691	9252203	9275727	23524	91	119
60000	2516395	9275728	9299269	23541	106	117
70000	2927594	9299271	9322765	23494	124	118
80000	2474331	9322769	9346417	23648	104	116
90000	2574155	9346419	9369915	23496	109	116
100000	2352736	9369917	9393464	23547	99	114
110000	2487948	9393465	9417091	23626	105	113
120000	2511750	9417094	9440604	23510	106	113
130000	2237704	9440608	9464209	23601	94	111
140000	2516855	9464211	9487651	23440	107	111
150000	2368842	9487652	9511169	23517	100	110
160000	2146324	9511177	9534633	23456	91	109
170000	2362713	9534645	9558182	23537	100	108
180000	2751315	9558186	9581697	23511	117	109
190000	2640420	9581700	9605103	23403	112	109
200000	2239181	9605274	9628745	23471	95	108
210000	2290690	9628746	9652364	23618	96	108
220000	2300192	9652369	9675975	23606	97	107
230000	2194209	9675978	9699360	23382	93	107
240000	2650320	9699370	9722848	23478	112	107
250000	2510739	9722858	9746398	23540	106	107
260000	2111959	9746402	9770038	23636	89	106
270000	2460946	9770040	9793773	23733	103	106
280000	2655856	9793778	9817370	23592	112	106
290000	2076764	9817372	9840775	23403	88	106
300000	2249015	9840776	9864419	23643	95	105
310000	2215661	9864421	9887931	23510	94	105
320000	2516644	9887933	9911525	23592	106	105
330000	2377183	9911528	9935279	23751	100	105
340000	2384781	9935290	9958902	23612	100	105
350000	2294352	9958905	9982471	23566	97	104
360000	2151639	9982474	10005969	23495	91	104
370000	2193786	10005972	10029460	23488	93	104
380000	2673651	10029464	10052294	22830	117	104
390000	2462367	10053207	10076414	23207	106	104
400000	2255827	10076421	10099881	23460	96	104
410000	1845552	10099896	10123411	23515	78	103
420000	2304304	10123414	10146963	23549	97	103
430000	2263251	10146973	10170663	23690	95	103
440000	2213691	10170666	10194283	23617	93	103
450000	2386660	10194290	10217875	23585	101	103
460000	2237182	10217879	10241437	23558	94	102
470000	2329259	10241440	10264969	23529	98	102
480000	2129668	10264971	10288385	23414	90	102
490000	2270822	10288386	10311956	23570	96	102
500000	2555766	10311958	10335390	23432	109	102
510000	2257722	10335797	10359227	23430	96	102
520000	1745221	10359229	10382797	23568	74	101
530000	2450031	10382801	10406256	23455	104	101
540000	2340534	10406259	10429749	23490	99	101
550000	2014829	10429758	10453333	23575	85	101
560000	2214078	10453337	10476749	23412	94	101
570000	1961592	10476751	10500118	23367	83	101
580000	1997552	10500119	10521828	21709	92	100
590000	1971612	10523644	10547226	23582	83	100
600000	2164251	10547231	10570821	23590	91	100
610000	2189046	10570822	10594375	23553	92	100
620000	2300909	10594376	10617968	23592	97	100
630000	1708795	10617969	10640683	22714	75	99
640000	2326123	10641865	10665039	23174	100	99

It can be observed that after some initial decline of the TPS from 140+ to around 100 TPS over the first 50,000 blocks (about a day and a half), the TPS stabilizes around 100 TPS and doesn't decline much further for the rest of the run.



We also decided to look at the maximum per-block transaction counts encountered during the run.

Top 20 highest counts of parent transactions per block:

```
sql> select count(*) c, height from transaction_fxt where type=-2 group by height
order by c desc limit 20;
```

Tx count	Height
9998	75641
8795	470494
6331	175558
6059	107818
5839	62569
5047	284554
4553	498434
4428	291574
3787	597586
3679	624551
3384	307580
3272	581496
3215	38276
3086	597531
3074	206504
3029	568367
3029	624569
2924	211434
2871	79540
2791	195614

Top 10 highest counts of child chain transactions per child chain block for the Ignis child chain:

```
sql> select count(*) c, height from ignis.transaction group by height order by c
desc limit 10;
```

Tx count	Height
2456	29284
2273	277331
2212	173686
2165	558270
2160	44489
2130	368116
2128	584542
2081	495969
2068	430378
2067	485338

