Information Management software

IBM® InfoSphere® Streams v3.0 Performance Report

Roch Archambault
Richard King
IBM Software Group

Contents

In IBM® InfoSphere® Streams v3.0, IBM significantly improved the performance of the Aggregate and Import/Export operators. Performance of several Streams applications have also been improved by 20% to 100%.

Introduction

This document presents performance measurements for IBM® InfoSphere® Streams Version 3.0. We will describe the tests carried out and the corresponding results.
In some cases, we will compare version 3.0 with version 2.0 (previous version).
In the remaining document, we will refer to these as v3.0 and v2.0.

Overall we find that for representative Application Benchmarks Streams 3.0 can be as much as 100% faster than Streams 2.0. On fine-grained analysis of transport performance, we have identified a reduction in throughput for small tuple sizes related to new functionality for monitoring transport health. For Streams Application Benchmarks, this reduction in transport throughput is generally observed to be offset by significant optimizations in the operators and compiler.

Application Benchmarks

In order to provide a broader view of InfoSphere Streams performance we created a series of application benchmarks. These originate from real world applications in the telecom and banking industries. For each benchmark we measured the throughput (in tuples/sec) of the final output stream. Those benchmarks were executed on two consecutive versions of InfoSphere Streams product (v2.0 and v3.0) on both RedHat Enterprise Linux (RHEL) 5.5 and RHEL 6.1.

Test Description

Here is a description of each application benchmark used to measure InfoSphere Streams performance.

BIGTELCO

- Large Telecom application which performs Call Detail Record (CDR) processing
- Composed of 90 Processing Elements (PE) and runs on 4 nodes
- Heavy use of strings in SPL and contains some C++ operators

SMALLTELCO

- Small Telecom application which performs CDR processing
- Composed of 12 PEs and runs on 2 nodes
- Heavy use of strings in SPL and contains some C++ operators

XMLPARSE

- Small banking application which performs XML processing
- Composed of 7 PEs and runs on 3 nodes
- Heavy use of strings in SPL and contains some C++ operators

LOGS

- Small Telecom application which performs log analysis
- This benchmark is called the log analysis benchmark on Streams Exchange. Streams Exchange can be accessed at the following IBM developerWorks URL: http://www.ibm.com/developerworks/wikis/display/streams/Home
- There are 3 variations of this application:
    - BAS: read and basic parsing of log record
    - FLT: filter to select events of interest
    - ENR: adds additional data to the source event such as mapping an identity to organization.
- Composed of 46 PEs and runs on 4 nodes

Test Results: Comparison to previous release

We compare the performance of BIGTELCO, SMALLTELCO and XMLPARSE. Figure 1 and Table 1 show across the board improvement in throughput performance of v3.0 versus v2.0. Throughput improvements range between 20% and 100%. We also observed improvements running on RHEL 6.1 versus RHEL 5.5 operating systems for the real-world telecommunications benchmarks running on v3.0. The XMLPARSE benchmark saw a small decline in performance for v3.0 on RHEL6.1 versus RHEL 5.5. This may be related to changes in string handling in RHEL6.1.

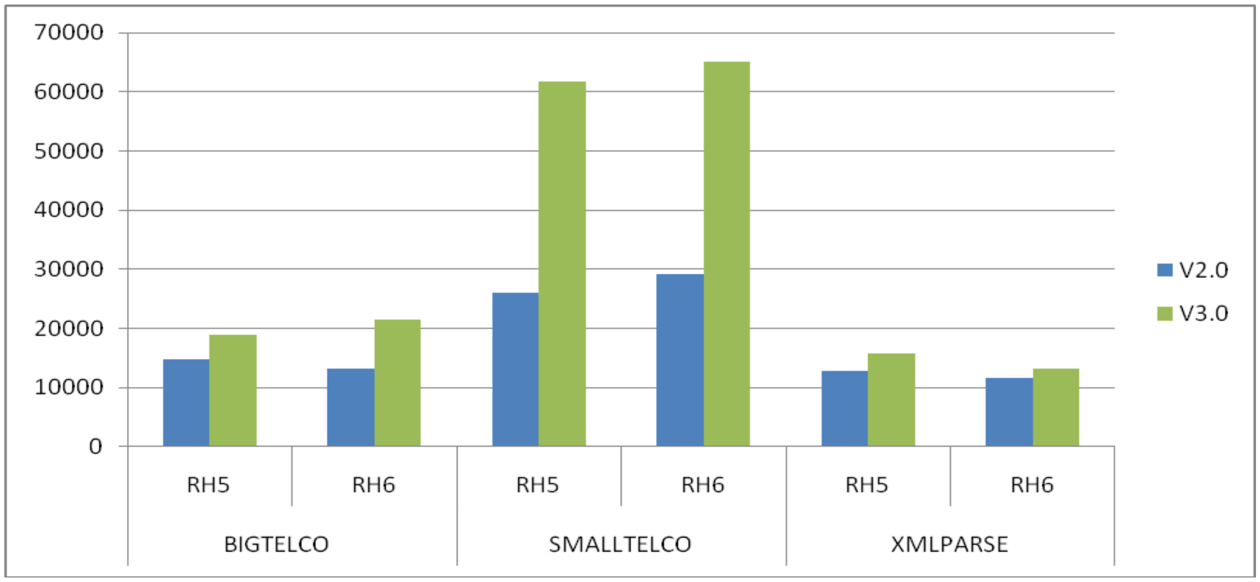


Figure 1 Throughput comparisons of application benchmarks (tuples/sec) [Larger is better]

	BIGTELCO		SMALLTELCO		XMLPARSE	
	RH5	RH6	RH5	RH6	RH5	RH6
V2.0	14760	13268	25975	29143	12758	11610
V3.0	18919	21487	61719	65028	15809	13286
% change	28	62	138	123	24	14

Table 1 Throughput comparisons of application benchmarks (tuples/sec) [Larger is better]

Figure 2 and Table 2 show the variations of the LOGS benchmark improving with v3.0 on RHEL 5.5. In all cases except for the LOGFLT variation on RHEL6.1, the performance of v3.0 is greater than v2.0.   The filter variation (LOGFLT) is slower with v3.0 on RHEL 6.1 compared to RHEL 5.5.

