STORM-2306
preliminary perf measurements

- Sapin Amin and Roshan Naik

Overview:

STORM-2306 brings the following changes to Apache Storm to improve performance:

1. Redesigned messaging subsystem for **communication within the worker** process.
2. A light weight **Back-pressure** model that is tightly integrated into the messaging subsystem. A small number of changes were required in the inter-worker communication path to communicate back-pressure between workers.

These performance sensitive areas were NOT tackled by 2306.

1. **ACK mode:** Numbers indicate that the ACKing-path is a critical bottleneck that needs lot of improvement.
2. **Inter-worker communication:** This has also surfaced as a key bottleneck that needs to be optimized for Storm to realize its potential.
3. **OutputCollectors:** Further optimizations are possible in the SpoutOutputCollector & BoltOutputCollector's emit() path, before the tuple hits the core messaging layer.
4. **Trident:** No changes were made specifically for improving Trident perf, it is unclear at this time how much potential there is or what bottlenecks exist. Nevertheless, Trident topologies were included as part of these runs.

The aim of these runs is not to benchmark the peak abilities of either STORM-2306 or master. Such runs can require fine tuning of several parameters. The goal here is to catch bugs and performance problems by running a collection of topologies on master as well as 2306 in many different configurations. We compare numbers of master and 2306 for similar configurations of the each topology.

There are close to 300 different runs recorded here. The key settings that were varied :

- With/without ack,
- Single/multi worker. Each worker on a separate host.
- Spout, Bolt, Acker parallelism
- topology.max.spout.pending
- Batch sizes (`topology.disruptor.batch.size` for master and `topology.executor.receive.buffer.size` for 2306)

Each additional setting, generally doubles the number of runs due to the combinatorial nature of the problem. Thus only some of the 'all possible combinations' for these settings have been run. Throughput, latency, mem consumption & cpu utilization were recorded. In a few cases, there were issues collecting the mem & cpu usage and consequently the numbers were omitted in those cases.

Common Settings for all runs:

```
-c topology.worker.childopts="-XX:GCLogFileSize=10M
-Dworker.memory_limit_mb=4096 -Xmx4096m"
-c topology.disable.loadaware.messaging=true
-c topology.stats.sample.rate: 0.0005
```

Setup:

Java Version: 1.8.0_112
Storm-2306: #84cb990
Storm-master branch: #aaebc3b

Cluster:
– 5-node cluster (1 nimbus and 4 supervisor nodes) running Storm master
– 5-node cluster (1 nimbus and 4 supervisor nodes) running Storm 2306
– 3-node separate Zookeeper cluster

Hardware:
All nodes had the following configuration:

1. **CPU:** sockets = 2 sockets, 6 physical cores per socket, Hyper threaded. (2 sockets x 6 cores x 2 hyper threads = 24 logical cores). Intel Xeon CPU E5-2630 0 @ 2.30GHz
2. **Memory:** 126 GB
3. **Network:** 10 GigE
4. **Disk:** 6 disks. Each 1TB. 7200 RPM.

1. TVL Topology

This is a WordCount topo that is part of Storm. It allows configuring the rate at which sentences should be generated by the spout. The topology may or may not be able to actually sustain that rate. For each configured rate so we measure the actual rate that is achieved (sentences/sec) along with the latency, mem and cpu.

Single Worker

(TVL1) MASTER -c topology.max.spout.pending=500 -c topology.workers=1 **(default bachSz=100)**

(run by roshan)

Config	Spout/Splitter/Counter/Ack	Sentences	Backlo	Avg	Avg	Avg

ured Rate	er parallelism	/sec	g millis	CPU in cores	Mem node1 mb	Late ncy
100,000	SO=1 | SL=1 | C=1 | Ack=1	**74.4k**	71,047	3.5	63.3	3.3
200,000	SO=1 | SL=2 | C=2 | Ack=2	**73k**	178,186	5.19	60.5	3.3
500,000	SO=4 | SL=10 |C=6 | Ack=4	229k	14,9451	13	65	4.675

(TVL1) STORM-2306 **-c topology.max.spout.pending=500** -c topology.workers=1 **(default bachSz=1)**

(run by roshan)

Config ured Rate	Spout/Splitter/Counter/Ack er parallelism	Sentences /sec	Backlo g millis	Avg CPU in cores	Avg Mem mb	Avg Late ncy (ms)
100,000	SO=1 | SL=1 | C=1 | Ack=1	100k	2	2	58.5	0.6
200,000	SO=1 | SL=2 | C=2 | Ack=2	199k	2	4	66	0.78
500,000	SO=4 | SL=10 |C=6 | Ack=4	316k	100,673	13	53	5.6

Observations (msp=500):

- Master had problem keeping up throughput at lower parallelism. 2306 was able to achieve configured throughput at same parallelism... and consuming similar CPU.
- For sustained throughputs, we see much lower latencies.
- Master peaked at 229k. 2306 peaked at 316k.

