Spark SQL 2.0/2.1
Experiences using TPC-DS

Berni Schiefer
IBM, Spark Technology Center

SPARK SUMMIT
EUROPE 2016

Motivation

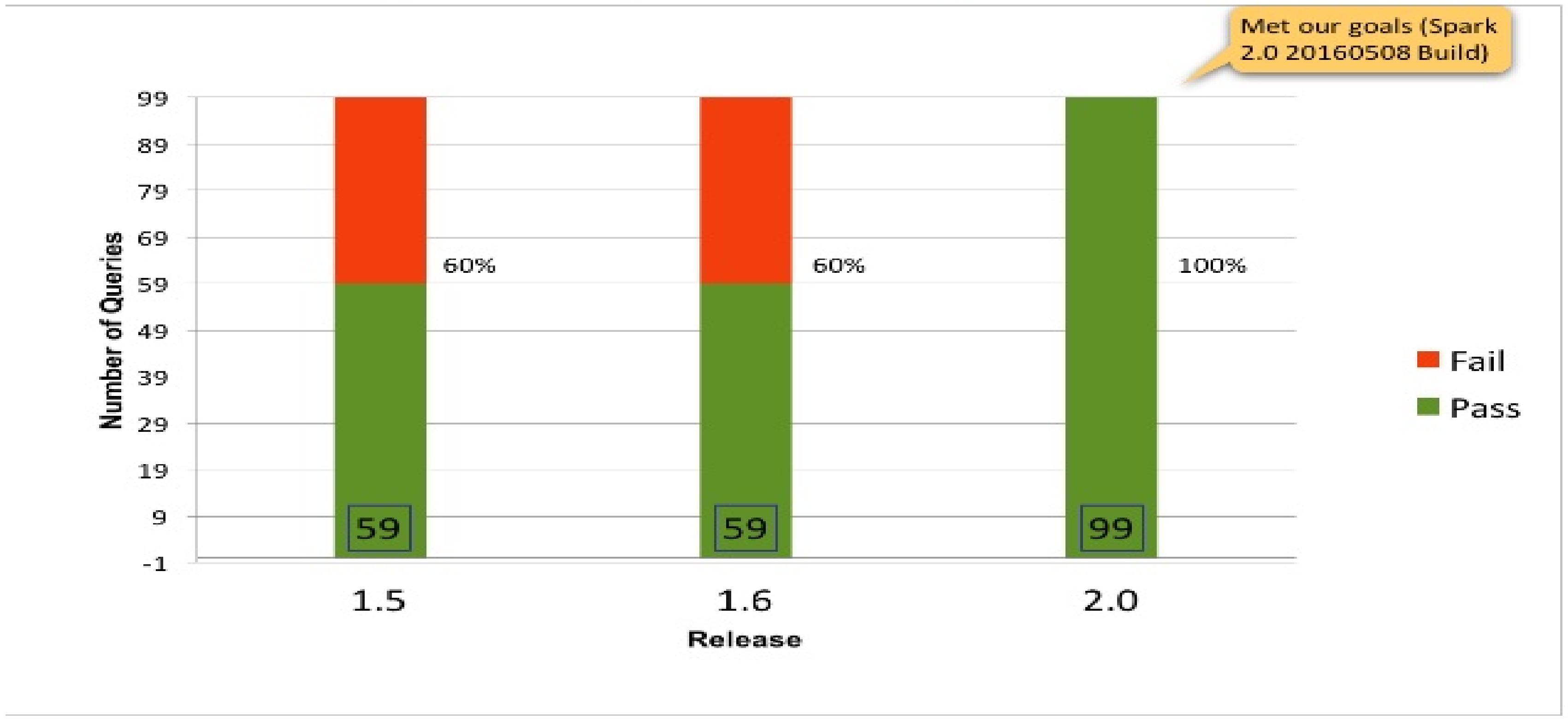
- Spark SQL is very popular and rapidly evolving
- Today it is still deployed mostly on ”traditional” Hadoop clusters
- What if we dared to “re-imagine” a different kind of cluster, one designed for Spark?
- Let’s try this out, using the latest version of Spark and a popular yet challenging industry-standard workload at a large enterprise scale