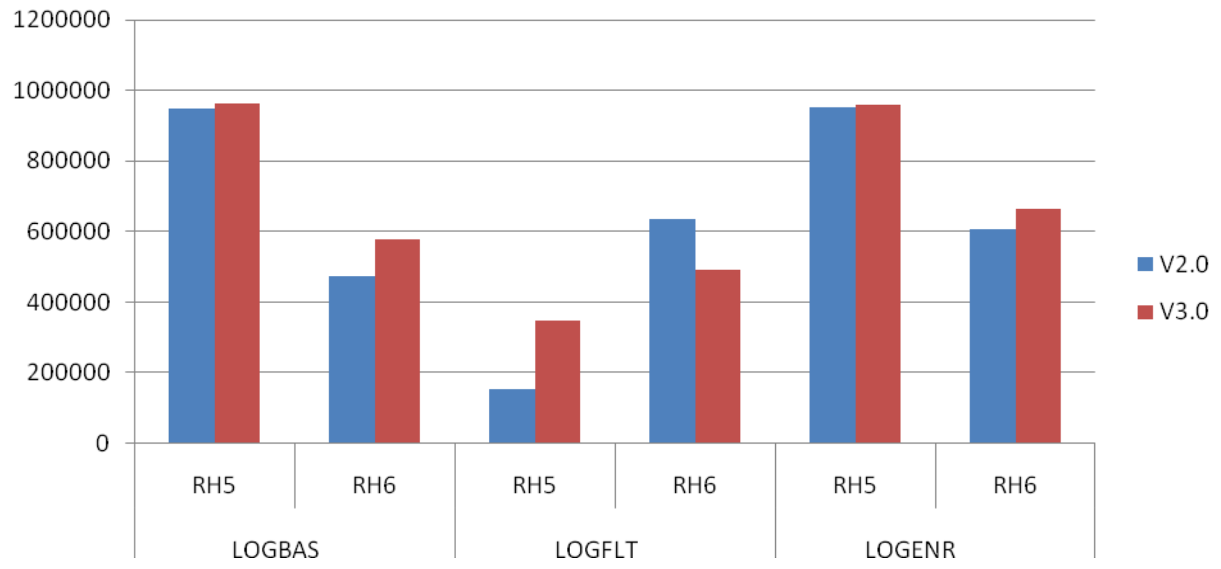


Figure 2 Throughput comparisons of log analysis benchmarks (tuples/sec) [Larger is better]

	LOGBAS		LOGFLT		LOGENR	
	RH5	RH6	RH5	RH6	RH5	RH6
V2.0	947720	475214	151536	633813	952517	605590
V3.0	961921	578369	346840	489820	959820	664498
% change	1	22	129	-23	1	10

Table 2 Throughput comparisons of log analysis benchmarks (tuples/sec) [Larger is better]

Throughput Test

Test Description

We measure the maximum throughput achievable by Streams product using a 2-operator SPL program: a source operator produces a stream that is consumed by a sink operator.
The sink operator performs the measurements; it records the arrival times of the first and last tuples and the number of tuples, together yielding the throughput (in tuples/sec).
Several runs are performed varying the tuple size from 4 bytes to 16K bytes.

The throughput testing is measured in 2 different configurations:

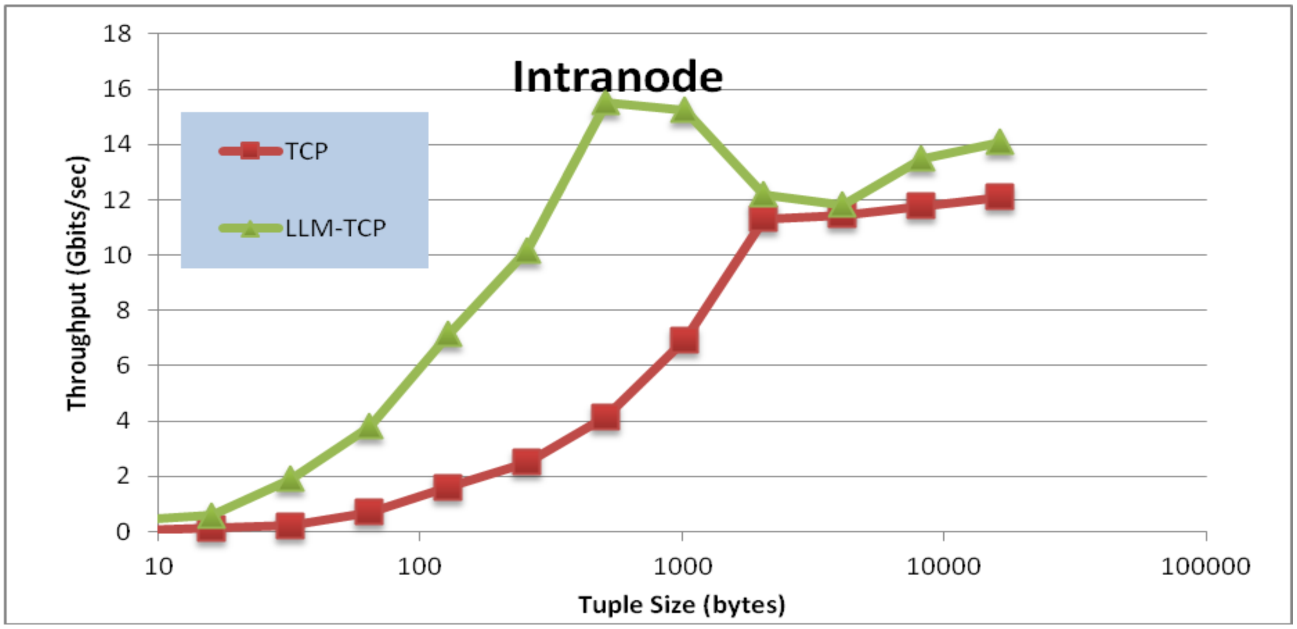
- **Intranode:** source and sink operators are placed on the same host; all communication is therefore within that one host.
- **Internode:** configuration uses 2 hosts, one for the source, the other for the sink operator.

For each configuration above, we measure throughput with 3 different transport protocols supported by the Streams product:

- **TCP Streams :** this is the standard TCP transport built into the OS
- **LLM-TCP:** TCP via LLM transport
- **LLM -IB:** Infiniband via LLM transport

Test Results

Intranode Configuration



.

Figure 3 Intranode throughput (Gbits/sec) with TCP based transports [Larger is better]

tuple size (bytes)	LLM-TCP	TCP Streams	LLM-IB
4	0.11	0.03	0.14
8	0.41	0.06	0.25
16	0.58	0.12	0.49
32	1.92	0.22	0.93
64	3.81	0.70	2.13
128	7.15	1.60	4.19
256	10.17	2.50	7.29
512	15.52	4.15	13.36
1024	15.25	6.94	15.52
2048	12.20	11.32	13.86
4096	11.81	11.45	17.55
8192	13.48	11.79	18.33
16384	14.09	12.10	18.34

Table 3 Intranode throughput (Gbits/sec) with TCP based transports [Larger is better]

Observations:

In an intranode configuration, for tuples beyond 2KB, the throughput flattens out for TCP transport. LLM may provide better performance for tuples under 512 bytes.

Using TCP via the LLM transport (LLM-TCP) provides from 2x to 8x greater throughput than plain TCP, at least up to 1KB tuples. This is due to some amount of batching that occurs in LLM, resulting in fewer system calls being made. For larger tuples, however the performance using the LLM transport, is equivalent to the standard TCP transport