(TVL2) MASTER -c **topology.max.spout.pending=5000** -c topology.workers=1 **(run by roshan)**

Config ured Rate	Spout/Splitters/counters/A ckers	Avg tuples/sec	Backlo g millis	Avg CPU in cores	Avg Mem mb	Avg Late ncy (ms)
100,000	SO=1 | SL=1 | C=1 | Ack=1	100k	15.3	4.13	104.8	**21.2**
100,000	SO=2 | SL=6 | C=3 | Ack=2	100k	8.5	8.17	749	5.7

100,000	SO=5 | SL=10 | C=5 | Ack=4	99.6k	8.9	10.4	?92	8.3
200,000	SO=1 | SL=2 | C=2 | Ack=2	**147.1k**	76,011	7.45	132.3	**31.16**
250,000	SO=5 | SL=10 | C=5 | Ack=4	**166.7k**	150,920	15	1,097	**101.8**
500,000	SO=2 | SL=5 | C=3 | Ack=2	**144.8k**	195,566	12.2	996	**63.4**
500,000	SO=4 | SL=10 |C=6 | Ack=4	**177k**	171,462	14.7	1,177	**107**

(TVL2)STORM-2306 -c **topology.max.spout.pending=5000** -c topology.workers=1 **(run by roshan)**

Configured Rate	Spout/Splitters/counters/Ackers	Avg tuples/sec	Backlog millis	Avg CPU in cores	Avg Mem mb	Avg Latency (ms)
100,000	SO=1 | SL=1 | C=1 | Ack=1	99.6k	2	2	68.8	0.95
100,000	SO=2 | SL=6 | C=3 | Ack=2	99.6k	1	3.4	236.9	0.984
100,000	SO=5 | SL=10 | C=5 | Ack=4	99.6k	96,777	5	296	1.5
200,000	SO=1 | SL=2 | C=2 | Ack=2	**167.4k**	54,137	8.4	58.3	**30.11**
250,000	SO=5 | SL=10 | C=5 | Ack=4	**249.4k**	1	8	516	**20.5**
500,000	SO=2 | SL=5 | C=3 | Ack=2	**145.7k**	195,283	12	**1509**	**65.5**
500,000	SO=4 | SL=10 | C=6 | Ack=4	**326k**	102,804	12	563	**65.5**

Observations (msp=5k):

- Neither was able to deliver 200k or 500k rates with the given parallelism. 2306 attained higher throughputs in these modes with lower CPU usage.
- When configured rate was achieved(i.e 100k), latencies are generally much better, otherwise similar.

Multi Worker

(TVL3) Master -c topology.max.spout.pending=5000,**-c topology.workers=4**

Configured Rate	Spout/Splitters/counters/Ackers	Avg tuples/sec	Backlog (ms)	Avg CPU in cores	Avg Mem (mb)	Avg Latency	Max Latency
100,000	SO=1 | SL=1 | C=1 | Ack=1	100,681	7883.15	8.74	265.11	41.49	47.45
100,000	SO=1 | SL=2 | C=2 | Ack=2	99,996	11.86	10.18	297.66	7.23	14.35
100,000	SO=1 | SL=4 | C=3 | Ack=2	99,999	12.36	12.34	241.12	6.01	21.44
100,000	SO=2 | SL=6 | C=3 | Ack=2	100,006	11.69	12.06	261.25	6.23	18.52
100,000	SO=3 | SL=8 | C=4 | Ack=3	100,011	12.34	13.01	274.41	6.35	15.19
100,000	SO=5 | SL=10 | C=5 | Ack=4	100,004	12.94	15.39	416.59	7.75	42.99
250,000	SO=2 | SL=5 | C=3 | Ack=2	242,028	24,093	21	279	35.4	
250,000	SO=2 | SL=5 | C=3 | **Ack=0**	250k	0.6	12	411	-	-
500,000	SO=2 | SL=5 | C=3 | Ack=2	**219,466**	246,314	19.85	312.82	40.71	41.59
500,000	SO=4 | SL=10 | C=6 | **Ack=0**	500k	0.7	22.8	2,175	-	-
500,000	SO=4 | SL=10 | C=6 | Ack=4	**296,624**	171,638	31.20	1200.5	53.66	96.53
500,000	SO=4 | SL=16 | C=8| **Ack=0**	500k	0.7	26.3	1,362	-	-
500,000	SO=4 | SL=16 | C=8| Ack=8	**273,348**	74,213	34	1267.9	59.2	
1mill	SO=4 | SL=16 | C=8| **Ack=0**	**725k -> 12k**	93,581	**24.6-> 2.8**	4,311		
1mill, BP = enabled	SO=4 | SL=16 | C=8| **Ack=0**	**735k <-> 0k**	-	**-**	-	-	-

(TVL3) Storm-2306, -c topology.max.spout.pending=5000**,-c topology.workers=4**
(run by roshan)

Configured Rate	Spout/Splitters/counters/Ackers	Avg tuples/sec	Backlog (ms)	Avg CPU in cores	Avg Mem mb	Avg Latency	Max Latency
100,000	SO=1 | SL=1 | C=1 | Ack=1	100k	41	5.37	245	23	
100,000	SO=1 | SL=2 | C=2 | Ack=2	100k	5	5	225	3.87	
100,000	SO=1 | SL=4 | C=3 | Ack=2	100k	6	6	214	2.8	