About the TPC-DS Benchmark

- Developed between 2004 and 2011 (Version 1) and enhanced in 2015 (Version 2)
  - But no vendor has ever formally published a result
  - Now at Version 2.3 (http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-ds_v2.3.0.pdf)
- Models a multi-domain data warehouse environment for a hypothetical retailer
  - Retail sales, web, catalog data, inventory, demographics & promotions
- Models several aspects of business operations
  - Queries, concurrency, data loading, data maintenance (we skip that)
  - Designed for relational data warehouse product offerings & adjusted for Hadoop
- Four broad types of queries in the set of 99
  - Reporting queries, Ad-hoc queries, Iterative OLAP queries, Data mining queries
- Designed for multiple scale factors
  - 100GB, 300GB, 1TB, 3TB, 10TB, 30TB and **100TB**
- Designed for multi-user concurrency
  - Minimum of 4 concurrent users running all 99 queries
- ***Has become the de-facto benchmark for SQL over Hadoop engines***

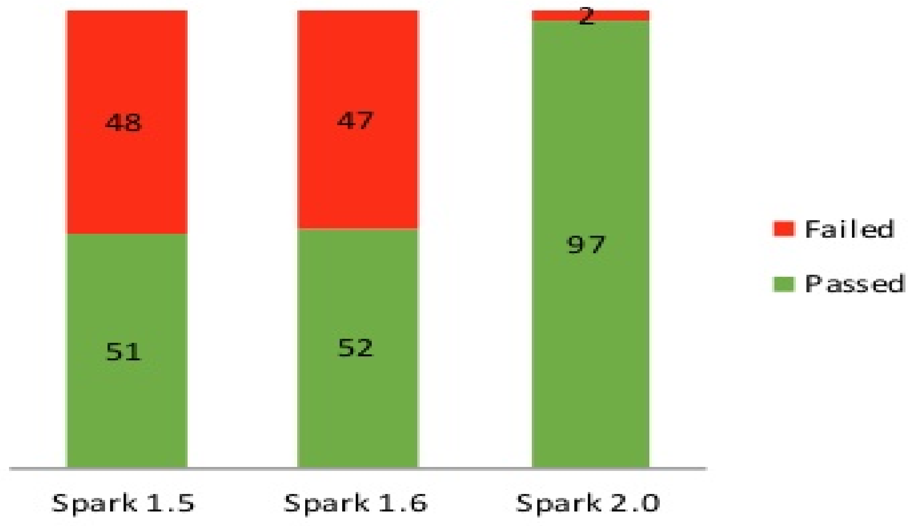
Note: Because it isn't feasible to run every element of the benchmark and audit it ,we call it Hadoop-DS

