PERFORMANCE BENCHMARK OF MYSQL NDB CLUSTER WITH SCALEARC CACHING USING SYSBENCH

Author: Aarti Karve
Prepared By: Great Software Laboratory
In collaboration with: ScaleArc
Date: January 29 2017

Contents

1 Introduction

The objective of this white paper is to evaluate the quantum of performance improvement possible by using **"Transparent In-Memory Query Caching"** feature offered by ScaleArc and identify if ScaleArc adds any overhead, being a database proxy.

The ScaleArc database load balancing software runs transparently between databases and application servers. It offers features including automatic read/write split for app-transparent scalability, app-transparent failover for zero downtime, caching and connection pooling for performance improvement, and real-time analytics for actionable insights.

ScaleArc improves application performance leveraging three key technologies:

- **Transparent Query Caching:** Enables users to cache frequently run Read-Only query responses. ScaleArc enables cache invalidation via Time-To-Live or automatic invalidation. Users can also pre-populate the cache.
- **Connection Pooling and Multiplexing:** ScaleArc establishes only those connections with the database servers that are required for executing queries. Idle connections from applications to database servers are multiplexed and pooled.
- **Dynamic Load Balancing:** ScaleArc automatically load balances the SQL traffic across multiple read servers, leveraging Time to First Byte and replication lag monitoring to select the optimal server. Users can use regex rules to customize load balancing as well.

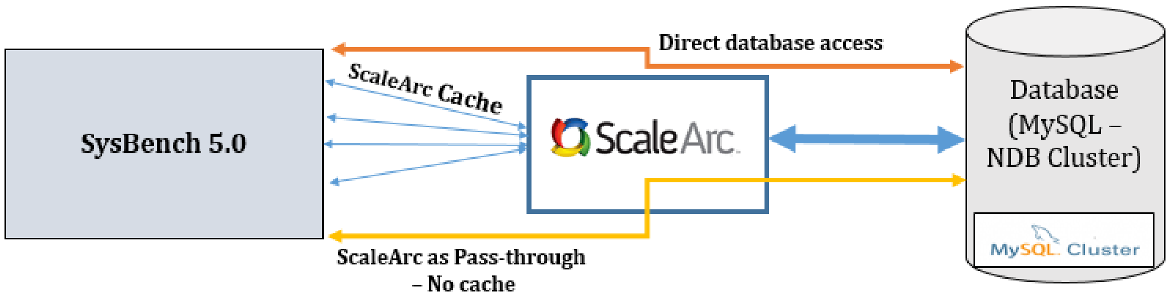
The performance engineering team at GS Lab, in collaboration with the ScaleArc team, conducted a study to evaluate the performance benefits of ScaleArc's transparent query caching on MySQL Cluster (NDB storage Engine). ScaleArc enables users to cache frequently run Read-Only query responses, without any changes to the application code or the database.

This document describes the details of this exercise, the test environment, process, experiments, tools, and the test results.

2 Test Environment

The test environment includes:

- ScaleArc for MySQL: database load balancing software
- MySQL NDB Cluster: The database under test
- SysBench: Load Generator



- **SysBench** – The SysBench benchmark tool supports performance testing in the areas of:
  - CPU
  - OLTP for database
  - File I/O for Disk I/O test
  - Threads – Threads subsystem performance test

  OLTP for database (Online Transaction Processing (OLTP) test profile) - Tests were carried out using SysBench 5.0 db test module for OLTP transactions. SysBench OLTP application benchmark runs on top of a MySQL database with multiple threads.

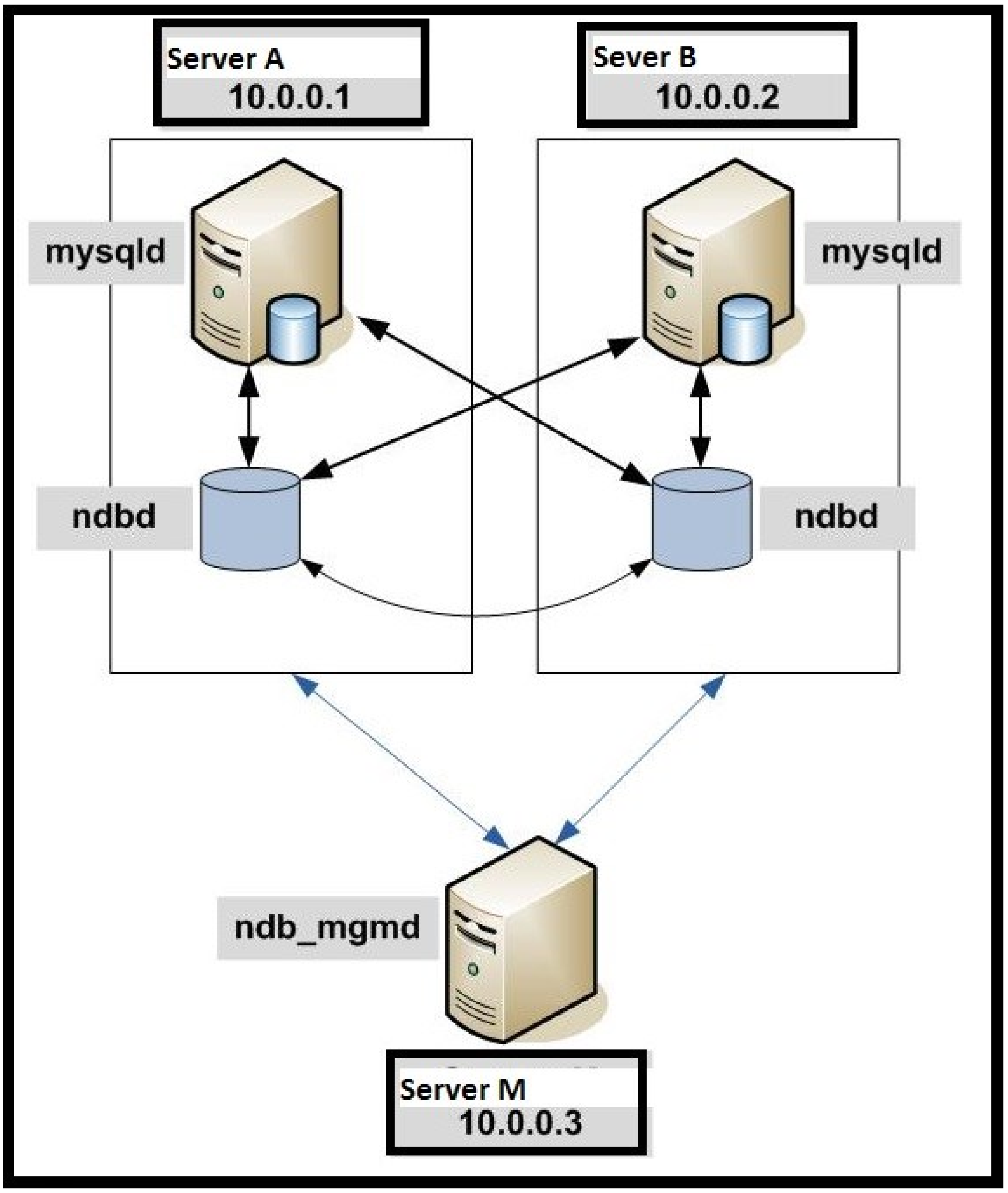
  SysBench helps create a test database, gathering test statistics at granular level on a MySQL database with a few commands. To generate the OLTP traffic, we use the OLTP.lua script.