100,000	SO=2 | SL=6 | C=3 | Ack=2	100k	6	6	251	5.5	
100,000	SO=3 | SL=8 | C=4 | Ack=3	99.5k	5	7	267	2.84	
100,000	SO=5 | SL=10 | C=5 | Ack=4	99.6k	5	7.9	279	7.67	
250,000	SO=2 | SL=5 | C=3 | Ack=2	242.4k	17,785	14	278	35.3	
250,000	SO=2 | SL=5 | C=3 | **Ack=0**	250k	0	11.3	381		
500,000	SO=2 | SL=5 | C=3 | Ack=2	**198.4k**	159,213	12.6	265	38.4	
500,000	SO=4 | SL=10 | C=6 | **Ack=0**	500k	1	22.8	2,175	-	-
500,000	SO=4 | SL=10 |C=6 | Ack=4	**238.2k**	143,578	17.6	918.3	57.01	
500,000	SO=4 | SL=16 | C=8| **Ack=0**	500k	0	12	411	-	-
500,000	SO=4 | SL=16 | C=8| Ack=8	**235.4k**	140,320	18.75	877.5	55.9	
1mill	SO=4 | SL=16 | C=8| **Ack=0**	**760k**	93,581	24.1	2,302		

Observations:[updated Dec 21]

- At 100k rate, both delivered on configured rate. Latencies are better again on 2306.
- At 250k, both achieved similar latency and throughput (albeit lower than 250k), 2306 had lower CPU.
- With ACKing enabled in multiworker mode, when configured rate far exceeded the limits of what the topos could achieve (i.e at 500k), master had better performance.
- CPU is consistently lower for 2306. Memory consumption is slightly better as well.
- Here inter-worker communication path seems to be a throughput bottleneck.
- **Back Pressure:** At 1mill rate with ack disabled, master started out with a throughput of 725k/s and kept degrading. It reached 12k/s after 8mins (at the end of the run). CPU usage dropped from 24.6 to 2.8 cores in the same period. 2306 delivered consistent throughput. With BP enabled on master, the drop in throughput was more rapid and fluctuated within the range of 725k/s and 0k/s... as the BP engaged and disengaged. On the other hand, 2306 (with its integrated BP) was able to deliver consistent throughput which fluctuated within the range 916k/s and 623k/s (avg=760k/s) over the same duration.

2. ConstSpout -> NullBolt Topo

Spout emits a constant stream of tuples (10 bytes each) that the bolt simply accepts and discards. This stresses the Spout -> Bolt communication path.

Single Worker

(CN1) Master, Worker=1, acker=0, -c topology.max.spout.pending=500, 1000, 5000, 10000, -c topology.stats.sample.rate=0.0005

disruptor.batch.size	Throughput	Avg CPU Node 1	Avg Mem Node1
1	1,455,412	281	504
100	**2,384,241**	**193**	**486**
500	2,037,991	185	482
1000	1,719,066	172	464

(CN1) Storm 2306, Worker=1, acker=0,
-c topology.max.spout.pending=500, 1000, 5000, 10000,

producer.batch.size	Throughput	Avg CPU Node 1	Avg Mem Node1
1	**5,002,012**	134	1298
100	**5,088,149**	132	1319
500	5,001,962	131	1312
1000	4,884,724	131	1319

Observations:

- Lowest 2306 throughput is 2x better than the best throughput on master.
- 2306 has lower CPU usage. Higher mem usage attributable to higher throughput.

(CN2) Master, Workers=1, Ackers=1, -c topology.max.spout.pending=500, 1000, 5000, 10000,

disruptor.batch.sz	max.spout.pending	Throughput (msgs/sec)	Avg latency (ms)	Avg max latency	Avg CPU Node 1	Avg Mem Node1

1	500	323,724	**0.1**	**0.1**	500	353
100	500	109,329	1.4	1.4	111	297
500	**500**	**79,991**	**2.1**	**2.1**	**88**	**322**
1000	**500**	**77,208**	2.1	2.1	86	305
1	1000	300,237	**0.1**	**0.1**	505	345
100	1000	234,124	1.1	1.2	214	331
500	1000	**159,583**	2.0	2.0	143	353
1000	1000	**161,753**	2.0	2.0	144	360
1	5000	297,783	2.7	2.8	521	405
100	**5000**	**600,341**	**0.8**	**0.8**	**361**	**345**
500	**5000**	**603,208**	**1.1**	**1.2**	**363**	**338**
1000	5000	527,504	1.2	1.2	352	348
1	10000	311,416	8.9	10.2	540	427
100	10000	**631,095**	**0.8**	**0.8**	**383**	**383**
500	10000	**618,691**	**1.1**	**1.2**	**361**	**354**
1000	10000	**622,312**	**1.2**	**1.2**	**376**	**357**

(CN2) Strom-2306, Workers=1, ackers=1, -c topology.max.spout.pending=500, 1000, 5000, 10000, -c topology.max.spout.pending=500, 1000, 5000, 10000

producer.batch.sz	max.spout.pending	Throughput (msgs/sec)	Avg latency (ms)	Avg of max latency	Avg CPU Node 1	Avg Mem Node1
1	500	905,108	**0.039**	0.041	242	1417
100	500	1,182,062	**0.096**	0.107	239	1404
500	500					
1000	500	**137,479**	0.677	0.704	76	1411
1	1000	913,191	**0.039**	0.051	238	1412