TPC-DS Query Compliance @1GB

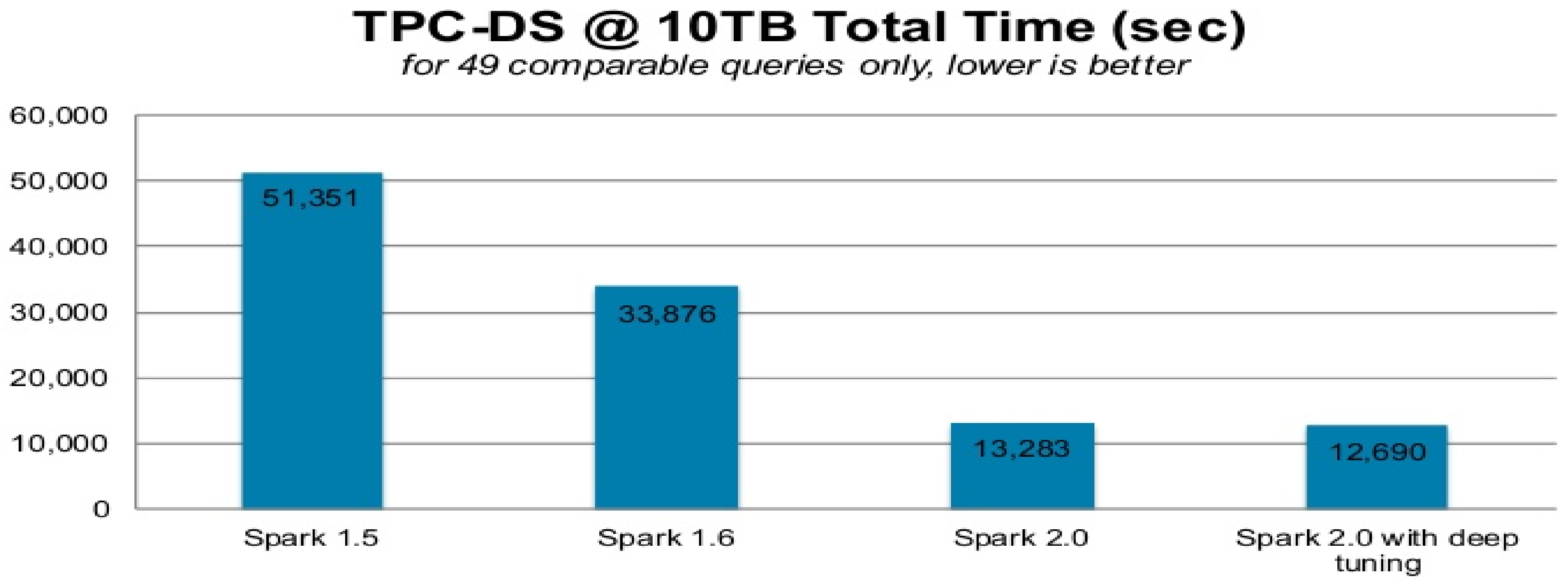


SPARK SUMMIT EUROPE 2016

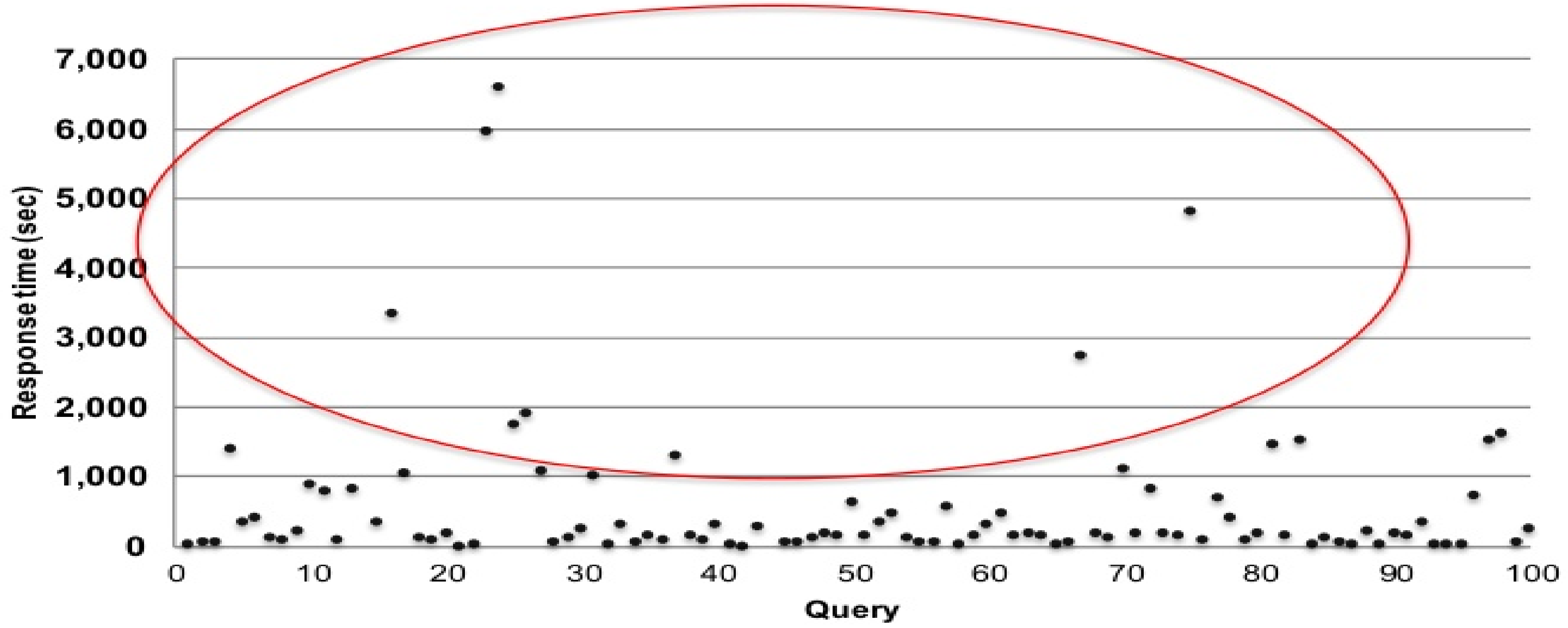
TPC-DS Query Compliance @10TB



TPC-DS Single User



TPC-DS Elapsed Time @10TB



Next Steps Post Spark 2.0

- More data
  - Go from 10TB to 100TB of raw data generated by TPC-DS
- More users
  - Go from single-user tests to 4 concurrent users and beyond
- More performance
  - Focus on long-running queries.
  - Develop and submit changes for 2.x (Spark Technology Center)

IBM Technology Partners to Evaluate "Spark SQL @ Scale"