- **ScaleArc for MySQL 3.9.0** – The ScaleArc software was hosted on a VM with 12 CPUs - (2 sockets with 6 cores per socket) with 4 Licensed CPUs for ScaleArc.
  More VM configuration details are in appendix ScaleArc software appliances.

- **MySQL NDB Cluster** MySQL Cluster is implemented using a distributed, shared-nothing architecture. MySQL cluster has an in-memory clustered storage engine called NDB (**N**etwork **D**ata**B**ase).

The cluster set up at GS lab has 5 nodes as shown in the following image:

Architecture of MySQL (NDB) Cluster



The testing environment included one management node for managing the cluster (process: ndb_mgmd), two MySQL server nodes for access to NDB data (process: mysqld), and two data nodes for storage of the data (process: ndbd)].

The server configuration details can be found in section Configuration of Test Environment Setup in the Appendix: 6.2

3 Test Scenarios

We considered only Read transactions over a mix of read/ write, to find out the maximum improvement from caching.

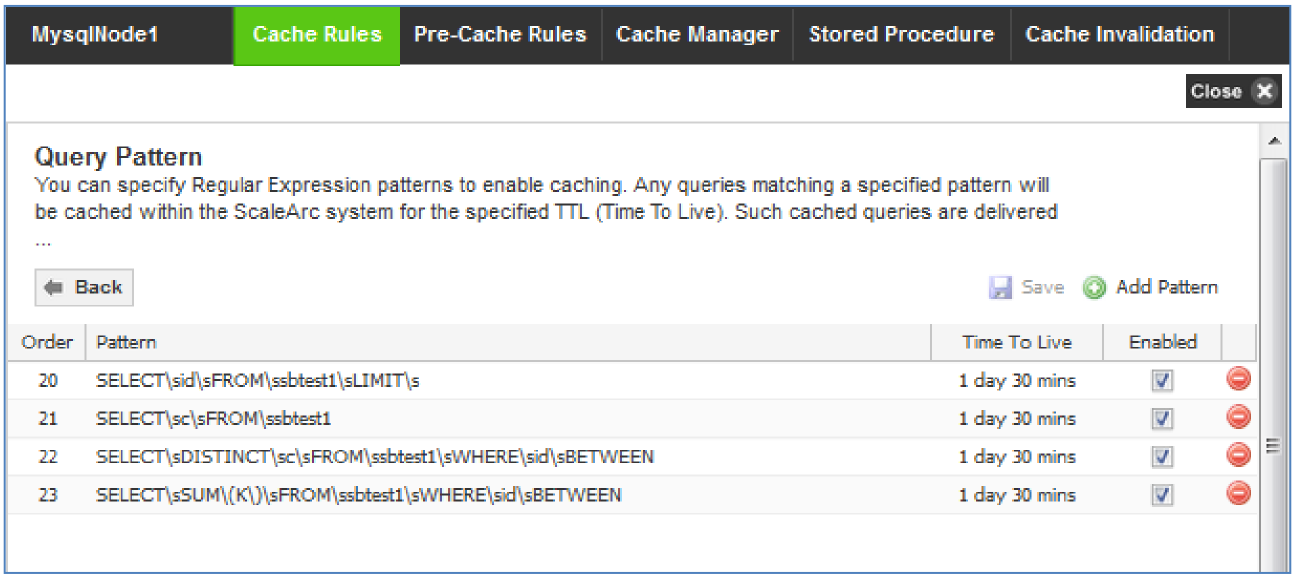
Performance testing measured the results of the following test scenarios:

1. Direct database access: This scenario measures performance of MySQL NDB cluster directly, since the application (SysBench) directly communicates with database without making a hop through the ScaleArc software.

2. ScaleArc with no caching: Connection to the database through ScaleArc, where ScaleArc acts only as a pass-through proxy. The purpose of this setup is to measure the overhead of ScaleArc. For a HA (High Availability) configuration, the ScaleArc proxy in pass-through mode would balance load between multiple servers.

3. ScaleArc with caching: Connection to database through ScaleArc, with caching enabled in ScaleArc for read queries. In this scenario, the frequently seen queries are added to the ScaleArc in-memory cache by updating an appropriate regular expression. ScaleArc allows its users to identify good candidates for query caching in its analytics views.

We cached the following patterns in our tests:



Query Pattern

You can specify Regular Expression patterns to enable caching. Any queries matching a specified pattern will be cached within the ScaleArc system for the specified TTL (Time To Live). Such cached queries are delivered ...

Order	Pattern	Time To Live	Enabled
20	SELECT\sid\sFROM\ssbtest1\sLIMIT\s	1 day 30 mins	✓
21	SELECT\sc\sFROM\ssbtest1	1 day 30 mins	✓
22	SELECT\sDISTINCT\sc\sFROM\ssbtest1\sWHERE\sid\sBETWEEN	1 day 30 mins	✓
23	SELECT\sSUM\(K\)\sFROM\ssbtest1\sWHERE\sid\sBETWEEN	1 day 30 mins	✓

3.1 Workload Details

- **Thread Ramp-up pattern**: 1 4 8 16 32 64 128 256 512 1024 2048

  (After every 600 seconds, the thread count would double in number)
- **Max Time** (for which each thread is running)**:** 600 seconds
- **Sleep Time** between thread ramp up - 1 Second
- **OLTP table Size** = 10,000 rows of data

3.2 SysBench Commands

1. **Data generation using SysBench Prepare flag:**

```
sysbench --test=/usr/share/doc/sysbench/tests/db/oltp.lua --db-
driver=mysql --oltp-table-size=10000 --mysql-host='10.XX.XX.53' --
mysql-db=sysbench --mysql-user=root --mysql-password=admin --mysql-
table-engine=NDBCLUSTER --oltp-tables-count=64 prepare
```

2. **SysBench Run command to send the OLTP traffic to the database :**

Direct, no caching	`for each in 1 4 8 16 32 64 128 256 512 1024 2048; do sysbench --test=/usr/share/doc/sysbench/tests/db/oltp_no_trx.lua --db-driver=mysql --oltp-table-size=10000 --mysql-host='10.XX.XX.59' --mysql-db=sysbench --mysql-user=root --mysql-password=admin --mysql-table-engine=NDBCLUSTER --max-time=600 --max-requests=0 --oltp-read-only=on --oltp-skip-trx--oltp-nontrx-mode=select --num-threads=$each run --report-interval=1 ; sleep 1; done`
ScaleArc, no caching **and** **ScaleArc, with caching**	`for each in 1 4 8 16 32 64 128 256 512 1024 2048; do sysbench --test=/usr/share/doc/sysbench/tests/db/oltp_no_trx.lua --db-driver=mysql --oltp-table-size=10000 --mysql-host='10.XX.XX.53' --mysql-db=sysbench --mysql-user=root --mysql-password=admin --mysql-table-engine=NDBCLUSTER –max-time=600 --max-requests=0 --oltp-read-only=on --oltp-skip-trx--oltp-nontrx-mode=select --num-threads=$each run --report-interval=1 ; sleep 1; done`

3.3 Scope and Limitations

- Measuring Read-Only queries
- Queries outside a transaction can be added to cache, hence changes were made to the oltp.lua script as follows:

  Remove (autocommit is on and no START TRANSACTION is issued)

  ```
  if not oltp_skip_trx then
  db_query(begin_query)
   end
  ```
- We used SysBench OLTP workload with only read-only queries instead of a particular application so that the results are widely reproducible. The extent of performance improvement in a real-world application would depend on the percentage of database traffic that can be cached.