100	1000	**114,8091**	**0.097**	0.098	246	1378
500	1000	**352,154**	0.568	0.610	113	1433
1000	1000	**255,012**	0.739	0.847	101	1390
1	5000	895,274	**0.062**	0.091	241	1377
100	5000	**1,241,895**	0.103	0.140	253	1396
500	5000	**1,204,216**	0.424	0.454	241	1333
1000	5000	**1,167,574**	0.717	0.755	242	1506
1	10000	990,358	**0.043**	0.150	242	1327
100	10000	**1,190,928**	0.113	0.171	249	1457
500	10000	**1,196,087**	0.432	0.494	243	1509
1000	10000	**1,135,933**	0.759	0.780	245	1542
1000	null					

Observations (ACK mode):

- Throughput was consistently better (~2x) for 2306. Some modes gave poor throughput, interestingly, often times it is the same mode that give bad perf for both.
- Latency was again consistently better for 2306, always remained in microseconds territory.
- CPU usage was consistently lower for 2036 if master's bad modes were to be ignored.
- 2306's CPU was generally lower even though higher throughput was being achieved.
- Memory consumption was consistently higher for 2306 (attributable to the higher throughput).

Multi-worker

(CN3) Master (default batching=100)
-c topology.max.spout.pending=1000

Wor	Spout	Bolt	Acker	Throughput	Avg Late	Max laten	Avg CPU	Avg CPU	Avg CPU	Avg CPU	Avg Mem	Avg Mem	Avg Mem	Avg Mem

kers	Count	Count	Count		ncy	cy	Node 1	Node 2	Node 3	Node 4	Node 1	Node 2	Node3	Node 4
				Single Worker										
1	4	4	0	7,082,507					792				1396	
1	4	4	4	902,983	1.4	1.4			930				1645	
				Multi Worker										
4	4	4	0	8,810,170			193	192	193	197	1249	899	1224	1174
4	4	4	4	675,299	1.4	1.4	281	281	278	281	1490	1446	1464	1449
4	12	12	0	18,885,053			535	534	545	491	1227	1320	1199	1195
4	12	12	12	1,709,808	1.4	1.4	788	797	788	772	1984	2028	1929	2016
4	18	18	0	26,976,837			927	733	970	720	1460	1268	1478	1328
4	18	18	18	953,279	1.3	1.3	939	904	797	797	1738	1796	1832	1832
4	24	24	0	36,452,232			1170	1200	1209	1236	2479	1560	1679	1641
4	24	24	24	2,518,283	1.4	1.4	1015	1050	1077	1024	1863	1741	1768	1801
4	36	36	0	43,396,907			1627	1594	1603	1603	1624	1653	1661	1585
4	36	36	36	2,870,574	1.4	1.4	1102	1111	1151	1151	1649	1669	1678	1678
4	48	48	0	45,113,529			1865	1852	1882	1882	1706	1688	1869	1869
4	48	48	48	2,655,899	1.3	1.3	1187	1198	1215	1215	1645	1688	1684	1684

(CN3) STORM-2306 -c topology.max.spout.pending=1000
-c nullbolt.sleep.micros=0 -c topology.producer.batch.size=1(default)

Workers	Spout count	Bolt count	Acker count	Avg no. of tuples per sec	Avg Latency (ms)	Avg max latency (ms)	Avg CPU Node 1	Avg CPU Node 2	Avg CPU Node 3	Avg CPU Node 4	Avg Mem Node 1	Avg Mem Node 2	Avg Mem Node3	Avg Mem Node 4
				Single Worker										
1	4	4	0	9,087,583				642				1296		

1	4	4	4	1,914,816	0.033	0.036		996				1527		
				Multi Worker										
4	4	4	0	19,932,633			140	136	132	130	1350	1347	1340	1353
4	4	4	4	1,600,878	0.029	0.030	252	261		241	1615	1587	1644	1612
4	12	12	0	27,547,428			451	458	410	430	1396	1330	1351	1358
4	12	12	12	1,909,662	0.024	0.025	379	378	361	365	1219	1176	1233	1181
4	18	18	0	34,201,178			613	732	617	735	1321	1443	1338	1346
4	18	18	18	2,036,712	0.034	0.043		452	426	437		1084	1225	1275
4	24	24	0	44,793,720			908	949	926	932	1316	1309	1328	1412
4	24	24	24	2,079,458	0.032	0.035	496	522	509	542	1199	1149	1258	1520
4	36	36	0	60,203,570			1396	1325	1366	1400	1502	1386	1504	1464
4	36	36	36	1,893,616	0.085	0.075	666	698	675	675	1210	1202	1245	1231
4	48	48	0	65,482,699			1635	1642	1631	1629	1573	1574	1547	1541
4	48	48	48	1,943,957	0.085	0.101	820	819	794	845	1347	1284	1374	1348

(CN4) STORM-2306 -c topology.max.spout.pending=1000
-c topology.producer.batch.size=100

Workers	**Spout count**	**Bolt count**	**Acker count**	**Avg no. of tuples per sec**	**Avg Latency**	**Avg max latency**	**Avg CPU Node 1**	**Avg CPU Node 2**	**Avg CPU Node 3**	**Avg CPU Node 4**	**Avg Mem Node 1**	**Avg Mem Node 2**	**Avg Mem Node3**	**Avg Mem Node 4**
				Single Worker										
1	4	4	0	11,848,266										
1	4	4	4	1,763,741	0.568	0.580		648				1554		
				Multi Worker										
4	4	4	0	19,717,287			138	131	131	132	1372	1318	1347	1346
4	4	4	4	1,292,162	0.240	0.241	208	222	206	194	1462	1554	1667	1453