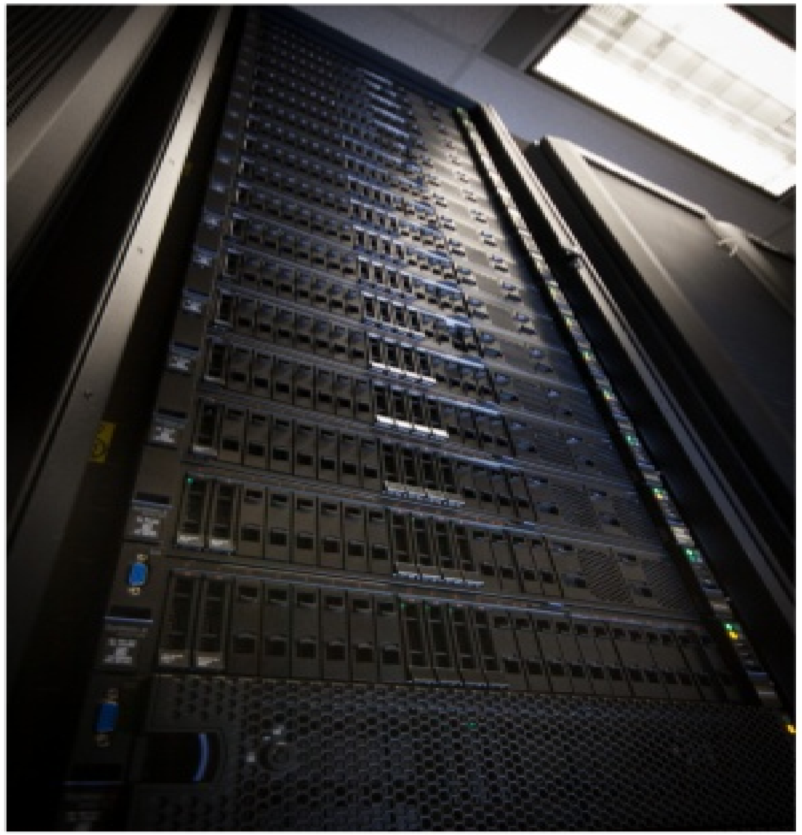
intel

Lenovo

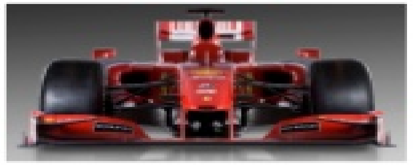
Mellanox
TECHNOLOGIES

The “F1” Spark SQL Cluster

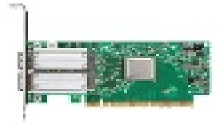
- A 30-node cluster
  - 2 management nodes
    - Lenovo x3550 M5 / x3650 M5
  - 28 data nodes
    - Lenovo x3650 M5
      - 2 sockets (18cores/socket)
      - 1.5TB RAM
      - 8x2TB SSD
  - 2 racks,
    - 20x 2U servers per rack (42U racks)
  - 1 switch, 100GbE, 32 ports, 1U
    - Mellanox SN2700



“F1” Spark SQL Platform Details



- Each data node
  - CPU: 2x E5-2697 v4 @ 2.3GHz (Broardwell) (18c)
  - Memory: 1.5TB per server 24x 64GB DDR4 2400MHz
  - Flash Storage: 8x 2TB SSD PCIe NVMe (Intel DC P3600), 16TB per server
  - Network: 100GbE adapter (Mellanox ConnectX-4 EN)
  - IO Bandwidth per server: 16GB/s, Network bandwidth 12.5GB/s
  - IO Bandwidth per cluster: 480GB/s, Network bandwidth 375GB/s
- Totals per Cluster (counting Data nodes)
  - 1,008 cores, 2,016 threads (hyperthreading)
  - 42TB memory
  - 448TB raw, 224 NVMe



SPARK SUMMIT EUROPE 2016

Configuring the OS

- Use a current OS (e.g. RHEL 7.2)
- We applied standard IBM Open Platform (IOP) guidance
  - Tune memory segment management
  - Increase process limit
  - Disable transparent huge pages
  - Tune the filesystem (we used XFS under HDFS)
    - Validate I/O throughput (we use “TestIO” - see http://tate.cx)

Configuring the Network

- Again, we applied our ”best practice” recipes
- Even though we had a 100GbE Mellanox switch there was ZERO unique tuning!
- Per-port and per-server tuning
- Lengthen timeouts & grow buffers – long/big queries
- Enable more retries and better keepalive
- Validate network performance
  - iperf -c mer03x.svl.ibm.com -l 1024k -w 128k -P 4 -i 3 -t 30