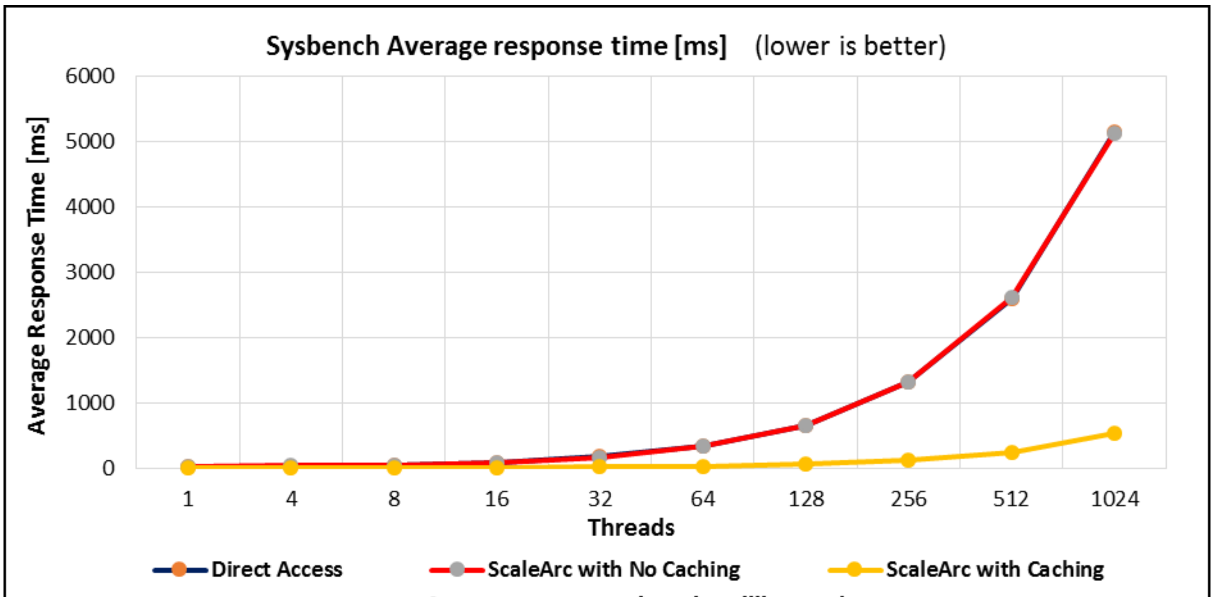
4 Effects of Caching on Read-Only Workload

4.1 SysBench Average Response Time

The following graph compares the SysBench average response time, in milliseconds, for the direct access, ScaleArc without caching, and ScaleArc with caching scenarios.

The results show a faster response (lower is better) when the request is served from ScaleArc cache (visible in Yellow). Our tests showed an 88% drop in response time when 100% caching was achieved (i.e., all the queries were served from the ScaleArc cache).

The response time shows little difference between the direct access (in blue) and the ScaleArc pass-through (in red) scenarios. It is therefore safe to say that ScaleArc introduces little to no overhead itself.



Average response time, in milliseconds

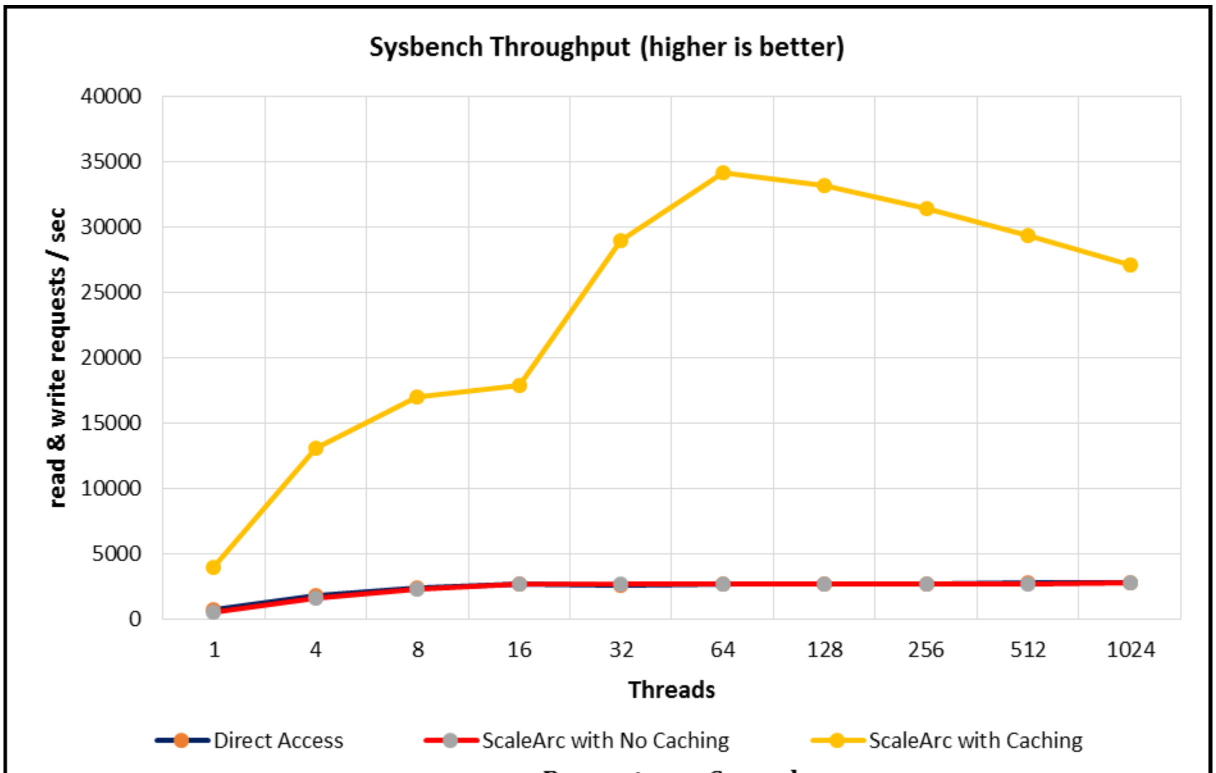
The yellow line highlights the improvement in response time in the caching scenario. The red and blue lines highlight that the ScaleArc software itself introduces no measurable latency. For the source data, refer to Result Data in Appendix section: Result Data **6.1.3**

4.2 SysBench Throughput

The following graph compares the SysBench throughput measured in requests per second (RPS) for the direct access, ScaleArc with no caching, and ScaleArc with caching scenarios.

The results show tremendous improvement in throughput when the request is served from ScaleArc cache (the yellow line). Performance improves approximately 9x when 100% caching is achieved.

Throughput for direct database access (blue line) is almost the same as the throughput for ScaleArc with no caching (red line), since the blue and red lines overlap so greatly. This result shows that ScaleArc introduces little to no latency, or overhead, when used as a pass-through proxy, in spite of introducing a hop between the application and the database.