4	12	12	0	47,091,537			444	469	432	409	1444	1434	1519	1423
4	12	12	12	1,969,099	0.702	0.706	359	370	362	364	1314	1572	1360	1338
4	18	18	0	58,841,308			623	746	613	681	1548	1485	1517	1538
4	18	18	18	2,003,754	0.883	0.895	417	458	424	436	1245	1227	1290	1265
4	24	24	0	63,601,387			870	890	873	904	1538	1493	1579	1566
4	24	24	24	2,111,691	0.931	1.178	502	508	485	NA	1307	1214	1352	NA
4	36	36	0	69,629,983			1239	1281	1354	1393	1501	1597	1579	1595
4	36	36	36	1,795,028	0.979	1.016	668	654	627	650	1387	1354	1380	1516
4	48	48	0	85,880,495			1627	1739	1696	1751	1623	1665	1639	1724
4	48	48	48	1,997,670	1.018	1.030	793	1442	765	784	1593	1442	1465	1528

Observations:

- Throughput was generally better for 2306 in all multi-worker modes other than some of cases involving ACK mode. At lower parallelism and ACKing enabled, 2306 was a bit better, But at higher parallelism, with ACKing enabled & batchSz=1, throughput was bit worse for 2306. This needs to be examined.
- Latency was again consistently better for 2306, often touching low 2 digit microsecs with batching disabled.(It is known to be possible to reduce latency further by tweaking sleep strategy and trading off some CPU)
- 2306's CPU was generally lower even though higher throughput was being achieved.
- Memory consumption was similar with batching enabled. 2306's mem consumption was reduced in batching disabled mode (default).

3. ConstSpout->IDBolt->NullBolt Topo

Spout emits a constant stream of tuples (10 bytes each) that the IDBolt clones and the NullBolt simply accepts and discards. This stresses both the Spout -> Bolt as well as Bolt->Bolt communication paths.

Single Worker

(CIN1) Master, Worker=1, acker=0,

disruptor.batc	**Avg no. of tuples per**	**Avg CPU**	**Avg Mem**

h.size	sec	Node 1	Node1
1	206,220	1442	3898
100	**73,370**	1540	4345
500	**708,666**	1203	4273
1000	275,233	1484	4279

(CIN1) Sorm-2306, Worker=1, acker=0

producer.batch.size	Avg no. of tuples per sec	Avg CPU Node 1	Avg Mem Node1
1	3,850,895		
100	3,873,437		
500	3,124,899		
1000	3,942,495		

Observations (no ACK):

- Master's throughput is much lower than what it achieved for ConstSpout->NullBolt (Down from ~2.5 mill)
- 2306 throughput is down (but not as much) from 5.1 mill in C->N. Several times faster than master.
- Batch size not impacting throughput very much for 2306.

(CIN2) Master, Workers=1, Ackers=1, -c topology.max.spout.pending=500, 1000, 5000, 10000,

batch.size	max.spout.pending	Avg no. of tuples per sec	Avg latency	Max latency	Avg CPU Node 1	Avg Mem Node1
1	500	250,608	**0.4**	**0.5**	625	1343
100	500	**83,303**	2.8	2.8	141	577
500	500	NA	NA	NA	NA	NA

1000	500	**61,270**	4.0	4.1	110	587
1	1000	255,112	**0.8**	**1.1**	645	1182
100	1000	181,328	2.3	2.4	463	1658
500	1000	**124,870**	3.8	3.9	183	719
1000	1000	**122,566**	3.8	3.9	180	711
1	5000	268,903	**12.1**	**12.3**	660	1556
100	5000	**578,537**	2.9	4.1		
500	5000	**576,253**	3.5	4.0	463	1686
1000	5000	**590,837**	3.1	3.7	465	1708
1	10000	251,816	**30.8**	**32.1**	648	1632
100	10000	**509,337**	**9.8**	**11.6**		
500	10000	**557,616**	4.7	5.0	443	1614
1000	10000	**563,562**	4.5	4.7	452	1620

(CIN2) Sorm-2306, Workers=1, ackers=1, -c topology.max.spout.pending=500, 1000, 5000, 10000,

batch. size	max.spout.pending	Throughput (msgs/sec)	Avg latency	Max latency	Avg CPU Node 1	Avg Mem Node1
1	500	671,478	**0.131**	**0.159**		
100	500	351,207	0.419	0.474		
500	500					
1000	500	**108,378**	1.781	1.834		
1	1000	683,895	**0.143**	**0.159**		
100	1000	792,253	**0.352**	**0.452**		
500	1000	**262,433**	1.493	1.505		
1000	1000	**203,774**	1.740	1.873		
1	5000	707,008	**0.358**	**0.788**		

100	5000	**951,845**	0.546	1.019		
500	5000	**826,062**	1.868	2.153		
1000	5000	**955,195**	1.992	2.235		
1	10000	739,283	1.072	6.533		
100	10000	798,845	0.525	1.381		
500	10000	**891,224**	1.209	1.264		
1000	10000	**813,607**	2.010	3.665		

Observations (with ACK):

- No extreme fluctuation in latencies in any mode. Higher throughputs.
- Interestingly, the modes having poor throughput in 2306 are bad in 2306 as well.