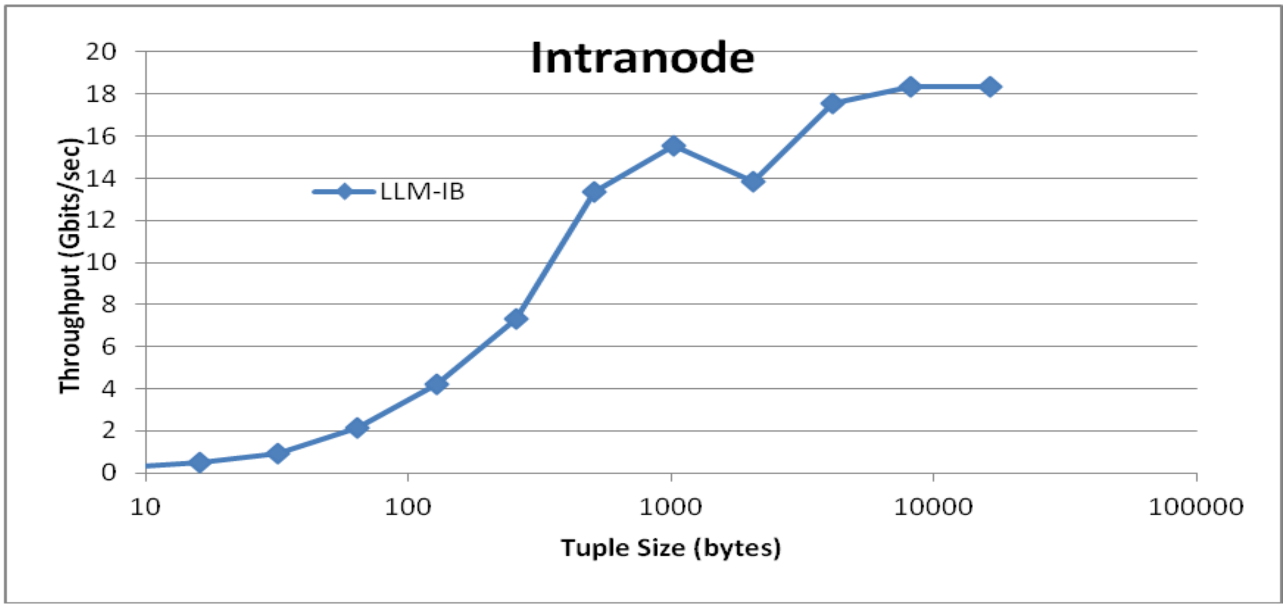


Figure 4 Intranode throughput (Gbits/sec) with Infiniband transport [Larger is better]

With LLM transport via InfiniBand, the throughput is as shown in Figure 4 and Table 3 (column 4). Even though the data is staying within the node, use of the InfiniBand adapter offers better performance for large tuples (4KB and above) than standard TCP by up to 50%.

Internode Configuration

We now consider the same application as before, but the sender and receiver are placed on different hosts (see Figure 5 and Table 4 columns 2, 3 and 4).

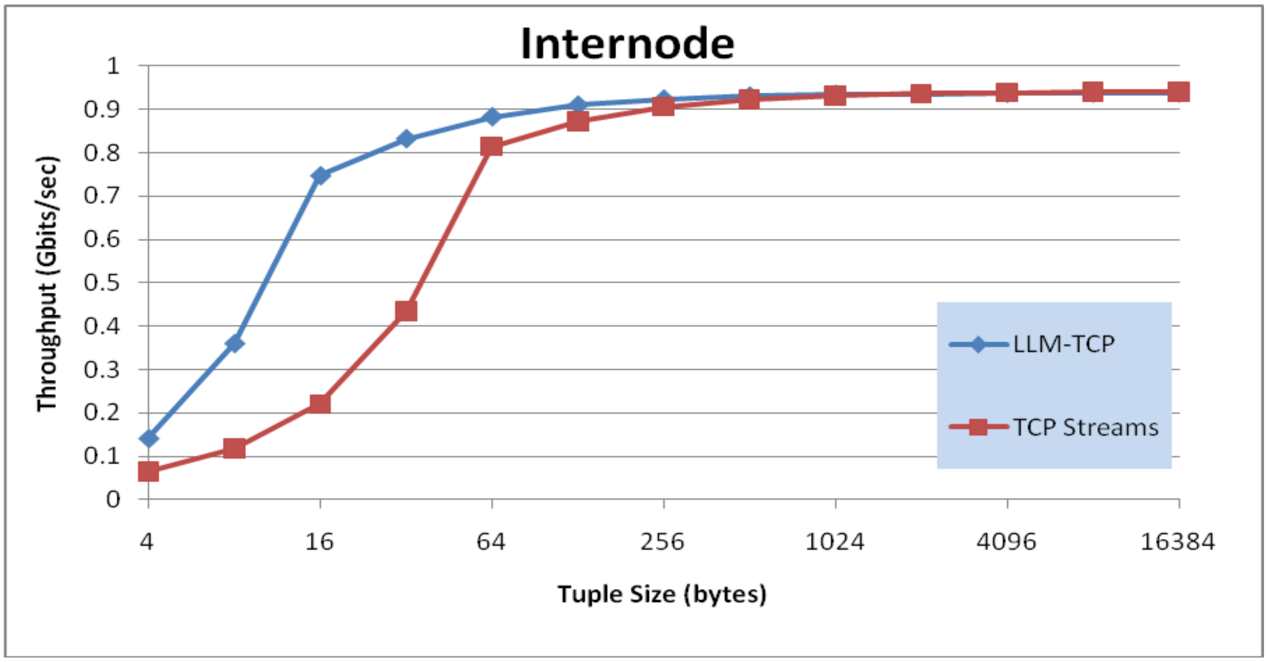


Figure 5 Internode throughput (Gbits/sec) with TCP transport [Larger is better]

tuple size (bytes)	LLM-TCP	TCP Streams	LLM-IB
4	0.14	0.06	0.14
8	0.36	0.12	0.29
16	0.75	0.22	0.52
32	0.83	0.43	1.02
64	0.88	0.81	1.86
128	0.91	0.87	4.31
256	0.92	0.91	7.77
512	0.93	0.92	17.44
1024	0.93	0.93	14.19
2048	0.94	0.94	22.18
4096	0.94	0.94	18.05
8192	0.94	0.94	21.99
16384	0.94	0.94	22.53

Table 4 Internode throughput (Gbits/sec) with TCP transport [Larger is better]

Observations:

In this case, the internode TCP throughput is limited by the 1Gbit/sec capacity of the Ethernet. Using TCP via the LLM option provides the best of both worlds: much higher throughput for small tuples (up to 32 bytes) and in the limit it is identical to the native TCP transport.