Requests per Second

The yellow line highlights the improvement in throughput in the caching scenario. The red and blue lines highlight that the ScaleArc software itself introduces no measurable latency. For raw data, refer to Result Data in Appendix section 6.1.1

4.3 Tuning Details

4.3.1 Bandwidth can help improve the MySQL Cluster performance

Prior to running the previous benchmarks, we observed that during initial test runs, the network bandwidth was capping out at around 100 Mbps. This cap was visible in the “Bandwidth Usage” graph on the “Live Monitor” tab of the ScaleArc analytics views. The reason for the limit was that the ScaleArc VM was hosted on a different XEN server than that of the MySQL Cluster.

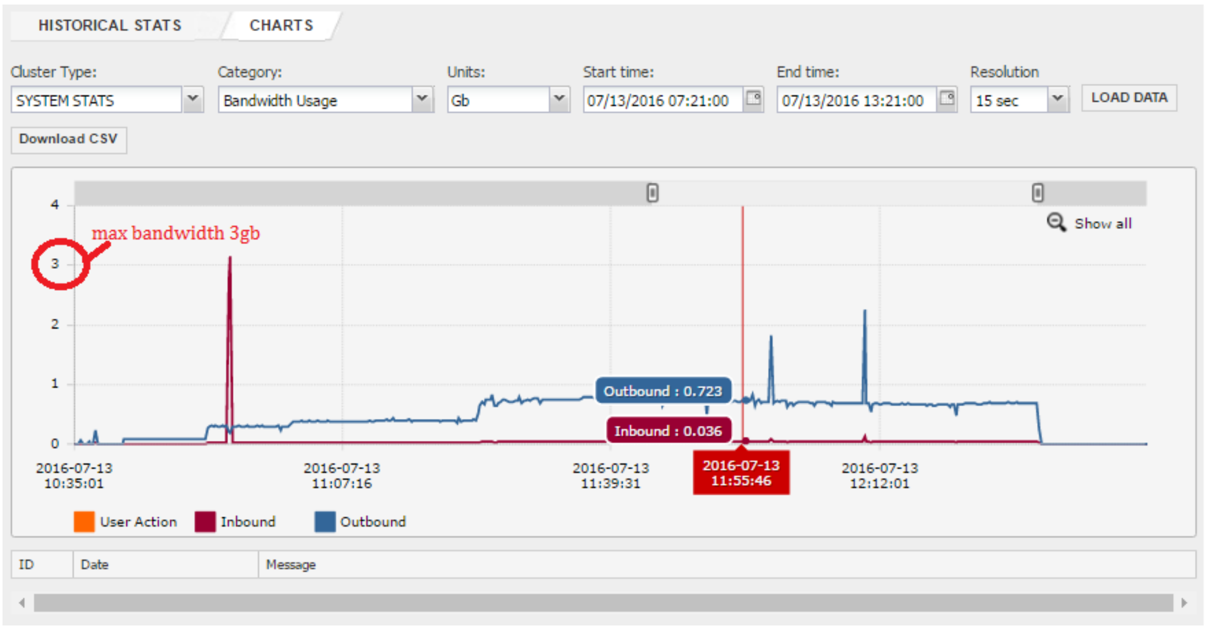
Putting the ScaleArc software in the same XEN server drastically increased performance:

Iperf Bandwidth between ScaleArc (server) and SysBench (Client)

[ID] Interval Transfer Bandwidth

[4] 0.0-10.0 sec 11.0 GBytes 9.47 gbps

The Historical Stats option in the ScaleArc GUI shows the details for bandwidth usage during the tests as shown in the following screen captures:



Cluster Type: SYSTEM STATS
Category: Bandwidth Usage
Units: Gb
Start time: 07/13/2016 07:21:00
End time: 07/13/2016 13:21:00
Resolution: 15 sec

ScaleArc’s Historical Stats screen showing Bandwidth Usage for the Test duration

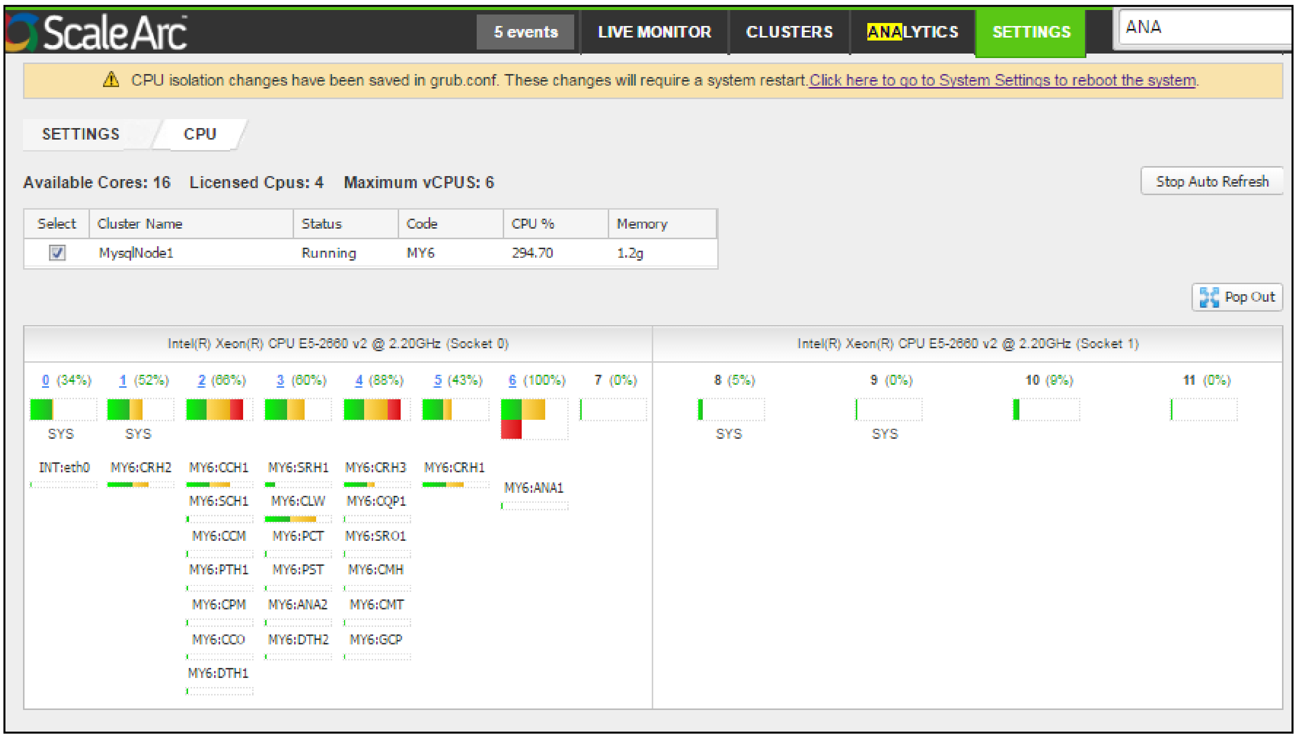
4.3.2 Using the CPU Distribution Option to Tune ScaleArc

ScaleArc lets users configure the allocation of CPUs to different threads/ activities used for various ScaleArc or system operations.

ScaleArc also provides an option to move or clone “Busy threads” to available spare CPU to ensure optimal CPU distribution. Please note that only those threads that contribute to traffic can be cloned.

ScaleArc was deployed as a virtual machine with 4-core ScaleArc license, and the VM had 12 CPUs assigned by Xen server for the experiment described in this section. For optimum CPU distribution, moving the “busy threads” to available cores enabled us to map underlying processes to different cores, which helped optimize CPU utilization of ScaleArc VM.

