# SocketPro server queue performance study and its comparison with Kafka

**Contents:**

## 1. Introduction

Performance is one of most important software quality metrics. Performance considerably determines application responsiveness, scalability and throughput. Therefore, UDAParts has been paying high attention to the performance of SocketPro communication framework continuously since the beginning of its development. Both SocketPro client and server core libraries have persistent queue implemented with excellent performance. This short article is focused on performance study about SocketPro server queue only. In order to attract your attention more, this concise article compares SocketPro server queue with popular persistent queue Apache Kafka in performance.

In fact, UDAParts compared SocketPro with Microsoft WCF (window communication framework) and RabbitMQ previously as described here. This is the second research paper about SocketPro server queue performance.

The performance study samples are located within the directory ../SocketProRoot/samples/qperf. The obtained performance data could be found at the file ../SocketProRoot/samples/socketpro_kafka.xlsx.

## 2. Experiment conditions

Tests are completed with a home desktop as server or broker for persistent message queue comparison and a laptop as a client for cross-machine communication with network bandwidth equal to 1 Gbps. Major hardware parts are listed as the below.

**Server queue machine**: Ubuntu 16.04 LTS, Intel Core i7-4770 CPU @ 3.40 GHz, 16 GB RAM.
**Message provider/consumer**: Windows 10 professional, Intel Core i7-5500U CPU @ 2.40 GHz, 8 GB RAM.
**Switch**: 1 Gb ethernet.
**Queue disk:** Seagate desktop HDD ST2000DM001-2TB, 64MB Cache , 7200-RPM spin speed.
**Software components**: ../socketpro/samples/qperf/jperf/src/sqclient/Program.java;SocketPro version 6.0.3.4; Kafka version 2.11-0.10.1.1.

Kafka performance test commands can be found at the file ../socketpro/samples/qperf/kafka_perf_test.txt. All Kafka performance data are always measured with one thread and acknowledgement set to 1 (acks=1) without any replication. All other configurations are defaulted, unless the file kafka_perf_test.txt labels explicitly.

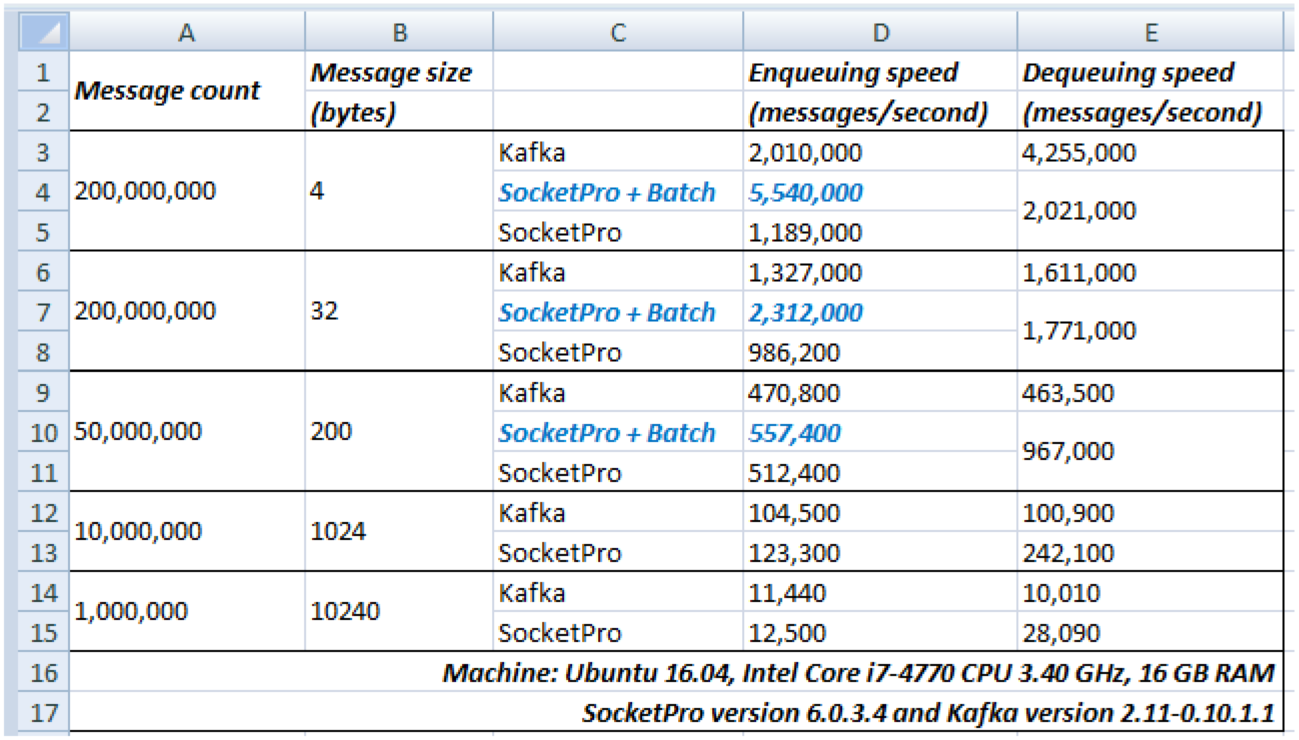
All these tests focus on throughputs of both producer and consumer. No consumer runs during testing provider throughput. Similarly, no provider runs when measuring consumer throughput. Since throughputs are very dependent on testing message sizes, the performance comparison study employs five different sizes of messages, 4, 32, 200, 1024 and 10240 bytes. The first two sizes (4 and 32 bytes) represent small messages. The third and fourth ones (200 and 1024 bytes) are scheduled for middle sizes of messages. The last one is designed for testing large messages. It is easy to obtain good throughput for large messages, but much more difficult to get good throughput for those small messages as processing each of small messages dominates and requires much more CPU costs.