Configuring Spark

- Definitely a “work in progress”

spark.yarn.executor.memoryOverhead 10240 (default 384)
spark.driver.maxResultSize 20g (default 1g)
spark.local.dir /data1/tmp,/data2/tmp,/data3/tmp,/data4/tmp,/data5/tmp,/data6/tmp,/data7/tmp,/data8/tmp (default /tmp)
spark.sql.crossJoin.enabled true
spark.network.timeout 36000 (default 120s)
spark.sql.broadcastTimeout 36000 (default 300s)
spark.buffer.pageSize 64m (default is computed but does not always work)
spark.shuffle.file.buffer 512k (default 32k)
spark.memory.fraction 0.8 (default 0.6)
spark.sql.starJoinOptimization true
spark.scheduler.listenerbus.eventqueue.size 130000 (default 10000)
…

I’ve learned about some new parameters to try at this Spark Summit!

Configuring Spark
The Magic Parameters: executor-memory

- Remember: 28nodes, 1.5TB/node = 42TB RAM
- Spark SQL needs ”enough memory”, but even more doesn’t help uniformly
  - For TPC-DS it seems 24GB is ”enough”
    - More helps some queries, but leads others astray
- But wait! 24x1008=24TB – am I not wasting memory?
  - Don’t forget the ”overheads”
    - ~12% for “Hadoop”/Linux and ~50% per executor (24x1.5=36)
  - 42TB – 5TB = 37TB / 36GB/core = ~1000
- A conservative strategy. Wanted to avoid failures and there was extremely limited time to run multiple experiments

Configuring Spark
The Magic Parameters: num-executors

- Remember: 28nodes, 36 cores/node = 1008 cores
  - Hyper-threading gives you 2016 hardware threads
- Spark SQL needs ”enough executors”, or you won’t get the parallelism you want
  - If you run multiple concurrent streams you have to ”share”
    - E.g. with 4 streams you get 1008/4 = 252 executors
- Set executor-cores = 2
  - Seems logical for number of threads
  - Matches wisdom of the internet advice - “1 < executor-cores < 5”

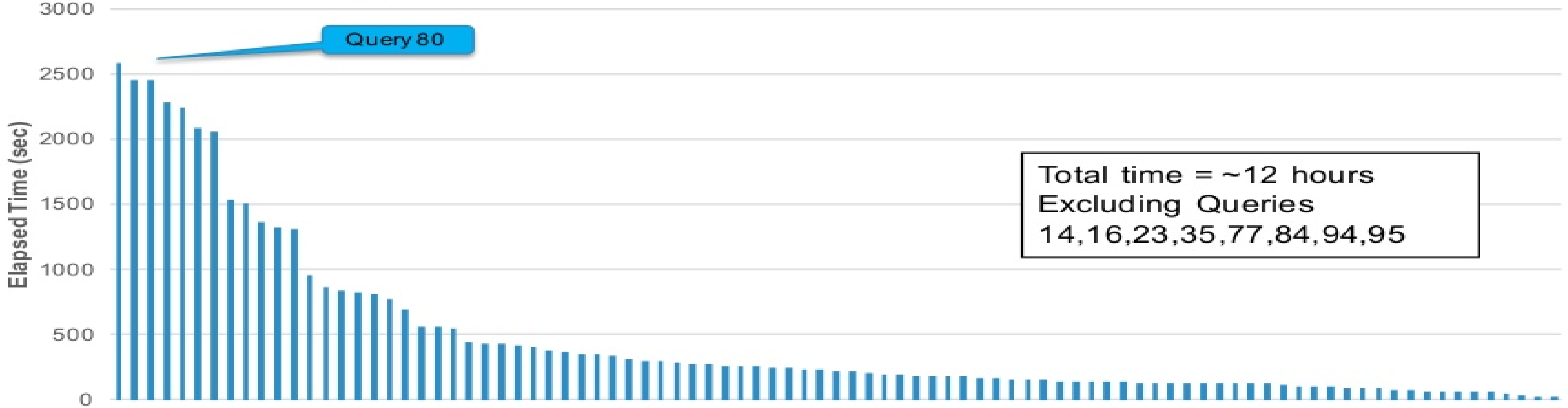
Configuring Spark Invocation

- spark-sql command line parameters

```
--driver-memory 20g
--verbose
--master yarn
--executor-memory 24g
--num-executors 1008
--executor-cores 2
```

- another parameter that we tried (but rejected due to side-effects)
  - spark.sql.shuffle.partitions=2001 (we also tried 2000)
    - Default = 200