The following screen capture shows the ScaleArc CPU distribution for 2 CPUs (6 cores per CPU) and 4-core ScaleArc license (with thread allocation tuned from CPU distribution option in ScaleArc UI):



⚠ CPU isolation changes have been saved in grub.conf. These changes will require a system restart. Click here to go to System Settings to reboot the system.

Available Cores: 16   Licensed Cpus: 4   Maximum vCPUS: 6

Select	Cluster Name	Status	Code	CPU %	Memory
☑	MysqlNode1	Running	MY6	294.70	1.2g

The analytics view in the ScaleArc GUI uses a red bar to represent 100% usage of threads. These “busy threads” can be cloned or moved based on availability of the CPU.

All the cores are busy as shown in the image and ScaleArc performance could improve from having more CPUs. Since 4 cores appear insufficient from these results, the next section describes experiments where we applied 8-core ScaleArc license.

4.3.3 ScaleArc license CPU Provisioning

Performance Comparison when 8-core ScaleArc license is hosted on a VM with 12 CPUs vs. 4 CPUs:

The ScaleArc software runs as a physical or virtual appliance and is licensed per core of ScaleArc used. To test provisioning of the ScaleArc CPU licenses, we executed a test with different CPU core combinations assigned to support the 8-core ScaleArc licenses.

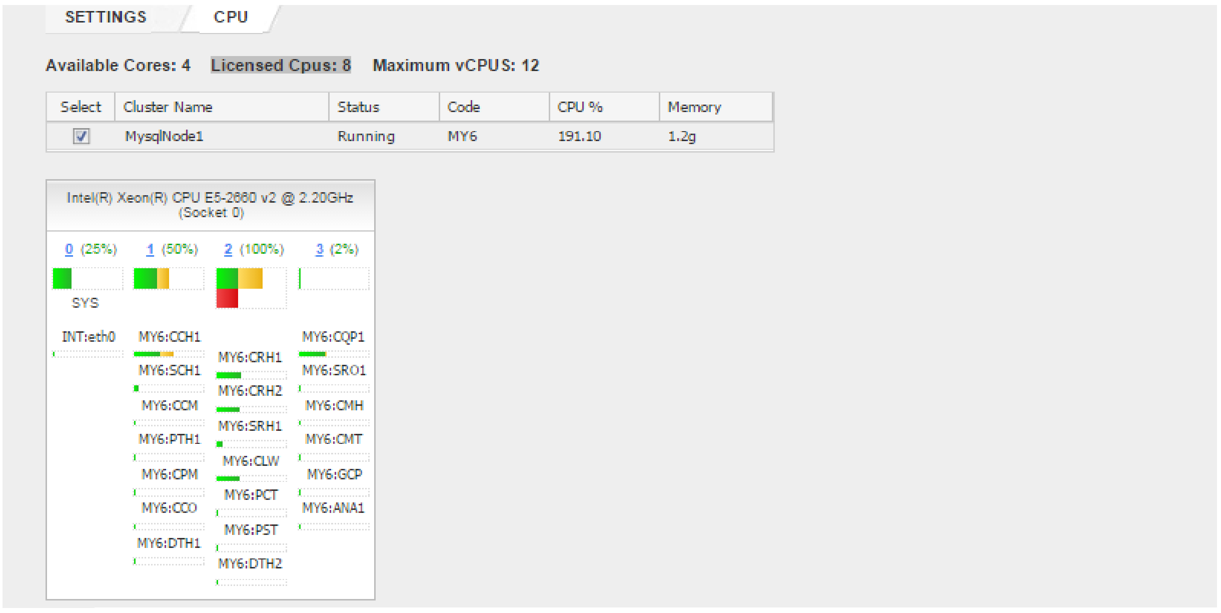
ScaleArc sizes its deployments based on the active database cores. Users should run approximately one core of ScaleArc for every eight active database cores (plus one passive HA ScaleArc instance per active ScaleArc instance). ScaleArc sells a minimum of four cores.

The following images compare CPU Distribution for the two scenarios when the ScaleArc software was hosted on a VM with 4 CPUs vs. a VM with 12 CPUs.

The two VM configurations used are:

- VM with **04 CPUs -** (1 socket with 4 cores per socket), 8 ScaleArc cores
- VM with **12 CPUs -** (2 sockets with 6 cores per socket), 8 ScaleArc cores

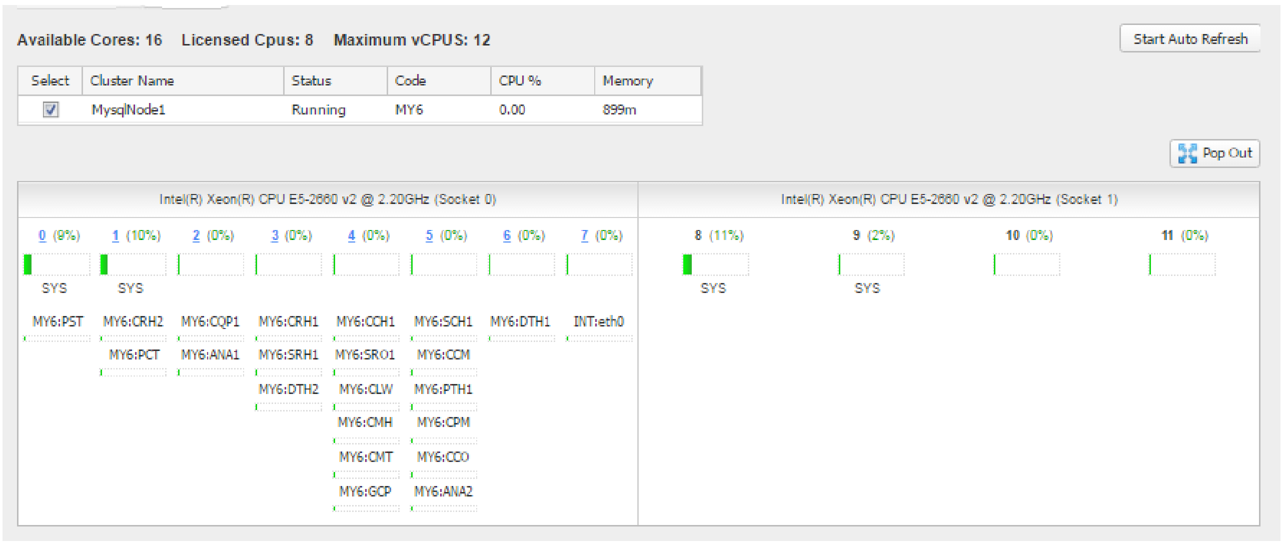
For a given load, as seen in a sample in image 1 that represents the 4-CPU configuration, all the CPUs are busy, with some cores used 100% and no headroom for extra tasks.
For the 12 CPUs configuration, as shown in a sample in image 2, the CPU is underutilized as 6 cores are idle. We saw that 2 or more cores were consistently idle during the experiment. Also, there was no further performance improvement in throughput and response time for the aforementioned scenarios by using 12 CPUs instead of 4 CPUs. It seems that the client has exhausted all its resources and cannot push additional load. Therefore, 6 CPUs should be sufficient, at this traffic load, for the ScaleArc VM to achieve close to peak throughput at a lower cost (although we did not run another set of experiments on a VM assigned 6 CPUs/ cores by Xen server).