At last, SocketPro server queue supports batch message en-queuing in addition to one- by-one, which could significantly improve message writing speed for those small message. In regards to Kafka, messages are saved with batch size set to 8196 except the large messages (10240)

## 3. Experiment results and analysis

### Localhost – all provider, consumer and server queue running on one Linux machine

First, the experiment is completed for cross-application queue communication within the same machine without considering networking as listed on the below Figure 1.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | ***Message count*** | ***Message size*** | | ***Enqueuing speed*** | ***Dequeuing speed*** |
| 2 | | ***(bytes)*** | | ***(messages/second)*** | ***(messages/second)*** |
| 3 | | | Kafka | 2,010,000 | 4,255,000 |
| 4 | 200,000,000 | 4 | *SocketPro + Batch* | *5,540,000* | 2,021,000 |
| 5 | | | SocketPro | 1,189,000 | |
| 6 | | | Kafka | 1,327,000 | 1,611,000 |
| 7 | 200,000,000 | 32 | *SocketPro + Batch* | *2,312,000* | 1,771,000 |
| 8 | | | SocketPro | 986,200 | |
| 9 | | | Kafka | 470,800 | 463,500 |
| 10 | 50,000,000 | 200 | *SocketPro + Batch* | *557,400* | 967,000 |
| 11 | | | SocketPro | 512,400 | |
| 12 | 10,000,000 | 1024 | Kafka | 104,500 | 100,900 |
| 13 | | | SocketPro | 123,300 | 242,100 |
| 14 | 1,000,000 | 10240 | Kafka | 11,440 | 10,010 |
| 15 | | | SocketPro | 12,500 | 28,090 |
| 16 | ***Machine: Ubuntu 16.04, Intel Core i7-4770 CPU 3.40 GHz, 16 GB RAM*** | | | | |
| 17 | ***SocketPro version 6.0.3.4 and Kafka version 2.11-0.10.1.1*** | | | | |

*Figure 1: Performance comparison between SocketPro and Kafka queues for cross-application communication within the same machine*

**Small messages:** First of all, let's consider small messages having sizes of 4 and 32 bytes. Without using SocketPro server queue batch, Kafka delivers better throughputs for both messages writing and reading. Without SocketPro server queue batch, SocketPro sends and reads messages one by one with inline continuous data batching algorithm at both client and server side. Although the algorithm improves both message writing and reading speed, it does require a lot of CPU power for the overhead of processing each of small messages. In addition, SocketPro internally supports acknowledging message de-queuing one-by-one for success or failure, which requires more CPU cost, network bandwidth and disk queue file position seeking and writing activity. It seems that Kafka doesn't support individual message de-queuing acknowledge, which makes its message reading super fast because it doesn't send acknowledge message back to server queue. Further, Kafka does not do disk queue file position seeking and writing for de-queuing success or failure either. All these reasons summed leads to better message de-queuing throughput.