Multi Worker

(CIN3) Master
-c topology.max.spout.pending=1000, **-c topology.disruptor.batch.size=1**

Workers	**Spout count**	**Bolt count**	**Acker Count**	**Throughput (msgs/sec)**	**Avg latency**	**Max latency**	**Avg CPU Node 1**	**Avg CPU Node 2**	**Avg CPU Node 3**	**Avg CPU Node 4**	**Avg Mem Node 1**	**Avg Mem Node 2**	**Avg Mem Node3**	**Avg Mem Node 4**
				Single Worker										
1	4	4	**0**	**24,958**					1774				4381	
1	4	4	4	634,333	2.6	2.7			1820				1605	
				Multi Worker										
4	4	4	**0**	**-298,234**			1523	1357	1506		4034	4056	3738	
4	4	4	4	676,112	0.5	0.36	732	729	732		1669	1673	1665	
4	12	12	**0**	**-38,938**			1759	1750	1769		4381	4327	4374	
4	12	12	12	1,747,295	0.3	0.4	1526	1486	1551		1745	1701	1729	
4	18	18	**0**	**-396,434**			1805	1772	1776		4247	4384	4478	

4	18	18	18	1,169,674	0.6	0.8	1459	1532	1469		1725	1677	1705	
4	24	24	0	7,859,716			1797	1918	1947		4432	4199	3437	
4	24	24	24	1,113,616	0.9	1.1	1518	1551	1651		1784	1746	1704	
4	36	36	0	18,557,608			2107	2096	2101		2515	2543	3276	
4	36	36	36	1,444,495	1.9	1.9	1779	1691	1797		1762	1746	1738	
4	48	48	0	18,934,333			2215	2219	2205		2724	2526	2537	
4	48	48	48	1,284,474	4.1	4.3	1928	1953	1905		1759	1725	1742	

(CIN3) Storm-2306 -c topology.max.spout.pending=1000
-c topology.producer.batch.size=1

Workers	**Spout Count**	**Bolt Count**	**Acker Count**	**Avg no. of tuples per sec**	**Avg Latency**	**Avg max latency**	**Avg CPU Node 1**	**Avg CPU Node 2**	**Avg CPU Node 3**	**Avg CPU Node 4**	**Avg Mem Node 1**	**Avg Mem Node 2**	**Avg Mem Node3**	**Avg Mem Node 4**
				Single Worker										
1	4	4	0	5,077,928										
1	4	4	4	1,361,112	0.178	0.252								
				Multi Worker										
4	4	4	0	15,552,045			320		323		1531	1532		
4	4	4	4	1,138,791	0.081	0.086	271		242		1388	1530		
4	12	12	0	17,105,587			670		693		1668	1713		
4	12	12	12	1,447,929	0.079	0.086	447		443		1353	1225		
4	18	18	0	21,624,924			920		1054		1728	1734		
4	18	18	18	1,472,258	0.111	0.119	572		579		1193	1247		
4	24	24	0	25,839,337			1350		1272		1758	1960		
4	24	24	24	1,444,224	0.144	0.174	694		658		1289	1408		
4	36	36	0	33,953,691			1747		1763		2031	2092		
4	36	36	36	1,436,170	0.255	0.320	851		950		1360	1411		

4	48	48	0	37,304,328			2045		2024		1723	1756		
4	48	48	48	1,395,587	0.315	0.332	982		979		1566	1473		

Observations (No batching):

- batch.size=1 on master resulted in many bad runs if ACK=0. No bad runs on 2306.
- Much better latencies on 2306.
- 2x throughput over master on similar settings with ACK disabled.
- Consistently much lower CPU
- Consistently lower Mem usage
- Had some errors collecting CPU and Mem on two of the nodes.

(CIN4) Master

-c topology.max.spout.pending=1000, **-c topology.disruptor.batch.size=100**

Workers	Spout Count	Bolt Count	Acker Count	Avg no. of tuples per sec	Avg Latency	Avg max Latency	Avg CPU Node 1	Avg CPU Node 2	Avg CPU Node 3	Avg CPU Node 4	Avg Mem Node 1	Avg Mem Node 2	Avg Mem Node3	Avg Mem Node 4
				Single Worker										
1	4	4	0	3,679,308										
1	4	4	4	662,966	2.7	2.7								
				Multi Worker										
4	4	4	0	2,427,766			322	655	1598		1996	4363	3519	
4	4	4	4	561,345	2.7	2.7	346	346	348		1585	1451	1588	
4	12	12	0	**806,558**			1735	1758	1324		4383	4116	4393	
4	12	12	12	1,419,320	2.7	2.7	911	976	955		1947	1989	1919	
4	18	18	0	**-598,792**			1738	1649	1659		4390	3892	4332	
4	18	18	18	1,770,828	2.9	2.9	1063	1133	1075		1774	1871	1915	
4	24	24	0	**6,269,849**			1778	1856	1768		4385	4394	1936	
4	24	24	24	1,877,528	2.7	2.8	1187	1166	1050		1739	1755	1724	
4	36	36	0	5,907,237			1846	1839	1838		4386	4359	4349	

4	36	36	36	2,030,891	2.6	2.6	1289	1289	1302		1637	1664	1664	
4	48	48	0	**12,167,095**			2008	1995	2031		4238	4219	4221	
4	48	48	48	1,881,424	2.5	2.6	1343	1335	1285		1710	1729	1777	

(CIN4) Storm-2306 -c topology.max.spout.pending=1000
-c topology.producer.batch.size=100