Available Cores: 4 Licensed Cpus: 8 Maximum vCPUS: 12

Select	Cluster Name	Status	Code	CPU %	Memory
✓	MysqlNode1	Running	MY6	191.10	1.2g

Image 1: 04 CPUs (1 socket with 4 cores per socket), 8 ScaleArc licensed CPUs


Available Cores: 16 Licensed Cpus: 8 Maximum vCPUS: 12

Select	Cluster Name	Status	Code	CPU %	Memory
☑	MysqlNode1	Running	MY6	0.00	899m

Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz (Socket 0)

0 (9%)	1 (10%)	2 (0%)	3 (0%)	4 (0%)	5 (0%)	6 (0%)	7 (0%)
SYS	SYS						
MY6:PST	MY6:CRH2	MY6:CQP1	MY6:CRH1	MY6:CCH1	MY6:SCH1	MY6:DTH1	INT:eth0
	MY6:PCT	MY6:ANA1	MY6:SRH1	MY6:SRO1	MY6:CCM		
			MY6:DTH2	MY6:CLW	MY6:PTH1		
				MY6:CMH	MY6:CPM		
				MY6:CMT	MY6:CCO		
				MY6:GCP	MY6:ANA2		

Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz (Socket 1)

8 (11%)	9 (2%)	10 (0%)	11 (0%)
SYS	SYS		

Image 2: 12 CPUs (2 sockets with 6 cores per socket) , 8 ScaleArc licensed CPUs
For raw performance data, refer to sections **6.1.5** [Throughput read and write requests/ second] for 12 CPUs vs. 4 CPUs] and **6.1.6** [SysBench Response time for 12 CPUs vs. 4 CPUs of ScaleArc System]

5 Conclusion

Using a reproducible benchmark, we verified that caching with ScaleArc results in tremendous improvement in SQL query performance. Although SysBench may not look representative of real-world applications, where all queries may not be cached, our results show the extent to which performance can be improved with caching.

Also, the ScaleArc software introduces negligible overhead compared to direct database access. Therefore, IT staff can leverage its features, aside from caching, to benefit uptime and performance of applications without introducing delay.

The key takeaway of this benchmarking exercise is that caching with ScaleArc resulted in a nearly 9x increase in throughput and about an 88% drop in query response time. Users running relational databases supported by ScaleArc (MySQL, SQL Server, and Oracle), looking for improvement in database performance, can look at the nature of their workloads and expect improvement proportional to the percentage of read queries that can be cached. They can run this benchmark to know the quantum of improvement they can expect in their environment before conducting full-scale performance testing of their product with and without ScaleArc.

6 Appendix

6.1 Result Data

6.1.1 SysBench Throughput (read requests per second)

Threads	Read/write requests per second		
	Direct Access	ScaleArc with No Caching	ScaleArc with Caching
1	707.16	573.2	3947.64
4	1779.47	1599.73	13103.95
8	2402.99	2291.84	17047.34
16	2667.55	2678.14	17865.9
32	2611.24	2716.6	28978.86
64	2695.13	2723.08	34144.9
128	2731.24	2723.35	33201.07
256	2734.51	2727.97	31456.17
512	2757.15	2732.93	29413.21
1024	2757.15	2788.37	27146.91

6.1.2 SysBench Total Reads (read requests per second)

Threads	(Total Reads/600 seconds)		
	Direct Access	ScaleArc with No Caching	ScaleArc with Caching
1	424298	343924	2368590
4	1067724	959882	7862400
8	1441888	1375206	10228484
16	1600704	1607074	10719688
32	1567174	1630370	17394090
64	1618260	1634640	20494656
128	1640310	1635564	19933606
256	1643712	1639834	18888128
512	1660694	1645868	17694292
1024	1660694	1684634	16332050

6.1.3 SysBench Average Response Time (in milliseconds)

Threads	Avg. Response Time, in milliseconds		
	Direct Access	ScaleArc with No Caching	ScaleArc with Caching
1	19.8	24.42	3.54
4	31.47	35	4.27
8	46.61	48.86	6.57
16	83.97	83.63	12.54
32	171.55	164.89	15.45
64	332.22	328.97	26.23
128	655.83	657.63	53.94
256	1309.57	1312.58	113.86
512	2595.84	2618.63	243.1
1024	5151.74	5125.82	526.85

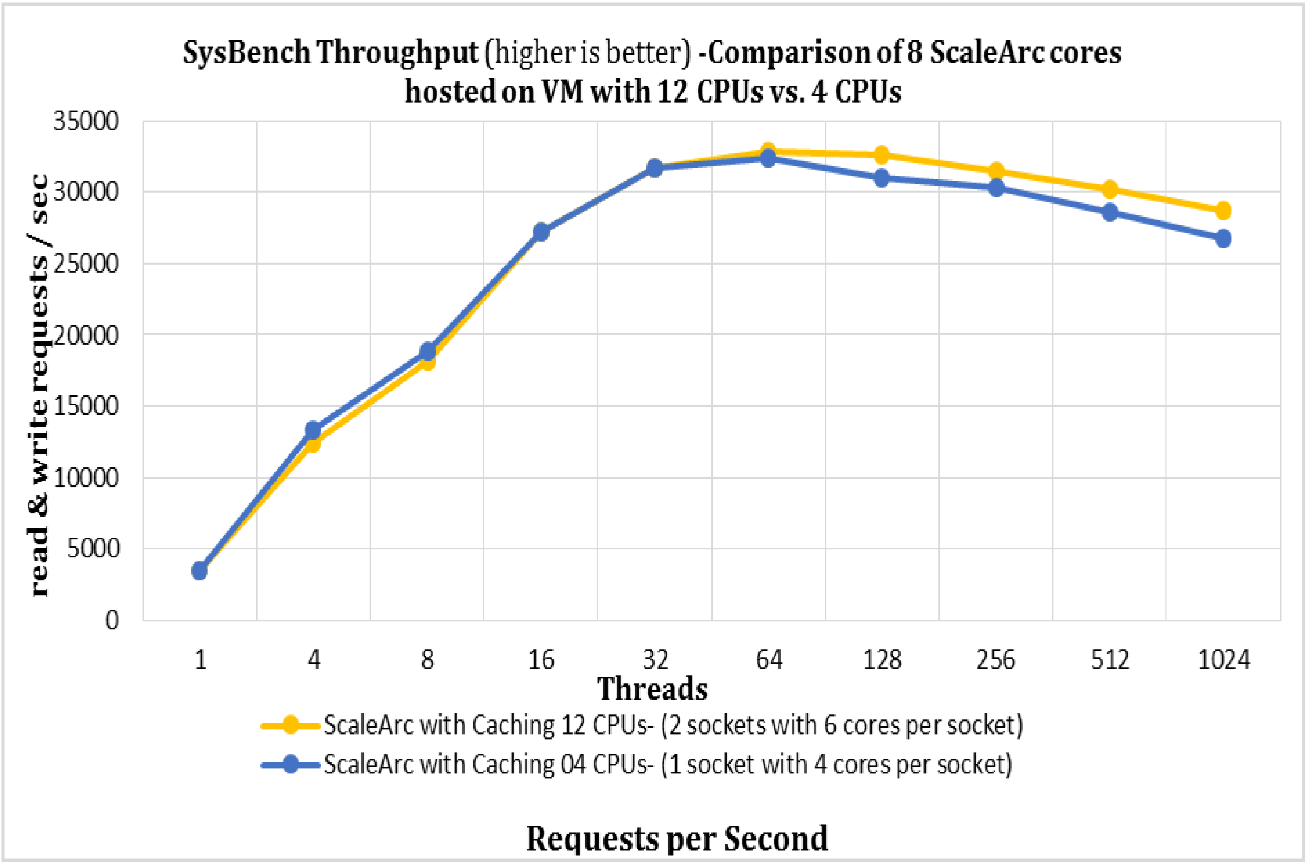
6.1.4 SysBench 95 Percentile Response Time (in milliseconds)