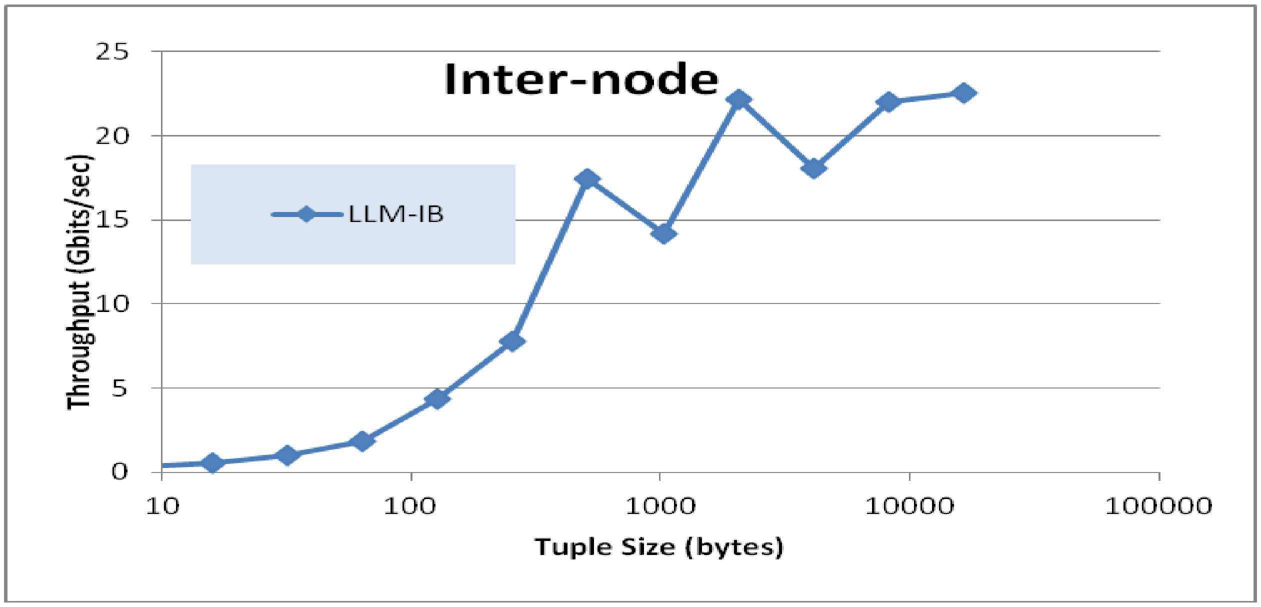


Figure 6 Internode throughput with Infiniband transport [Larger is better]

Infiniband in the internode case has a much higher theoretical capacity relative to TCP: 32 Gbit/sec effective throughput. In Figure 6 and Table 4 (column 4), we see that the throughput achieved with a single sender-receiver pair seems to reach a limit around 20-22 Gbit/sec with larger tuples. Similar to LLM with TCP, using LLM with IB helps achieve good throughput with smaller tuples (up to 1KB). In most applications, the default TCP transport is adequate. If additional performance is required then we suggest building with LLM with TCP and LLM with IB (if hardware is available).

Throughput Comparison to Previous Release

To verify that Streams changes between v2.0 and v3.0 were improvements to its performance, we made a variety of comparison runs.

Intranode Configuration

In Figure 7 and Table 5 columns 2 through 5 we compare the v2.0 and v3.0 performance within a single node using TCP (and LLM over TCP) as the transport between source and sink operators. The plain TCP transport results show a slight degradation for a few tuple sizes, but LLM is largely unchanged. We attribute this to additional functionality in transport services, such as monitoring the health of individual connections.

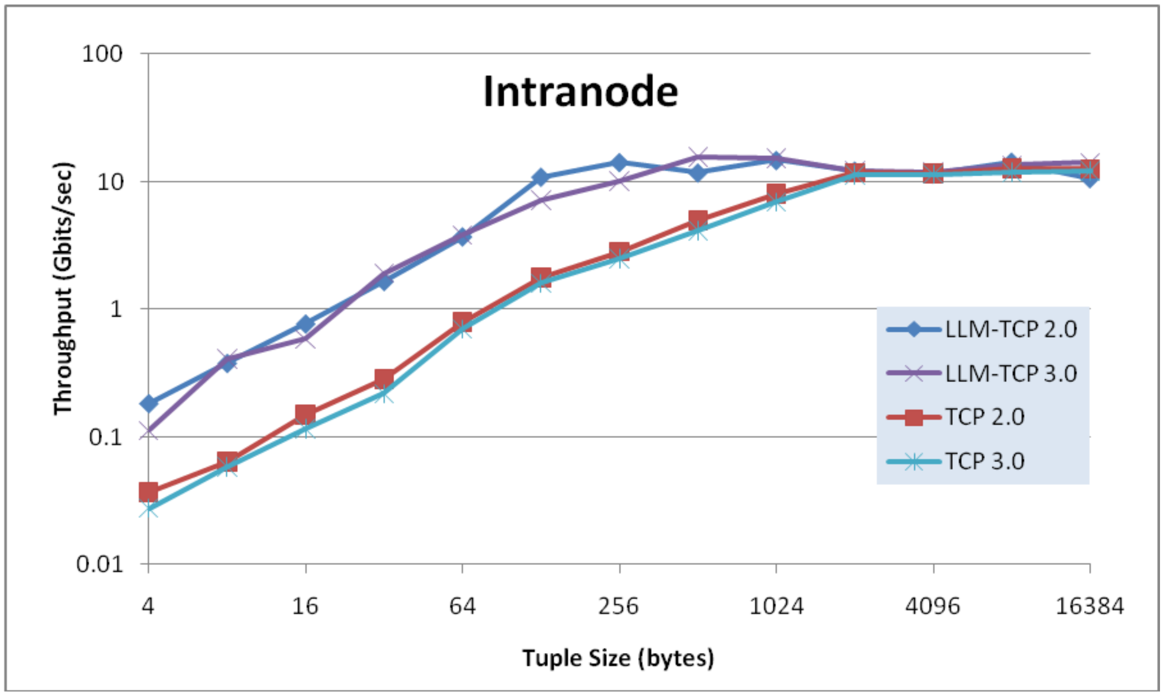


Figure 7 Comparison between Streams v2.0 and v3.0 using TCP intranode [Larger is better]

tuple size (bytes)	LLM-TCP 2.0	LLM-TCP 3.0	TCP 2.0	TCP 3.0	LLM-IB 2.0	LLM-IB 3.0
4	0.18	0.11	0.04	0.03	0.14	0.14
8	0.38	0.41	0.06	0.06	0.31	0.25
16	0.77	0.58	0.15	0.12	0.75	0.49
32	1.66	1.92	0.28	0.22	1.54	0.93
64	3.71	3.81	0.80	0.70	3.24	2.13
128	10.92	7.15	1.76	1.60	5.09	4.19
256	14.21	10.17	2.83	2.50	8.03	7.29
512	11.78	15.52	4.94	4.15	15.40	13.36
1024	14.75	15.25	8.08	6.94	13.70	15.52
2048	12.20	12.20	11.75	11.32	13.61	13.86
4096	11.56	11.81	11.63	11.45	17.82	17.55
8192	14.35	13.48	12.63	11.79	17.84	18.33
16384	10.64	14.09	12.61	12.10	17.70	18.34

Table 5 Intranode Throughput Comparison (Gbits/sec) [Larger is better]

In Figure 8 and Table 5 columns 6 and 7 we have the results when still staying with a single node, but switching to InfiniBand transport.