However, SocketPro could be 125% (4bytes + SocketPro + batch) faster than Kafka in message enqueuing (writing) when messages are batched manually as shown in the previous Figure 1. It is noted that SocketPro queue message batch size is also set to 8196 bytes, which is the same as Kafka's configuration setting batch.size.

**Middle messages**: Let's move onto middle messages (200 and 1024 bytes). SocketPro is always faster than Kafka for both messages writing and reading. This performance study shows that SocketPro is about 20% faster than Kafka in messages writing, and could be 140% faster in messages reading as shown in the previous Figure 1.

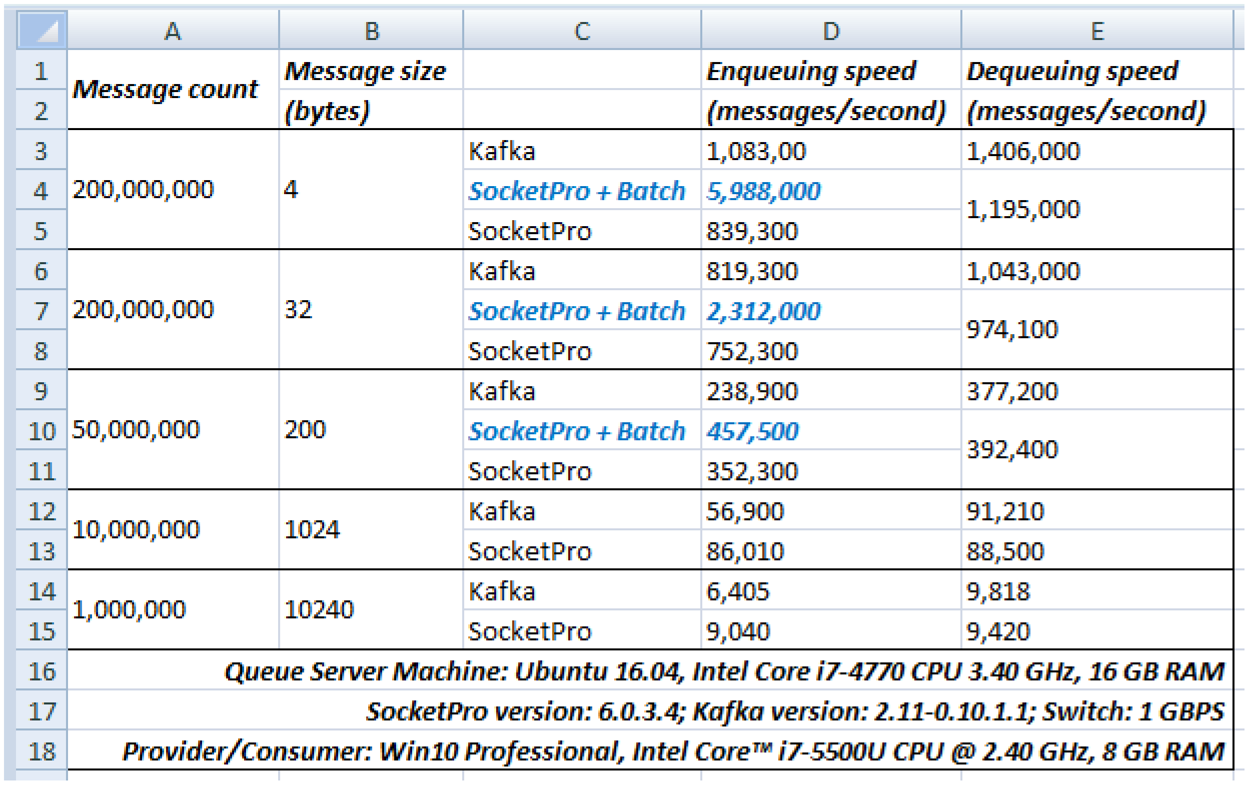
SocketPro queue is designed with more features for your easy development. It requires you do very few things for queue file managements. SocketPro queue supports individual message de-queuing acknowledge, which obviously leads to more CPU cost, disk queue file position seeking and writing activity. If messages are large in size, these overheads are largely reduced silently without your input all. Once you do performance calculation on these data, you will find the actual throughputs are majorly determined by hard disk writing and reading speeds. SocketPro queue writing speed could reach about 120 MB per second (123,300 * 1024=120 MB), which is very close to actual disk writing speed. Further, SocketPro queue reading speed could amazingly reach to 240 MB per second (242,100 * 1024=240 MB)!