Threads	95% Response Time, in milliseconds		
	Direct Access	ScaleArc with No Caching	ScaleArc with Caching
1	22.07	28.55	3.99
4	35.66	39.99	5.08
8	50.83	56.15	8.54
16	89.17	89.76	17.08
32	185.72	174.72	30.66
64	352.42	343.16	212.18
128	678.44	680.47	256.6
256	1396.6	1408.35	462.63
512	2839.91	2868.1	821.69
1024	5506.4	6983.75	1366.82

6.1.5 Throughput in read/write requests per second for 12 CPUs vs. 4 CPUs

(Performance Comparison of 8 ScaleArc cores hosted on VM with 12 CPUs vs. 4 CPUs)

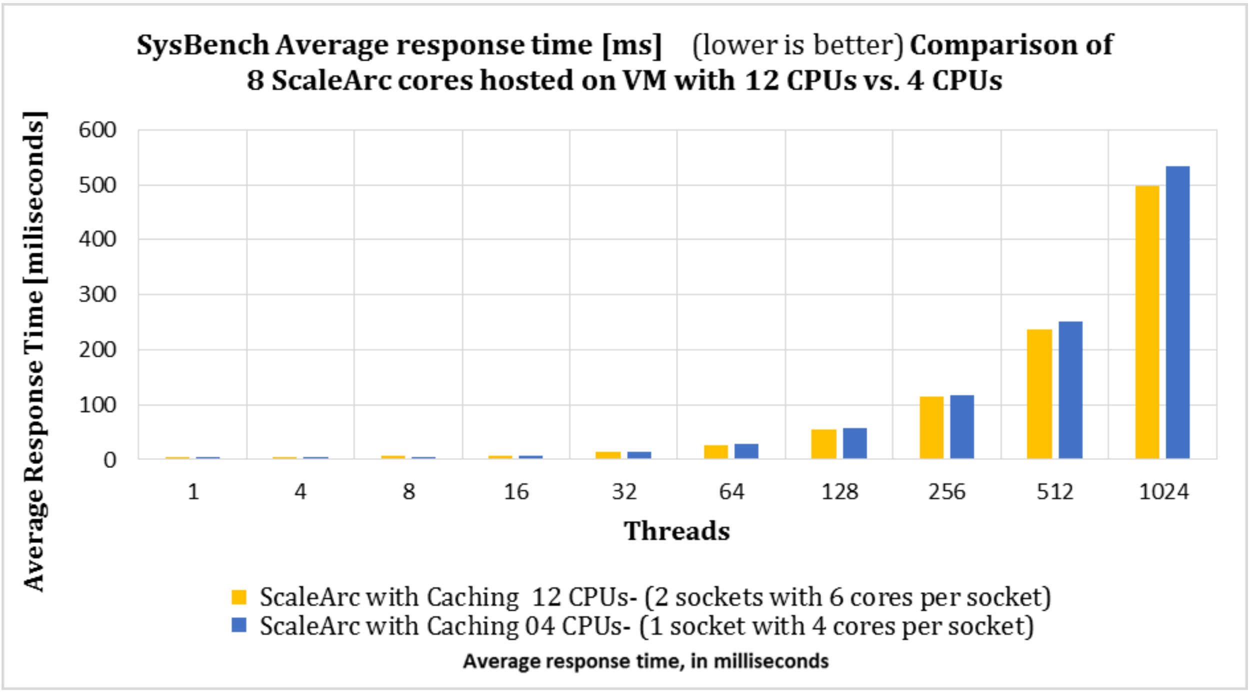
Threads	Read/write requests per second (Licensed CPU: 8)			
	Direct Access	ScaleArc with No Caching	ScaleArc with Caching 12 CPUs (2 sockets with 6 cores per socket)	ScaleArc with Caching 04 CPUs (1 socket with 4 cores per socket)
1	707.16	573.2	3464.76	3530.48
4	1779.47	1599.73	12416.78	13346.06
8	2402.99	2291.84	18131.13	18852.01
16	2667.55	2678.14	27164.74	27245.37
32	2611.24	2716.6	31650.23	31666.38
64	2695.13	2723.08	32826.42	32375.18
128	2731.24	2723.35	32586.51	31026.07
256	2734.51	2727.97	31416.61	30369.04
512	2757.15	2732.93	30237.49	28630.62
1024	2757.15	2788.37	28716.86	26812.89



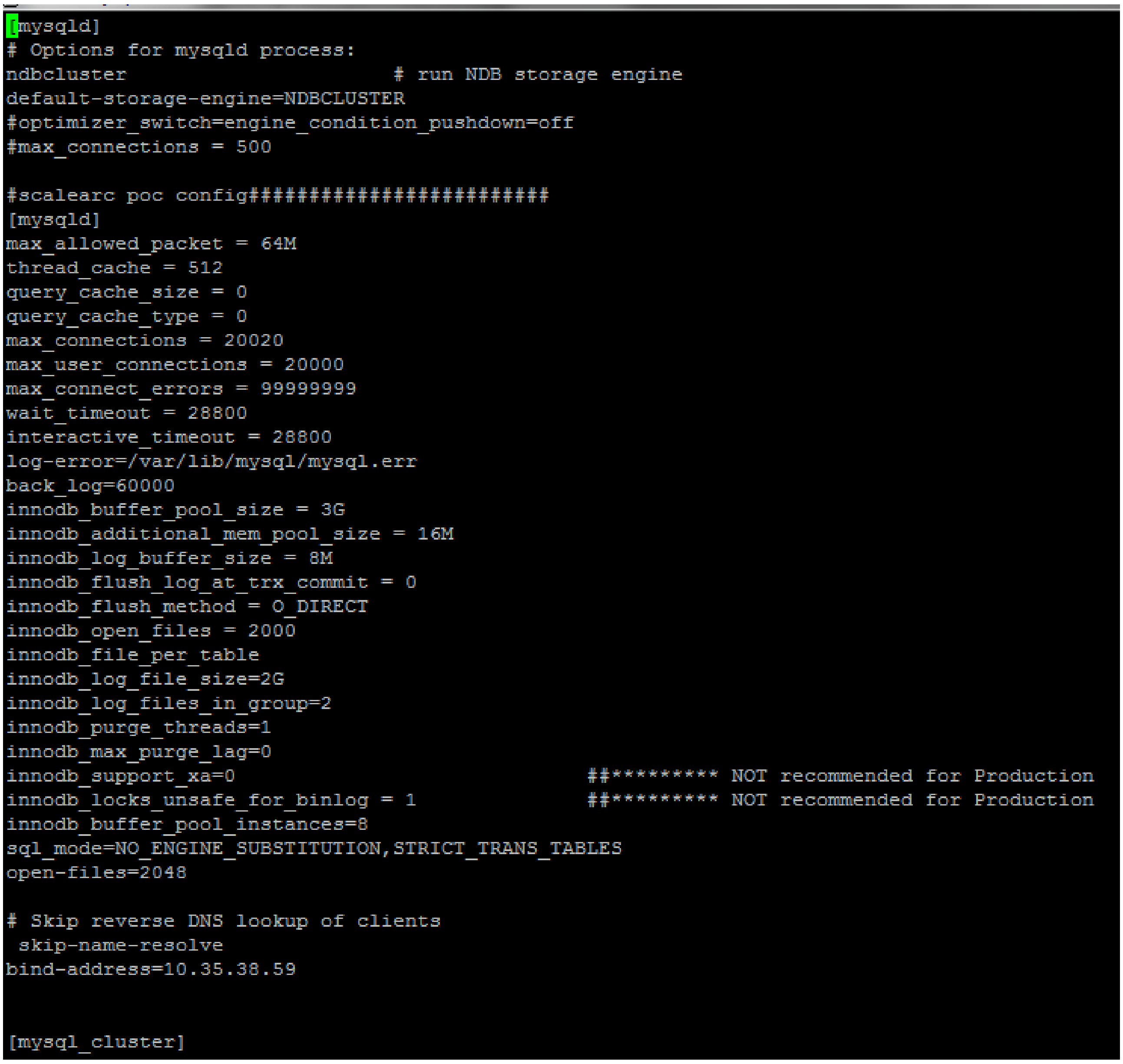
6.1.6 SysBench Response Time for 12 CPUs vs. 4 CPUs of ScaleArc Software