Workers	**Spout count**	**Bolt count**	**Acker count**	**Thru-put (msgs/sec)**	**Avg Latency (ms)**	**Max Latency**	**Avg CPU Node 1**	**Avg CPU Node 2**	**Avg CPU Node 3**	**Avg CPU Node 4**	**Avg Mem Node 1**	**Avg Mem Node 2**	**Avg Mem Node3**	**Avg Mem Node 4**
				Single Worker										
1	4	4	0	8,370,245										
1	4	4	4	1,085,320	1.590	1.601								
				Multi Worker										
4	4	4	0	15,616,445			331		330		1539	1540		
4	4	4	4	1,029,157	0.449	0.515	244		228		1337	1443		
4	12	12	0	26,654,512			713		776		1833	1720		
4	12	12	12	1,531,357	1.239	1.257	427		415		1272	1335		
4	18	18	0	30,318,516			934		1083		1790	1763		
4	18	18	18	1,086,954	1.543	1.554	520		529		1410	1400		
4	24	24	0	36,160,187			1260		1225		1936	2001		
4	24	24	24	1,550,812	1.612	1.651	679		640		1487	1366		
4	36	36	0	42,446,358			1732		1667		2036	2079		
4	36	36	36	1,453,641	1.696	1.705	801		885		1617	1485		
4	48	48	0	**44,643,937**			2025		2095		1961	1923		
4	48	48	48	1,498,841	1.835	1.983	1002		1036		1610	1572		

Observations (with batching):

- On master, throughput surprisingly worsened in some cases with increase in spout & bolt parallelism (when ACK is enabled), but then regained its composure at higher parallelism.
- 2306 had solid and steady throughput. What master achieved with 48 spouts and 48 bolts (no ack), 2306 was able to achieve with just 4 spouts and 4 bolts. At 48 parallelism, 2306 achieved 4x of master.
- With ACKing enabled, the throughput plateaus early on… so increasing parallelism doesnt help for both 2306 and master. [ACK path is a known bottleneck that is not tackled in 2306]
- Consistent latencies on 2306
- Had trouble collecting CPU and mem on some of the nodes. But where available, they are similar or better on 2306, even though it is achieving higher throughput.

4. TridentConstSpout -> Each

Trident version of the ConsSpoutNullBolt topo. Uses .Each() instead of the NullBolt.

Link to topology (authored by Satish Duggana):
https://github.com/satishd/storm/commit/416c5df8abe9a15f767df53eef77461fb872b695#diff-0288cc4a70140db7a646049844af6e05

Single and Multi Worker

(TCS1) Master
-c topology.trident.batch.emit.interval.millis=1, kafka spout fetch size = 8192,

Here, MSP = topology.max.spout.pending

Workers	batch.size	MSP	Parallelism	Throughput (batches/sec)	Avg Latency	Max latency	CPU Node 1	CPU Node 2	Mem node 1	Mem node 2
Single Worker										
1	1	5	1	**78**	14.4	14.4	72		352	
1	1	5	8	158	18.6	19.6	584		1086	

1	1	5	16	**891**	21.1	43.0	1103		1974	
1	1	20	1	120	16.0	28.0	75		383	
1	1	20	8	228	18.3	19.0	545		1045	
1	1	20	16	**520**	19.1	21.5	1102		2002	
1	500	5	1	107	14.9	25.3	73		358	
1	500	5	8	453	15.1	31.0	565		1052	
1	500	5	16	374	17.4	36.5	1010		1789	
1	500	20	1	112	15.8	17.5	72		380	
1	500	20	8	187	18.3	38.0	596		1237	
1	500	20	16	308	**69.4**	**445.5**	892		2654	
Multi Worker										
4	1	5	1	103	16.6	16.6	15	68	247	289
4	1	5	8	366	21.1	21.1	133	126	436	410
4	1	5	16	533	18.8	44.0	257	240	593	599
4	1	20	1	**53**	20.9	22.8	14	62	521	287
4	1	20	8	416	18.4	20.0	129	121	422	398
4	1	20	16	**653**	20.3	24.3	280	275	635	644
4	500	5	1	78	17.2	18.0	15	66	249	299
4	500	5	8	249	18.1	18.7	134	126	418	407
4	500	5	16	670	22.5	24.4	279	248	624	584
4	500	20	1	91	23.0	25.0	14	63	247	298
4	500	20	8	249	19.9	20.8	142	132	403	405
4	500	20	16	737	19.21	20.8	266	268	620	635

(TCS1) Storm-2306
-c topology.trident.batch.emit.interval.millis=1

Workers	batch.sz	MSP	Parallelism	Throughput (batches/sec)	Avg latency	Max latency				
Single Worker										
1	1	5	1	**83**	7.5	14.4				
1	1	5	8	378	9.2	9.2				
1	1	5	16	478	10.7	20				
1	1	20	1	108	7.8	15				
1	1	20	8	553	9.2	10.8				
1	1	20	16	**670**	10.9	21.7				
1	500	5	1	**78**	20.3	36				
1	500	5	8	254	21.3	30				
1	500	5	16	362	35.1	55				
1	500	20	1	**83**	15	15.1				
1	500	20	8	274	26.2	82				
1	500	20	16	374	33.3	34.5				
Multi Worker										
4	1	5	1	**49**	-	-				
4	1	5	8	362	16.688	25.5				
4	1	5	16	578	14.813	33				
4	1	20	1	**45**	-	-				
4	1	20	8	212	14.389	29.5				
4	1	20	16	**787**	16.22	35				
4	500	5	1	**0**	**NAN**	0.0				
4	500	5	8	**0**	**NAN**	0.0				
4	500	5	16	**0**	**NAN**	0.0				
4	500	20	1	**0**	**NAN**	0.0				