It should be pointed out that SocketPro manual batch doesn't improve middle-size messages writing speed as overheads of sending middle messages are ignorable from data in the above Figure 1.

**Large messages**: At last, let's look at large messages data. Both SocketPro and Kafka share similar performance in messages writing or en-queuing. However, SocketPro could be 180% faster than Kafka in messages reading or de-queuing. As you can see, SocketPro is significantly more efficient than Kafka.

## Cross-machine -- provider/consumer and server queue running on two different machines

It is more important to analyze queue performance comparison data for cross-machine queue communication involved with eithernet network as shown in the below Figure 2. This scenario is much closer to real queue applications.



| Message count | Message size (bytes) | | Enqueuing speed (messages/second) | Dequeuing speed (messages/second) |
|---|---|---|---|---|
| 200,000,000 | 4 | Kafka | 1,083,00 | 1,406,000 |
| | | *SocketPro + Batch* | *5,988,000* | 1,195,000 |
| | | SocketPro | 839,300 | |
| 200,000,000 | 32 | Kafka | 819,300 | 1,043,000 |
| | | *SocketPro + Batch* | *2,312,000* | 974,100 |
| | | SocketPro | 752,300 | |
| 50,000,000 | 200 | Kafka | 238,900 | 377,200 |
| | | *SocketPro + Batch* | *457,500* | 392,400 |
| | | SocketPro | 352,300 | |
| 10,000,000 | 1024 | Kafka | 56,900 | 91,210 |
| | | SocketPro | 86,010 | 88,500 |
| 1,000,000 | 10240 | Kafka | 6,405 | 9,818 |
| | | SocketPro | 9,040 | 9,420 |

***Queue Server Machine: Ubuntu 16.04, Intel Core i7-4770 CPU 3.40 GHz, 16 GB RAM***

***SocketPro version: 6.0.3.4; Kafka version: 2.11-0.10.1.1; Switch: 1 GBPS***

***Provider/Consumer: Win10 Professional, Intel Core™ i7-5500U CPU @ 2.40 GHz, 8 GB RAM***

*Figure 2: Queue performance comparison between SocketPro and Kafka for cross-machine queue communication*

***Small messages:*** Similar to the previous localhost test cases as shown in the Figure 1, Kafka is still somewhat (~20%) faster than SocketPro for both small messages (4 + 32 bytes) writing and reading if SocketPro queue is not batched manually. However, SocketPro could deliver significantly higher (450%, 4 bytes + SocketPro + batch) throughput than Kafka in messages en-queuing if SocketPro messages batch is employed as shown in the above Figure 2.

**Middle messages**: The above Figure 2 clearly shows that SocketPro is about 25% faster than Kafka for middle size message writing. SocketPro is especially great in networking, which leads to better throughputs as SocketPro is specially engineered with continuous inline request/result batching, real-time stream processing, asynchronous data transferring, and parallel computation in mind.

Again, it should be pointed out that SocketPro manual batch doesn’t significantly improve middle size messages writing speed as shown in the above Figure 2. Also, overheads of sending individual middle or large messages are ignorable. Both localhost and cross-mchine test results show similar conclusions. Overall, both SocketPro and Kafka show similar message reading or dequeuing speed.

**Large messages**: Both SocketPro and Kafka have similar throughputs or performance for both messages writing and reading as they are determined by network bandwidth and disk accessing speed. However, SocketPro consistently shows about 20% faster than Kafka in messages writing as shown in the previous Figure 2.

## 4. Conclusions

SocketPro is written with inline continuous data batching, non-blocking and parallel computation in mind so that networking and CPU processing at both client and server sides are extremely efficient. This design leads to SocketPro queue excellent performance or throughput. As shown in previous Figures, Kafka would be faster than SocketPro server queue only when reading or dequeuing small messages having sizes of less than 64 bytes. For all other situations, SocketPro queue will be faster than or equal to Kafka in both reading and writing middle or large messages having sizes of larger than 200 bytes. Under extremely cases, SocketPro could be about 200% faster than Kafka in both writing and reading messages.

In addition, SocketPro manual batch could significantly improve writing or enqueuing small messages performance or throughput. As expected, SocketPro manual batch will not improve performance for middle or large messages as overheads of processing (writing and reading) each of messages can be ignorable.

At last, SocketPro and Kafka may share the same throughputs in reading/dequeuing middle and large messages when their throughputs are majorly determined by network bandwidth and hard disk accessing speed.