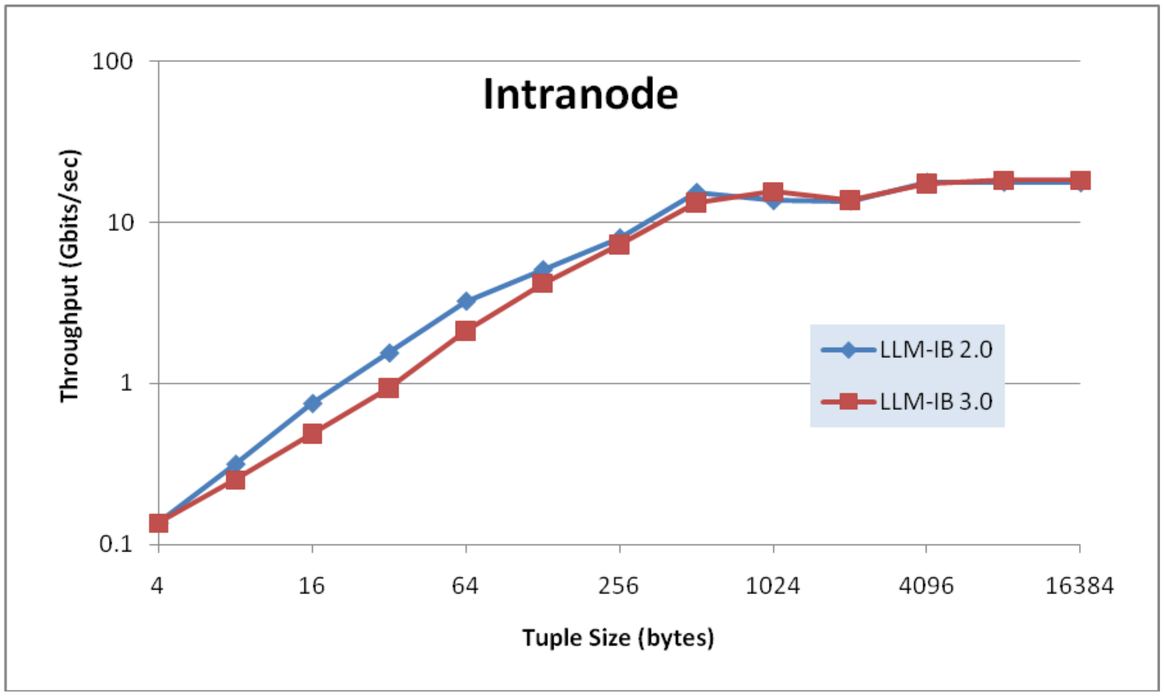


Figure 8 Comparison between Streams v2.0 and v3.0 using InfiniBand intranode [Larger is better]

Internode Configuration

Internode throughput comparisons using TCP can be seen in Figure 9 and Table 6 columns 2 through 5. The only significant difference between Streams v2.0 and v3.0 is for standard TCP. We attribute this to additional functionality in transport services, such as monitoring the health of individual connections.

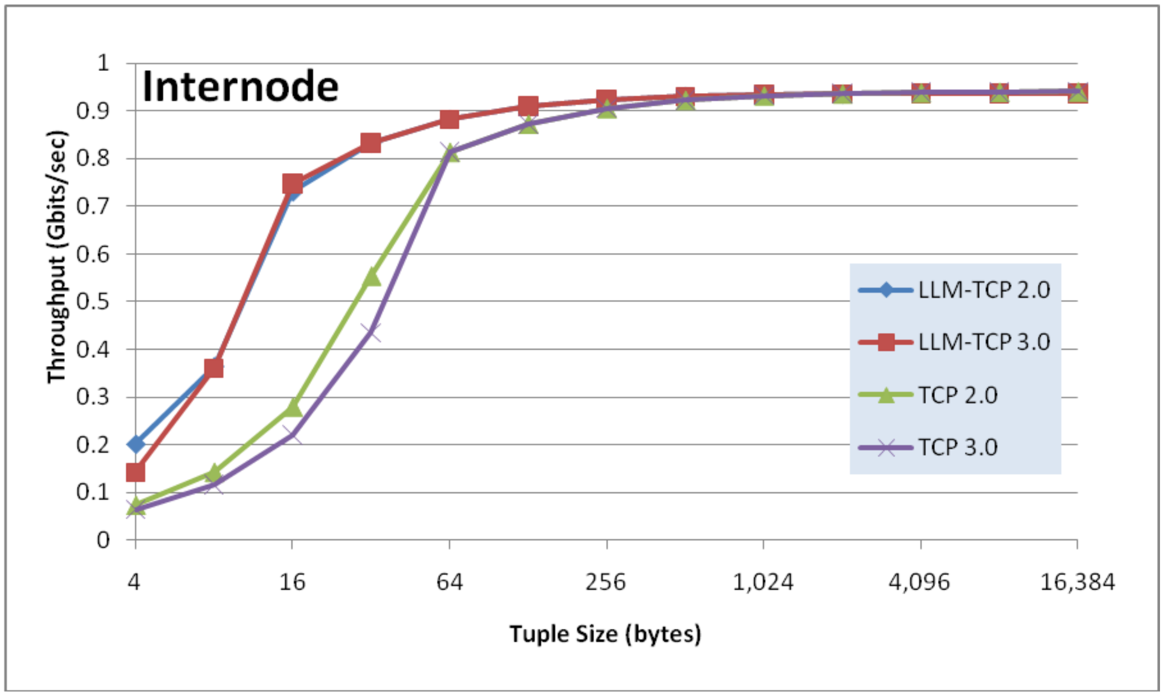


Figure 9 Comparison of Streams v2.0 with v3.0 using TCP between 2 nodes [Larger is better]

tuple size (bytes)	LLM-TCP 2.0	LLM-TCP 3.0	TCP 2.0	TCP 3.0	LLM-IB 2.0	LLM-IB 3.0
4	0.20	0.14	0.07	0.06	0.18	0.14
8	0.36	0.36	0.14	0.12	0.34	0.29
16	0.73	0.75	0.28	0.22	0.58	0.52
32	0.83	0.83	0.55	0.43	1.52	1.02
64	0.88	0.88	0.81	0.81	2.94	1.86
128	0.91	0.91	0.87	0.87	6.81	4.31
256	0.92	0.92	0.91	0.91	8.95	7.77
512	0.93	0.93	0.92	0.92	14.49	17.44
1024	0.93	0.93	0.93	0.93	15.21	14.19
2048	0.94	0.94	0.94	0.94	22.01	22.18
4096	0.94	0.94	0.94	0.94	20.15	18.05
8192	0.94	0.94	0.94	0.94	22.44	21.99
16384	0.94	0.94	0.94	0.94	23.09	22.53

Table 6 Internode Throughput Comparison (Gbits/sec) [Larger is better]

The internode comparison between Streams v3.0 and v2.0 using InfiniBand is shown in Figure 10 and Table 6 columns 6 and 7. The apparent differences for larger tuples we attribute mostly to noise, but there are differences for smaller tuples as well.
We attribute this to additional functionality in transport services, such as monitoring the health of individual connections