TPC-DS Power Run Elapsed Time Distribution

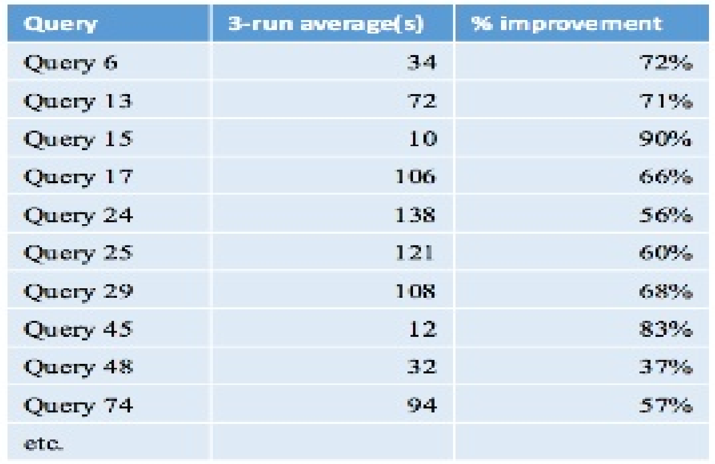


Total time = ~12 hours
Excluding Queries
14,16,23,35,77,84,94,95

SPARK-17791

Join Reordering Using Star Schema Detection

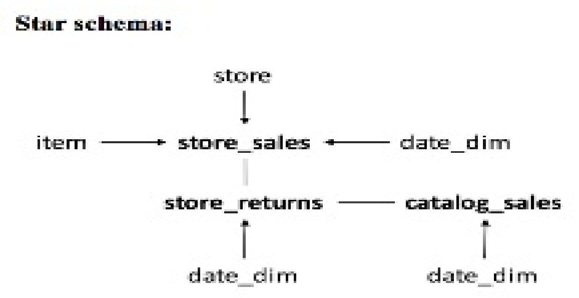
- A "Star schema" consists of one or more fact tables referencing a number of dimension tables.
- Queries against a star schema are expected to run fast because of the established RI constraints among the tables.
- This work proposes a join reordering based on natural, generally accepted heuristics for star schema queries:
  - Finds the star join with the largest fact table and places it on the driving arm of the left-deep join.
  - This plan avoids large tables on the inner, and thus favors hash joins.
  - Applies the most selective dimensions early in the plan to reduce the amount of data flow.
- Our results using a 1TB TPC-DS workload running on a regular 20-node Hadoop cluster are shown at right.



Query	3-run average(s)	% improvement
Query 6	34	72%
Query 13	72	71%
Query 15	10	90%
Query 17	106	66%
Query 24	138	56%
Query 25	121	60%
Query 29	108	68%
Query 45	12	83%
Query 48	32	37%
Query 74	94	57%
etc.		

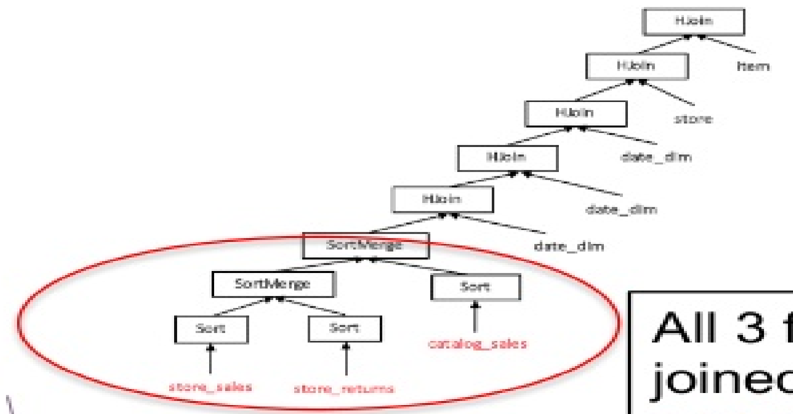
SPARK-17791

Join Reordering Using Star Schema Detection



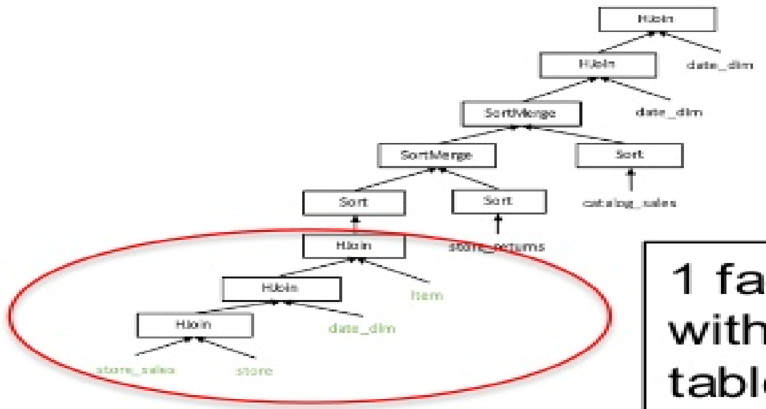
- Execution plans for TPC-DS Query 25:

Default, positional join reordering: suboptimal join of large fact tables.



