EOSIO Benchmark

Sungmin Ma <mrmsm@neowiz.com>
Lead System Engineer, **EOSeoul**

April 23, 2018

Software Configuration

- Ubuntu 16.04
- EOSIO git tag : dawn-v3.0.0
- patch : DOD patch
  - https://github.com/EOSIO/eos/pull/2297/files/437eb13a8a3f5724595f06ec20bc89ef9f967fdc

Client Environments

- Hardware
  - Client-PC : i7-6700 / 16G Mem / 1TB HDD
    - Number of machine : 1
  - Client-EC2 : AWS EC2 T2.medium + 150GB SSD(450 IOPS)
    - Number of machine : 4
- Linux parameters
  - ulimit file descriptor to 65535
  - sysctl tcp_tw_reuse=1
- Wallet daemon
  - 4 keosd instances per machine
- EOSIO accounts

    - Randomly-named 400 accounts per each machine
    - Keys of all accounts are imported to all keosd instances of all machines
- General scenario of client transfer action
    - One action per one transaction
        - We assume that transactions originated from general users are 1 action on 1 transaction.
    - Generate (source account, target account) pairs, total 10,000 pairs per machine (100 source account and 100 target accounts)
    - 10,000 transfer description
        - Each machine runs test script 40 times
        - Each test script runs EOS token transfer 250 times
        - Then 10,000 transfer action per machine
    - Generate total 50,000 multithreaded transfer action using 1 Client-PC and 4 Client-EC2

Block Producing Node Environment #1

Benchmark description

- Scenario description
    - Single System
    - Single or multiple nodeos processes
- Linux parameters
    - ulimit file descriptor to 65535
    - sysctl tcp_tw_reuse=1
- AWS Instance type hardware specification
    - M5 CPU : Intel Xeon Platinum 8175M CPU @ 2.5Ghz
    - M5.xlarge : 4 cores, 8GB RAM
    - M5.2xlarge : 8 cores, 16GB RAM
    - M5.4xlarge : 16 cores, 32GB Ram

Observations

- CPU error messages
    - “Block exhausted allowed resource (block has insufficient cpu resource)”
    - “Ensure that the reference block exist in the blockchain”
    - “Transaction’s reference block did not match, Is this transaction from a different fork?”

AWS BP Instance type	# of machines	# of nodeos	client action request	Total TPS	system usage
M5.xlarge	1	1	threads : 40 req per thread : 250	~200	

M5.xlarge	1	1	threads : 40 req per thread : 250	315	CPU 40%
M5.xlarge	1	1	threads : 40 req per thread : 250	315	CPU 25%
M5.xlarge (client using m5.large)	1	2	threads : 40 req per thread : 250	413	CPU 45%
M5.xlarge (client using m5.large)	1	4	threads : 40 req per thread : 250	366	CPU 100% 5~6 MBps
M5.2xlarge (client using m5.large)	1	4	threads : 40 req per thread : 250	472 till CPU errors 360 with CPU errors	
M5.2xlarge (client using m5.large)	1	4	threads : 40 req per thread : 250	466 with lower CPU errors	
M5.2xlarge (client using m5.large)	1	4	threads : 50 req per thread : 250	350 with CPU errors	
M5.4xlarge (client using m5.large)	1	4	threads : 50 req per thread : 250	~412 with CPU error	
M5.4xlarge (client using m5.large)	1	4	threads : 10 req per thread : 1000	393 without CPU error no failed trxs	
M5.4xlarge (client using m5.large)	1	4	threads : 20 req per thread : 500	~300 with CPU error 18,000 failed trxs	13 MBps CPU 100%

Block Producing Node Environment #2

Benchmark description

- Scenario description
    - Multiple systems
    - Multiple nodeos process
- Linux parameters (same with BP Env #1)
    - ulimit file descriptor to 65535
    - sysctl tcp_tw_reuse=1
- AWS Instance type hardware specification
    - M5 CPU : Intel Xeon Platinum 8175M CPU @ 2.5Ghz
    - M5.xlarge : 4 cores, 8GB RAM
    - C5 CPU : Intel Xeon Platinum *))) @ 3.0Ghz
    - C5.2xlarge : 8 cores, 16GB RAM

Observations

AWS BP Instance type	# of machines	# of nodeos	client action request	Total TPS	system usage
M5.xlarge	2	3	threads : 20 req per thread : 500	312 with CPU errors	
C5.2xlarge	2	3	threads : 40 req per thread : 250	413~442 without CPU errors	
C5.2xlarge (client using C5.xlarge)	2	3	threads : 40 req per thread : 250	472 till CPU errors	
C5.2xlarge (client using C5.xlarge)	2	3	threads : 40 req per thread : 250 req : Hello World contract	peak 1042 avg 485 till CPU errors	

Conclusions

To get maximum TPS performance using single-threaded nodeos of dawn-v3.0.0:

- Higher CPU clock, higher TPS.
- When CPU hits 100%, CPU resource error appears.
- When CPU resource error occurred, block forked.
- Disk performance is not that crucial when using 1 nodeos on 1 machine.
- Max sustainable(not peak) TPS is under 500 with multiple nodeos daemons on single or multiple machines.

To get maximum from keosd:
- Also higher CPU clock, higher request performance.
- Transaction signing requires CPU power.

Appendix - Test scripts

- scripts : http://testnet01.eoseoul.io/script/bmt_client.tar.gz
- usage
    - ./bmt.sh prepare
    - ./bmt.sh run_job