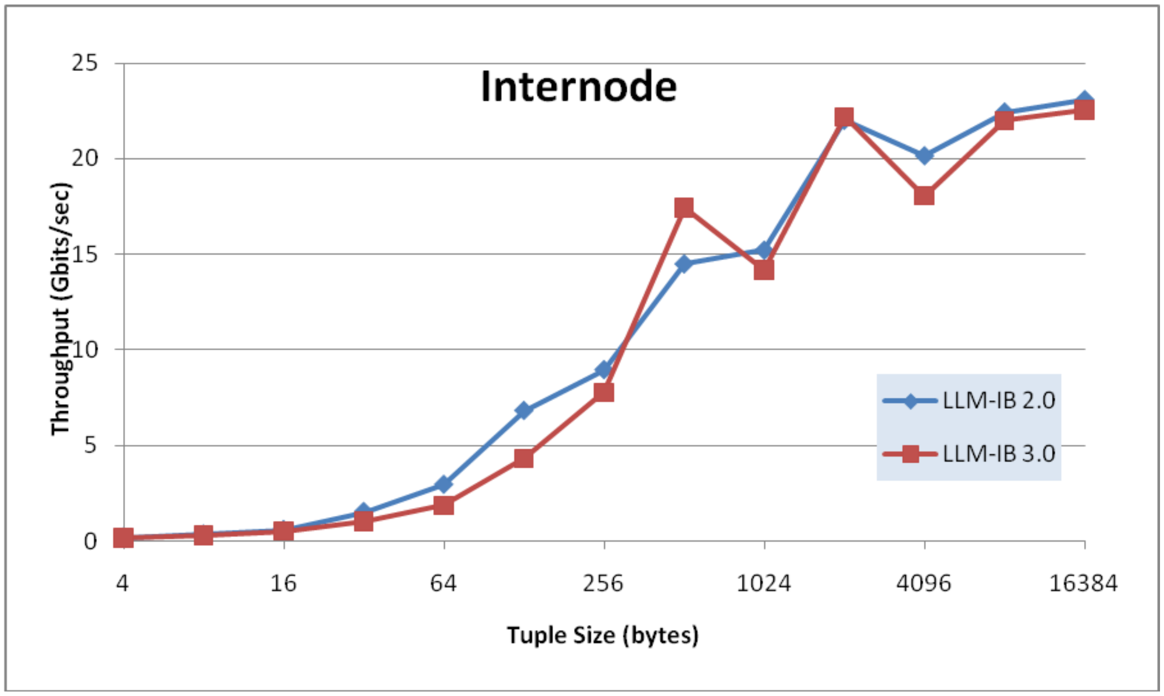


Figure 10 Comparison of Streams v2.0 with v3.0 using InfiniBand between 2 nodes [Larger is better]

Latency Test

Test Description

Clock skew is always an issue with latency measurements. We control that by always doing round-trip measurements: from a source operator, to one that receives tuples and passes them along, to the final sink operator. The source operator records the current time in each tuple, and the sink operator compares that time with the arrival time there at the sink operator in order to compute the round-trip latency. When measuring the latency between 2 different nodes, clock skew is avoided by always putting the source and sink operators on the same node, with the relay operator being on the other node. That way, the timestamp recorded by the source operator is compared to a timestamp read by the sink operator from the very same clock. We also avoid the use of the high-performance cycle counter as a clock since these can be different on different cores within a single node. This avoids all of the problems that arise when trying to keep the clocks on 2 different nodes in sync.

Various queueing and batching effects can come into play as the workload varies, and these can have a significant impact on latency. To see if that's the case, we measure latency while having the source offer tuples at both a rate of 1,000 tuples per second and also, separately, at a rate of 5,000 tuples per second.

Results: Intranode

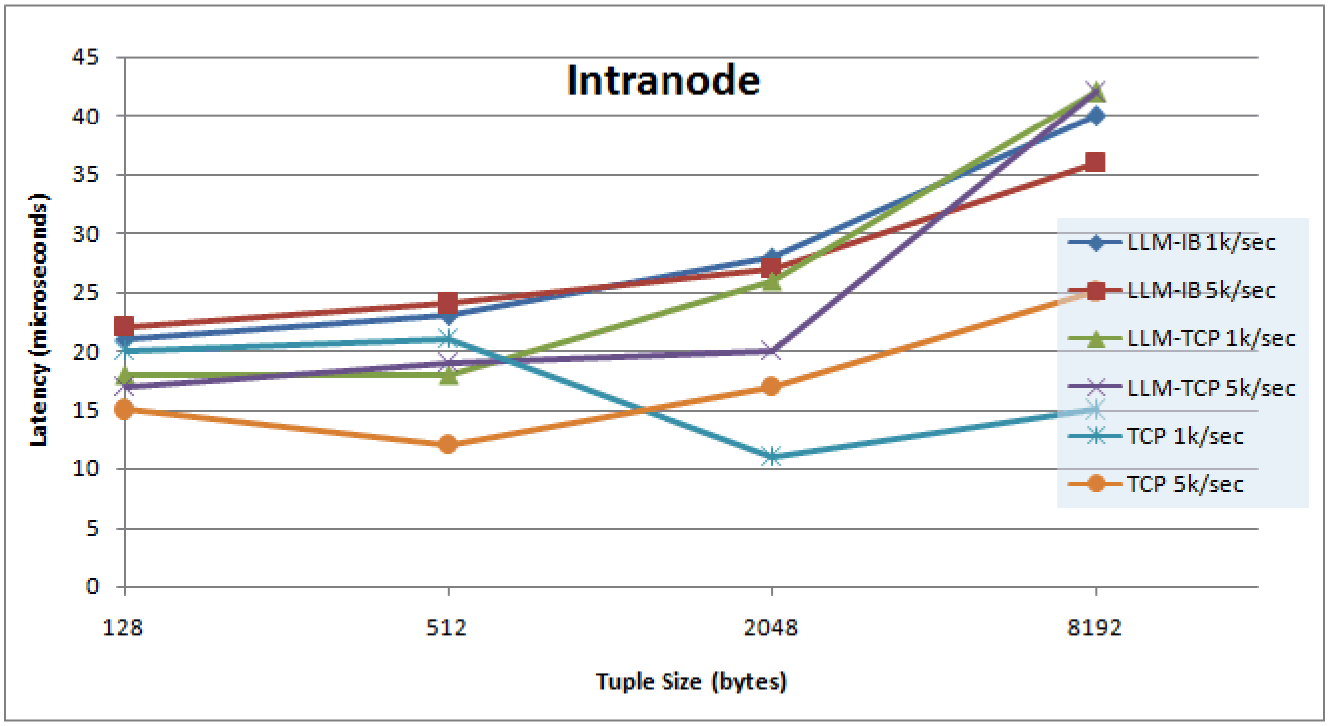


Figure 11 Latency within a single node using different transports [Smaller is better]

Observations:

When the source, relay, and sink operators are all on the same node, there is little to distinguish the latencies seen, regardless of the Streams transport chosen, as we see in Figure 11. LLM batching is based on timers, so the lack of variation among the different flavors and flow rates for LLM is unsurprising; it behaves as designed. Standard TCP cannot achieve low latency at higher throughput because it is not doing the advanced batching of LLM.