All 3 fact tables joined early with sort-merge join

Star join reordering: selective star join early in the plan.



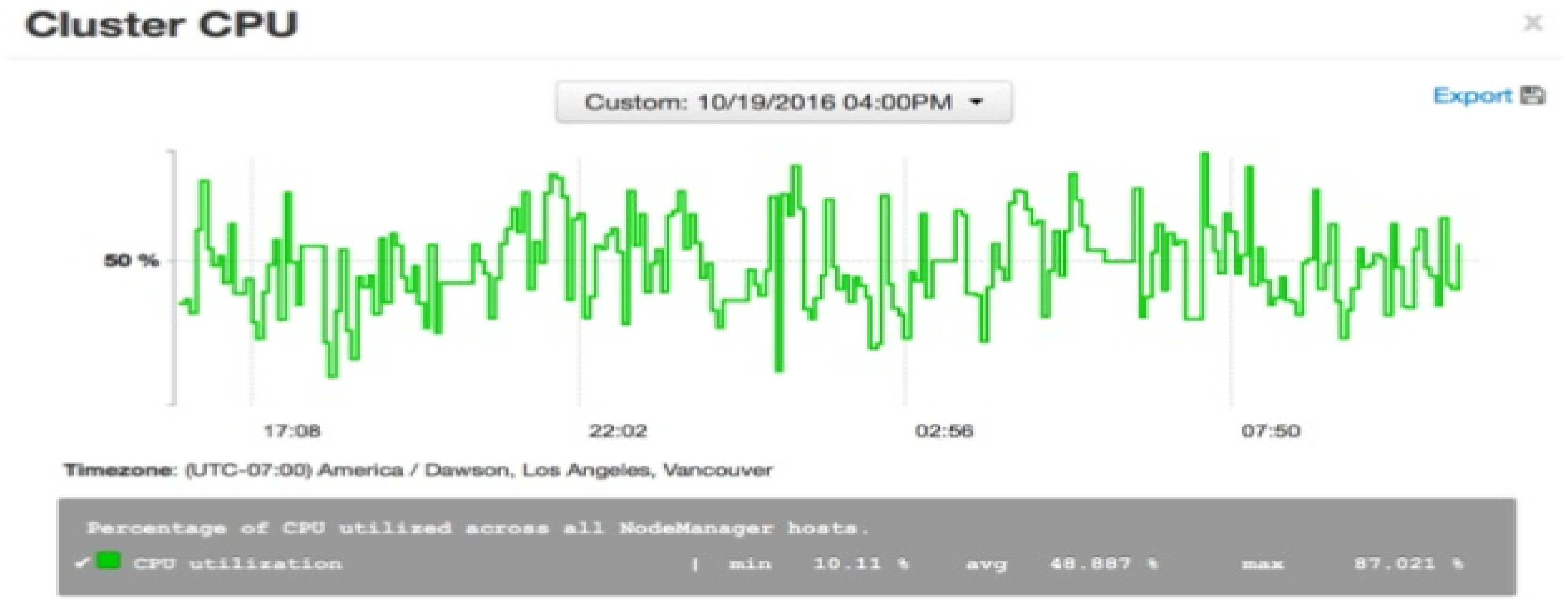
1 fact table joined with dimension tables using hash join

Initial Analysis of “Failures”

- Bad physical plans (4 queries)
  - 5+ hours and never finish at 30TB
    - Did not attempt these at 100TB
  - Optimizer fails to produce reasonable plans
    - Query 14,16,23,95
- Killed after > 1000 “core hours”
  - Investigation required
    - Query 35,77,94
- Exception from container-launch
  - YARN container JVM exits with code 134, core dumps
  - Query 84

TPC-DS Throughput (CPU)

- What kinds of speeds and feeds? During level-load on the cluster with 4 concurent users...
  - Peak CPU utilization = 87%
  - Avg CPU utilization = 49%