4	500	20	8	**0**	**NAN**	0.0				
4	500	20	16	**0**	**NAN**	0.0				

Observations:

- **Single Worker:** 2306 prefers batching disabled (default) for Trident and in that mode it has better latency and throughput than master. With batch=500, master is often better than 2306. Enabling batching did not necessarily improve perf for either 2306 or master.
- **Multi Worker:** Master often performed better than 2306 when batching is disabled. 2306 breaks when batching is enabled. This needs investigation.
- 2306 run had issues collecting cpu/mem

5. TridentKafkaSpout ->Each

Trident topology. Uses KafkaSpout instead of TridentConstSpout. Uses ForEach instead of the NullBolt. Here the kafka partitions=3 and spout.parallelism= 3

Link to topology (authored by Satish Duggana):
https://github.com/satishd/storm/commit/416c5df8abe9a15f767df53eef77461fb872b695#diff-0d07945a1e7a49f1f9cdd30b0bcbf16c

Single Worker

(TKSN1) Master
-c topology.trident.batch.emit.interval.millis=1

batch	MSP	Fetch Size	Transfers (msgs/sec)	Avg latency	CPU node1	CPU node 1	CPU node 1	Mem node 1	Mem node 2	Mem node 3
1	5	8								
1	**5**	**32**	839,926	212.5	366	287	300	789	784	659
1	20	8								
1	**20**	**32**	782,601	459.0	307	275	**1298**	815	796	518
500	5	8								

500	5	32								
500	20	8								
500	20	32								

(TKSN1) Storm 2306
topology.trident.batch.emit.interval.millis=1

batch	MSP	Fetch Size	Transfers (msgs/sec)	Avg latency	Max latency	CPU node1	CPU node1	CPU node1	Mem node 1	Mem node 2	Mem node 3
1	5	8									
1	**5**	**32**	889,146			340	328	300	796	813	659
1	20	8									
1	**20**	**32**	860,537			374	334	274	805	799	643
500	5	8	**0**	**NAN**							
500	5	32	**0**	**NAN**							
500	20	8	**0**	**NAN**							
500	20	32	**0**	**NAN**							

Observations:

- Bit Better throughput. UI had problems in measuring latency and throughput in terms of batches/sec. So we recorded the transfer/sec on the spout.
- Again, 2306 had an Issue in runs with Batching enabled. Needs fix.

Summary: [updated Dec 21]

A few high level observations.

- **Latency:** appears to be a notable strength of 2306 and in many cases it is significantly lower. It touches low 2 digit microseconds in both single and multi-worker modes in some runs.
- **Throughput:**

- **Single Worker:** The improvement is significant for communication within a single worker (the primary focus of 2306), esp with ACKing disabled. Even with ACKing enabled numbers for single worker show noticeable improvement in majority of the cases.
- **Multi-worker** mode combined with ACKing enabled is generally a significant perf hit for both master and 2306, and the differences between master and 2306 are narrower and topology dependent. This is expected as the two modes are known bottlenecks and not tackled in 2306. 2306 numbers shows degradation compared to master in Trident topos with batching enabled. [needs investigation]
- **Batching:** 2306 runs were taken with batching disabled(default) and master had batchSz=100 (default). Based on other runs (not recorded here) Master tends to deliver better throughput and latency with its default batch size compared to batching disabled.

BackPressure: If BP is disabled, running topos with ACKing disabled on master (or with max.spout.pending=null in ACK mode) is not recommended unless topo is known to always operate at low throughputs. We can see in the TVL3 runs that at 1mill rate, master's throughput quickly degraded to just 12k. With BP enabled on master, the drop in throughput was more rapid and fluctuated within the range of 725k/s and 0k/s... as the BP engaged and disengaged. On the other hand, 2306 (with its integrated BP) was able to deliver consistent throughput which fluctuated within the range 623k/s and 916k/s over the same duration.

- **CPU and Memory:** At similar throughputs, 2306 has lower CPU. ...todo mem..
- **Trident:** Single worker mode worked well but overall Improvements are not significant. 2306 breaks when batching is enabled in multi-worker mode. 2306 works better with batching disabled even in single worker mode.
- **GC settings:** Choice of GC can have a big impact on numbers. The runs here used default GC for all topos other than TVL. TVL internally sets its GC to '-XX:+UseParNewGC -XX:+UseConcMarkSweepGC'. For 2306, other experiments indicate that G1 gc (-XX:+UseG1GC) generally gives much better performance on multi socket (Linux) machines as compared to default GC. On single socket machine (MacBook Pro, mid 2015), G1 GC made the performance worse than default GC.

Observed Issues:

- Trident with batching enabled is not working for 2306
- In multiworker mode with ACKing enabled (like TVL topo), master was able to do better than 2306 in a few cases. There may be a regression in some cases when multiworker +ack mode with 2306.
- Improving multi worker and ACK modes are the next logical steps for Storm.