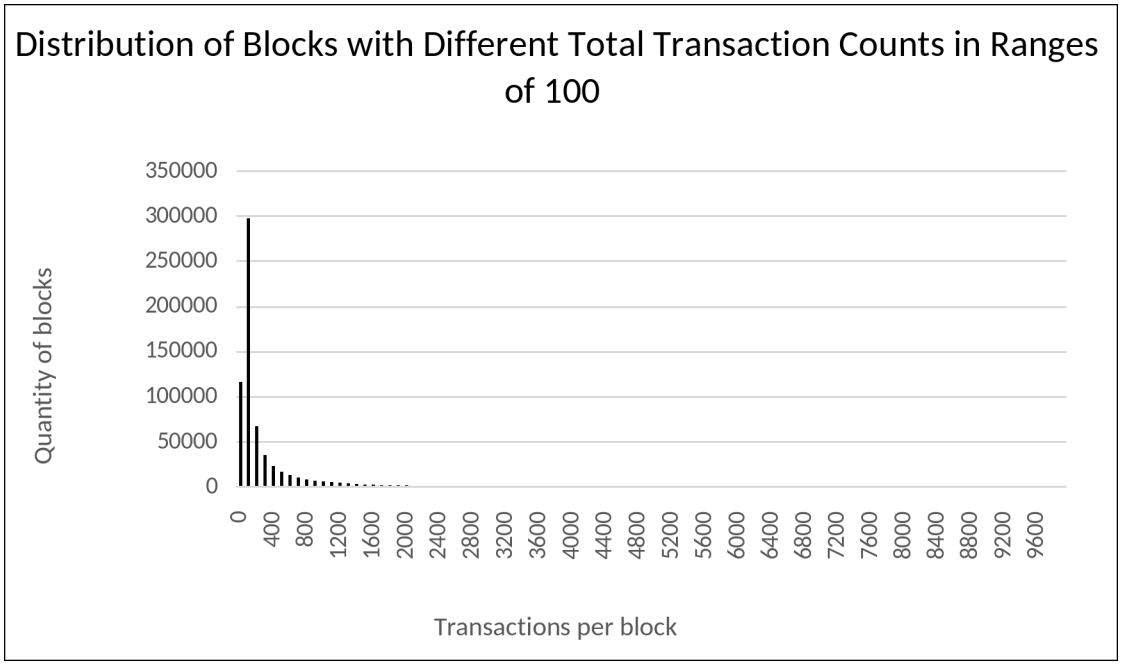
Top 20 highest counts of child chain transactions per block, aggregate of all child chains:

```
sql> select count(*) c, height from (select id, height from ignis.transaction union
all select id, height from aeur.transaction union all select id, height from
bitswift.transaction union all select id, height from mpg.transaction) group by
height order by c desc limit 20;
```

Tx count	Height
9827	29284
7662	1819
7205	30495
7112	277331
7073	173686
6922	558270
6911	44489
6851	4635
6844	29161
6829	8645
6816	368116
6815	1002
6812	584542
6805	4916
6795	6897
6653	28086
6615	485338
6600	430378
6532	24301
6526	20595

Lastly, the distribution of blocks with different total transaction counts, in ranges of 100, has been summarized in the following table and graph:

COUNT RANGE	NUMBER OF BLOCKS	COUNT RANGE	NUMBER OF BLOCKS
0	116554	5000	70
100	298037	5100	52
200	67472	5200	65
300	34820	5300	48
400	23007	5400	39
500	16871	5500	39
600	12974	5600	32
700	10354	5700	26
800	8400	5800	37
900	6887	5900	21
1000	5772	6000	18
1100	4952	6100	22
1200	4414	6200	22
1300	3754	6300	20
1400	3151	6400	21
1500	2634	6500	21
1600	2392	6600	18
1700	2035	6700	14
1800	1868	6800	13
1900	1544	6900	8
2000	1431	7000	9
2100	1274	7100	6
2200	1160	7200	8
2300	1014	7300	7
2400	823	7400	6
2500	739	7500	6
2600	694	7600	5
2700	583	7700	5
2800	519	7800	4
2900	458	7900	4
3000	470	8000	6
3100	427	8100	5
3200	351	8200	2
3300	338	8300	3
3400	287	8400	2
3500	227	8500	1
3600	231	8600	6
3700	207	8700	2
3800	185	8800	3
3900	165	8900	0
4000	173	9000	1
4100	150	9100	1
4200	155	9200	1
4300	118	9300	1
4400	124	9400	1
4500	79	9500	1
4600	81	9600	1
4700	83	9700	0
4800	82	9800	0
4900	51	9900	0



It can be concluded from the above data that transaction counts of above 2000 transactions per child block or parent block, or above 6000 total child chain transactions per block, are still sometimes reached in individual blocks. The hardcoded limits of 10000 transactions per child block or 10000 parent transactions in parent block were never reached, therefore the system throughput has not been artificially constrained by hardcoded restrictions.

It should also be noted that with the current transaction sizes of 185 bytes for a child chain ordinary payment transaction, and 149 bytes for a parent chain ordinary payment transaction, the payload of a child chain block with 2000 transactions is 360 kbytes, and a block with 6000 child chain transactions and 2000 parent chain transactions is 1.3 MB. This is still way below the hardcoded limit of 10,000 kbytes (9.8 MB) for child chain block with which the loadtest was run. For the average observed block transaction count of 235, the payload size would be 42 kbytes.

With almost half of the blocks, however, having less than 200 transactions, and about 1/6 of the blocks being empty, it is understandable that the average transaction count per block is only 235. This again suggests that such excessively short block times of 2 s are not really helping the overall transaction throughput and for a production system 5 to 10 s blocks are perhaps most reasonable to set.

7. Conclusion

In a test environment, a transaction throughput exceeding 100 TPS is sustainably achievable with only minor tuning of the current production Ardor blockchain software. Consensus is maintained and the blockchain can perform under such constant load for more than two weeks, processing over 150 M transactions. The results of this testing, together with a field test under the expected production operational parameters of the blockchain, can be used to arrive at an estimate of reasonable limits to use for transaction counts per block, block sizes, and recommended block times.