Results: Internode

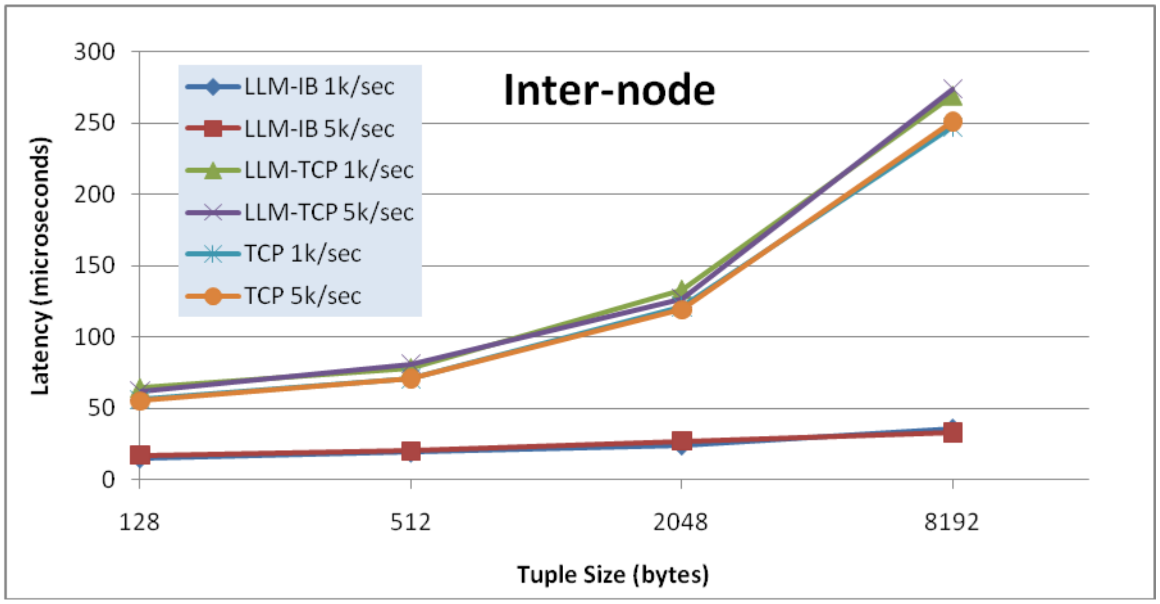


Figure 12 Latency between 2 nodes using different transports [Smaller is better]

Observations:

When we look at the internode results shown in Figure 12 (source and sink operators are on one node, while the relay operator is on another node), we see that LLM over InfiniBand protocol has much lower latencies. Unlike the intranode measurements, here we have a clear upward trend as the tuple size grows. The time on the wire really does change when sending more data.

Comparison to Previous Release

Results: Intranode

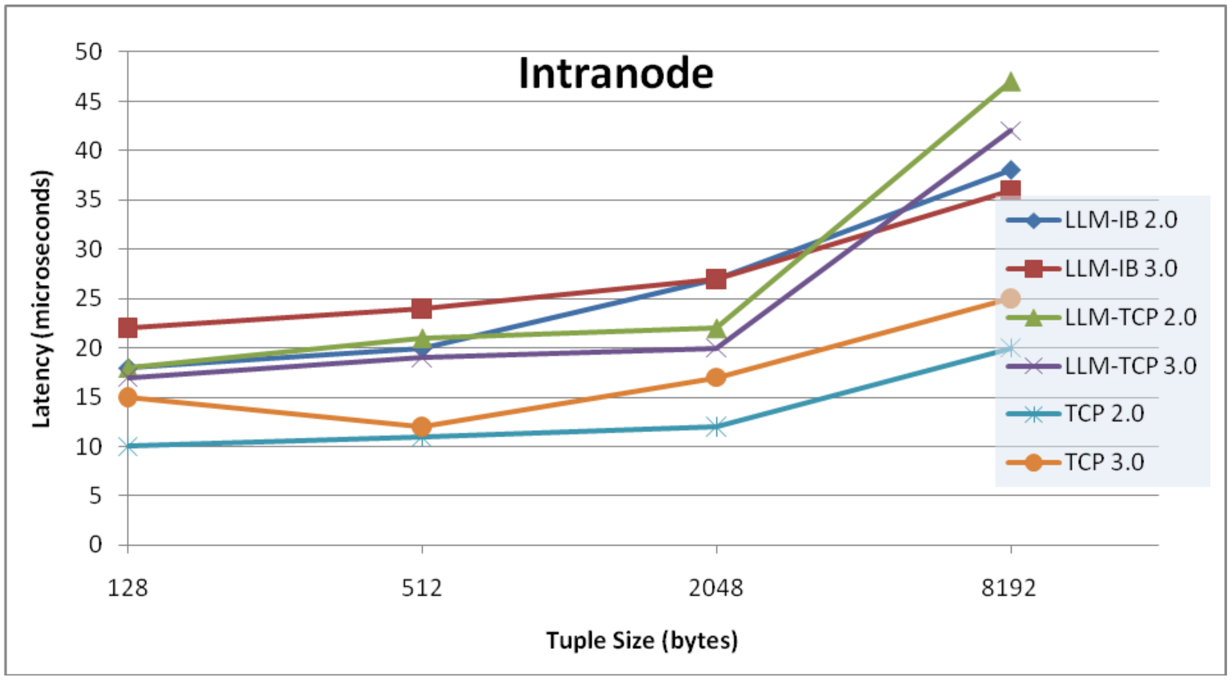


Figure 13 Comparison of Streams v2.0 and v3.0 latencies within a single node [Smaller is better]

Observations:

Intranode latency with plain TCP and with LLM over IB was unchanged from Streams v2.0 to v3.0, as we can see in Figure 13. The one difference is an improvement (reduction, that is) in latency with LLM over TCP protocol. Intranode throughput numbers for these tuple sizes also showed a big improvement. This improvement is perhaps related to the newer level of LLM used in v3.0.

Results:Internode

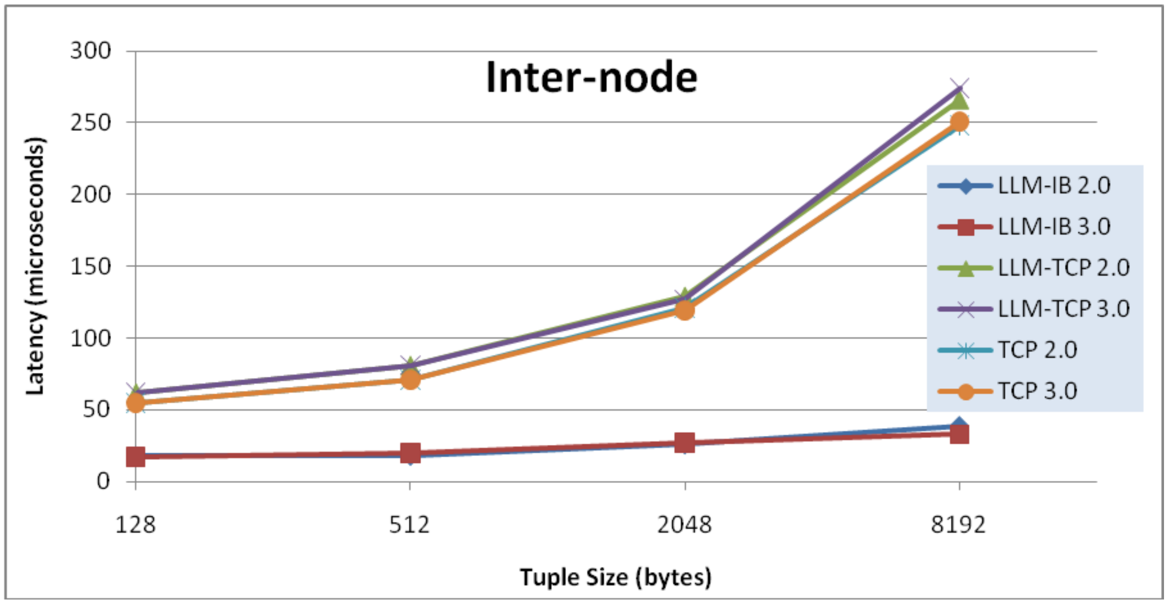


Figure 14 Comparison of Streams v2.0 and v3.0 latencies between 2 nodes [Smaller is better]

Observations:

Figure 14 shows the result of measuring internode latency at 5,000 tuples per second. Latencies for LLM over InfiniBand latencies were unchanged.