(Performance Comparison of 8 ScaleArc cores hosted on VM with 12 CPUs vs. 4 CPUs)

Threads	(Avg. Response Time, in milliseconds)			
	Direct Access	**ScaleArc with No Caching**	**ScaleArc with Caching 12 CPUs** (2 sockets with 6 cores per socket)	**ScaleArc with Caching 04 CPUs** (1 socket with 4 cores per socket)
1	19.8	24.42	4.04	3.96
4	31.47	35	4.51	4.19
8	46.61	48.86	6.17	5.94
16	83.97	83.63	8.24	8.22
32	171.55	164.89	14.15	14.14
64	332.22	328.97	27.27	27.67
128	655.83	657.63	54.97	57.72
256	1309.57	1312.58	114.01	117.94
512	2595.84	2618.63	236.77	250.2
1024	5151.74	5125.82	498.11	533.42



6.1.7 MySQL Configuration



```
[mysqld]
# Options for mysqld process:
ndbcluster                      # run NDB storage engine
default-storage-engine=NDBCLUSTER
#optimizer_switch=engine_condition_pushdown=off
#max_connections = 500

#scalearc poc config########################
[mysqld]
max_allowed_packet = 64M
thread_cache = 512
query_cache_size = 0
query_cache_type = 0
max_connections = 20020
max_user_connections = 20000
max_connect_errors = 99999999
wait_timeout = 28800
interactive_timeout = 28800
log-error=/var/lib/mysql/mysql.err
back_log=60000
innodb_buffer_pool_size = 3G
innodb_additional_mem_pool_size = 16M
innodb_log_buffer_size = 8M
innodb_flush_log_at_trx_commit = 0
innodb_flush_method = O_DIRECT
innodb_open_files = 2000
innodb_file_per_table
innodb_log_file_size=2G
innodb_log_files_in_group=2
innodb_purge_threads=1
innodb_max_purge_lag=0
innodb_support_xa=0                                ##********** NOT recommended for Production
innodb_locks_unsafe_for_binlog = 1                 ##********** NOT recommended for Production
innodb_buffer_pool_instances=8
sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES
open-files=2048

# Skip reverse DNS lookup of clients
 skip-name-resolve
bind-address=10.35.38.59


[mysql_cluster]
```

6.2 Configuration of Test Environment Setup

MySQL Cluster Nodes:

MySQL Cluster - Data Node 1 & 2	**MySQL Cluster - SQL Node 1 & 2**	**MySQL Cluster - Management Node**
Architecture : x86_64 CPU op-mode(s): 32-bit, 64-bit Byte Order: Little Endian CPU(s): 4 On-line CPU(s) list: 0-3 Thread(s) per core: 1 Core(s) per socket: 1 CPU socket(s): 4 NUMA node(s): 1 Vendor ID: GenuineIntel CPU family: 6 Model: 62 Stepping: 4 CPU MHz: 2194.702 BogoMIPS: 4390.07 Hypervisor vendor: Microsoft Virtualization type: full L1d cache: 32K L1i cache: 32K L2 cache: 256K L3 cache: 25600K NUMA node0 CPU(s): 0-3	Architecture: x86_64 CPU op-mode(s): 32-bit, 64-bit Byte Order: Little Endian CPU(s): 4 On-line CPU(s) list: 0-3 Thread(s) per core: 1 Core(s) per socket: 1 CPU socket(s): 4 NUMA node(s): 1 Vendor ID: GenuineIntel CPU family: 6 Model: 62 Stepping: 4 CPU MHz: 2194.967 BogoMIPS: 4389.48 Hypervisor vendor: Microsoft Virtualization type: full L1d cache: 32K L1i cache: 32K L2 cache: 256K L3 cache: 25600K NUMA node0 CPU(s): 0-3	Architecture: x86_64 CPU op-mode(s): 32-bit, 64-bit Byte Order: Little Endian CPU(s): 4 On-line CPU(s) list: 0-3 Thread(s) per core: 1 Core(s) per socket: 1 CPU socket(s): 4 NUMA node(s): 1 Vendor ID: GenuineIntel CPU family: 6 Model: 62 Stepping: 4 CPU MHz: 2194.839 BogoMIPS: 4389.01 Hypervisor vendor: Microsoft Virtualization type: full L1d cache: 32K L1i cache: 32K L2 cache: 256K L3 cache: 25600K NUMA node0 CPU(s): 0-3
Physical disk: 20GB	**Physical disk: 20GB**	**Physical disk: 20GB**
RAM :10GB	**RAM :8GB**	**RAM :4GB**

ScaleArc software appliance **In the initial test we used 4 CPUs** (1 socket with 4 cores per socket) **In the subsequent tests we increased the CPUs to 12** (2 sockets with 6 cores per socket)	**Client Machine**
Architecture: x86_64 CPU op-mode(s): 32-bit, 64-bit Byte Order: Little Endian CPU(s): 4 On-line CPU(s) list: 0-3 Thread(s) per core: 1 Core(s) per socket: 1 Socket(s): 4 NUMA node(s): 1 Vendor ID: GenuineIntel CPU family: 6 Model: 62 Stepping: 4 CPU MHz: 2194.778 BogoMIPS: 675.25 Hypervisor vendor: Xen Virtualization type: full L1d cache: 32K L1i cache: 32K L2 cache: 256K L3 cache: 25600K NUMA node0 CPU(s): 0-3	Architecture: x86_64 CPU op-mode(s): 32-bit, 64-bit Byte Order: Little Endian **CPU(s): 8** On-line CPU(s) list: 0-7 Thread(s) per core: 1 Core(s) per socket: 1 Socket(s): 8 NUMA node(s): 1 Vendor ID: GenuineIntel CPU family: 6 Model: 62 Stepping: 4 CPU MHz: 2194.697 BogoMIPS: 4385.43 Hypervisor vendor: Microsoft Virtualization type: full L1d cache: 32K L1i cache: 32K L2 cache: 256K L3 cache: 25600K NUMA node0 CPU(s): 0-7
Physical disk: 40GB	Physical disk: 20GB
RAM :4GB	RAM :8GB