New Features

Applications, which contain an **aggregate operator with tumbling windows**, may experience a speedup when using v3.0. Throughput measurements of this form of aggregate operator in v3.0 shows 20%-30% speedup compared to the same test using v2.0.

For applications, which exploit dynamic composition, v3.0 has introduced filtering when exporting tuples. When a **filter parameter** is specified on the **Import** operator, it will instruct the corresponding **Export** operator to filter the tuples before exporting them. We have measured throughput improvements of 30%-40% depending on the number of connection subscribers. Figure 15 best describes this.

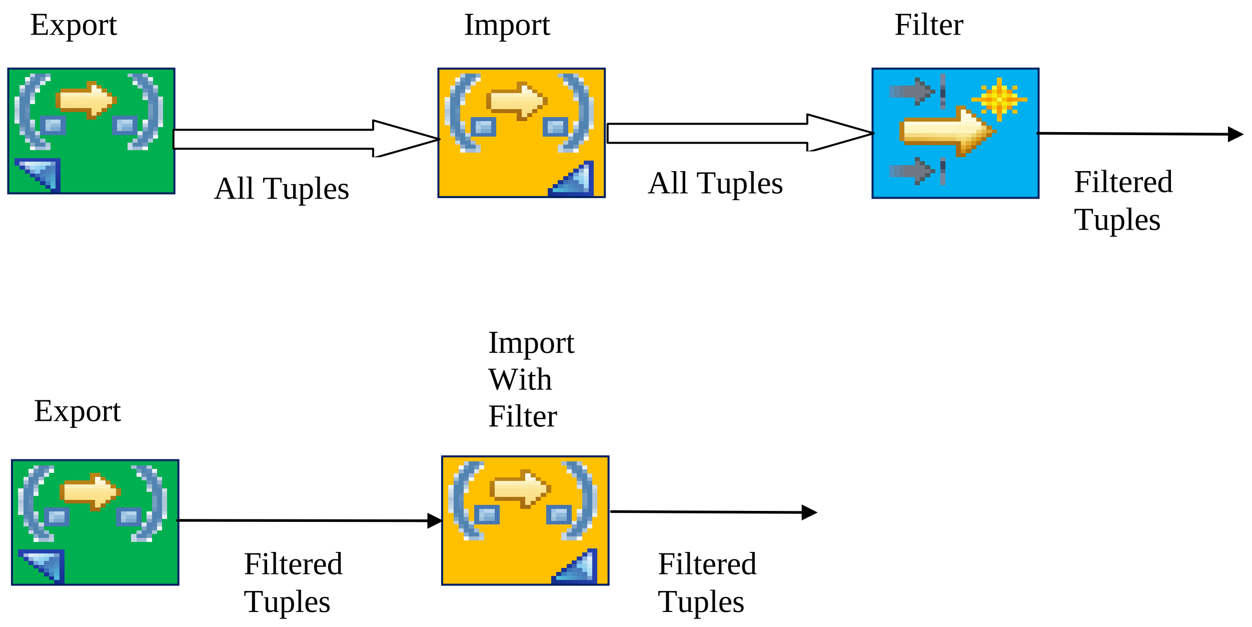


Figure 15 Filter on Export operator

New tracing and logging capabilities were added to v3.0. We created tests to quantify the overhead of various logging and tracing levels. Our results are summarized in Table 7. We observed that the default tracing level has insignificant impact on performance relative to no tracing. However, full tracing is very costly and should only be used when necessary. Note the tests created are variations of the throughput and application tests previously described.

Performance Test	Overhead of default tracing relative to no tracing	Overhead of full tracing relative to no tracing	Notes
Intranode basic throughput test (operator to operator)	0 to 3% slower	80 times to 500 times slower	Depends on tuple size
Internode basic througput test	0 to 3% slower	6 times to 600 times slower	Depends on tuple size
BIGTELCO application	3% slower	13 times slower	

Table 7 Relative overhead of logging and tracing

Another new feature in v3.0 is **stream visualization**. This allows one to visualize samples of any streams in an application. Our conclusion is that this adds **no overhead to the application** even in extreme cases when the application is dominated by transport performance.

Conclusions

A wide variety of changes have been introduced with Streams v3.0. Functional changes in data transport have slightly reduced performance in return for the improvements in application and system problem solving. On the other hand, changes in the SPL language processor and runtime support have significantly improved performance. The overall effect of these changes, as seen in the performance of various application codes, is a significant improvement.

In the transport area, we see a reduction in throughput in our transport stress tests. That is, these tests do no application processing whatsoever. This reduction is seen most readily for small (32-byte and smaller) tuples being transmitted over TCP between operators that have been placed on the same system. This combination of factors is precisely those that would most greatly bring out any changes in transport performance.

In the SPL and runtime area, there are changes to such things as the Aggregate operator and the addition of an option to filter out exported tuples before they are transmitted. These translate into both reduced SPL operator CPU loads and reduced communication traffic, with corresponding benefits to overall, cluster-wide performance.

These overall benefits can be seen in the improvements in throughput for the various applications we have measured. For example, the 12-operator SMALLTELCO application improved by 100% when moved from Streams V2.0 to V3.0. At the same time, the 90-operator BIGTELCO application saw improvements in the 30% to 60% range with Streams V3.0. We anticipate that similar applications will show improvements of a similar magnitude.

While we have provided observations about performance under different conditions. Providing a guide to the programmer based on these and other results is beyond the scope of this document.

Appendix: Test Environment

IBM obtained all performance data contained in this publication in the specific operating environment and under the conditions described herein. Performance obtained in other environments may vary, so customers should conduct their own testing.

IBM obtained results using a cluster of IBM BladeCenter blades with specific hardware details as follows:

- IBM HS-22 Blades (model 7870-AC1) each with:
    - Two Intel® Xeon® X5570 CPU with:
        - 2.98 GHz clock speed
        - 4-cores
        - Cache: 32KB L1 (per core), 256KB L2 (per core), 8MB L3 (shared)
        - 6.4 GigaTransfer/sec Intel® QPI interconnect
    - 32GB RAM
    - 1Gbit Ethernet NIC on motherboard
    - Infiniband QDR NIC: Mellanox Technologies ConnectX VPI PCIe 2.0 MT25408
        - Up to 32 Gbit/sec effective throughput
        - 1.2 microsecond MPI ping latency
- Infiniband switch module: IBM BCH QDR HSSM, Voltaire 40Gb/s Switch module
- OS: RedHat Enterprise Linux 5.5 64-bit
- Infiniband driver:OFED v1.5.2

The measurements correspond to the following versions of Streams:

- InfoSphere Streams v2.0 GA version
- InfoSphere Streams v3.0 GA version

Some variations of transport tests use LLM.
In those variations, the LLM configuration is specified as follows:

- For throughput tests we specified: MinBatchingMicro = 400 and MaxBatchingMicro = 800.
- For Latency tests we specified: MinBatchingMicro = 25 and MaxBatchingMicro = 50.

Glossary

PE : processing element (part of a steams application which runs as a separate process)

SPL: Stream Processing Language.

CDR: Call Detail Record.

LLM: IBM® WebSphere® MQ Low Latency Messaging

For more information

To learn more about IBM InfoSphere Streams and associated products to build them, visit:

http://www.ibm.com/software/data/infosphere/streams/

IBM®