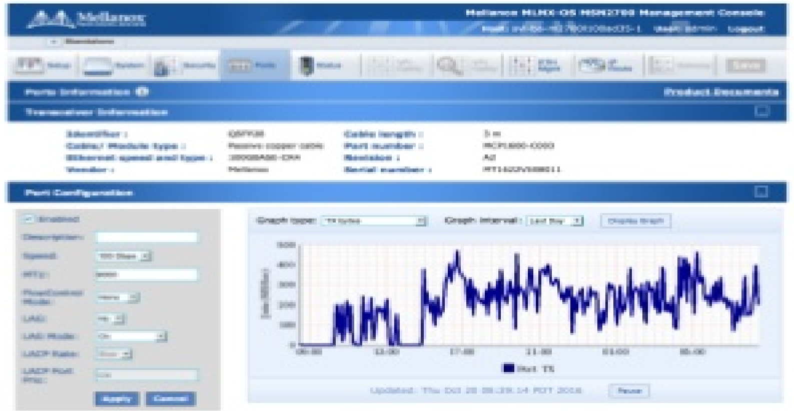
Timezone: (UTC-07:00) America / Dawson, Los Angeles, Vancouver

```
Percentage of CPU utilized across all NodeManager hosts.
✓ 🟩 CPU utilization                  |  min   10.11 %    avg   48.887 %      max     87.021 %
```

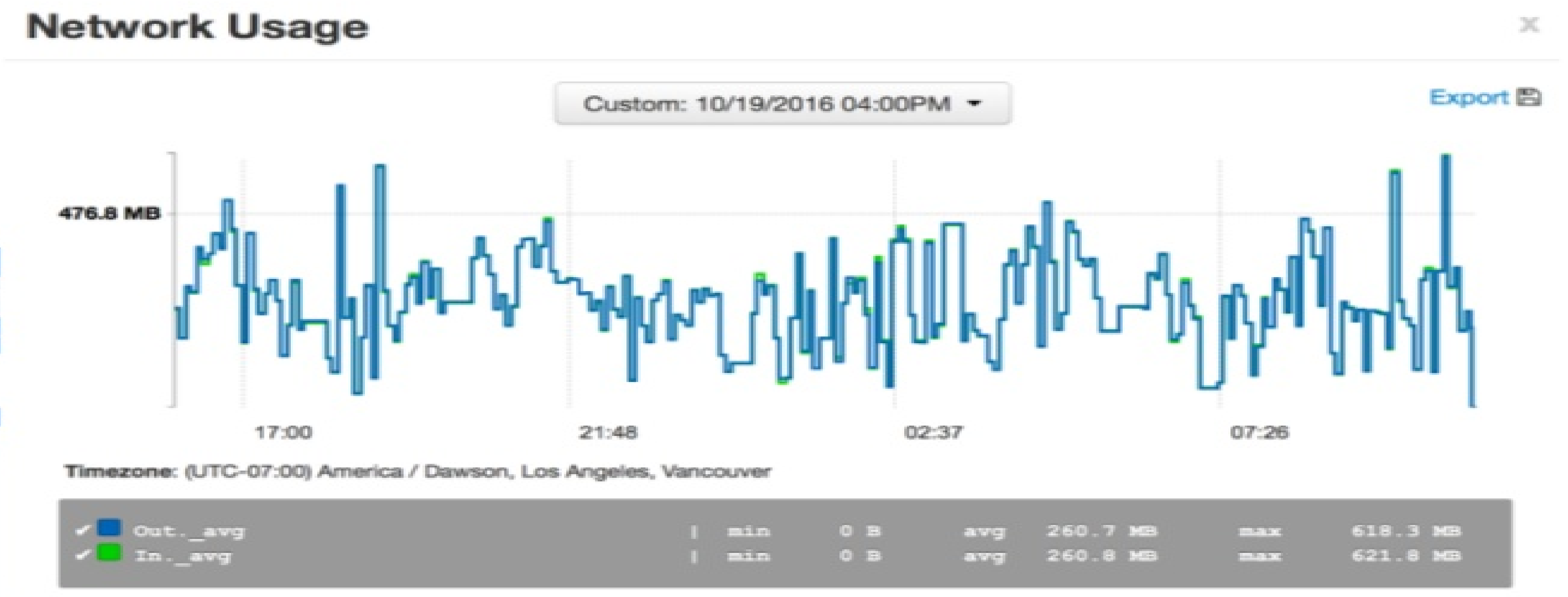
TPC-DS Throughput (Network)

During level-load with 4 concurrent users …

- Peak RX rate = 621 MB/s
- Peak TX rate = 618 MB/s
- Avg RX rate = 261 MB/s
- Avg TX rate = 261 MB/s
- MTU = 9000
- Avg TX total cluster = 58.5 Gb/s



Mellanox console



Timezone: (UTC-07:00) America / Dawson, Los Angeles, Vancouver

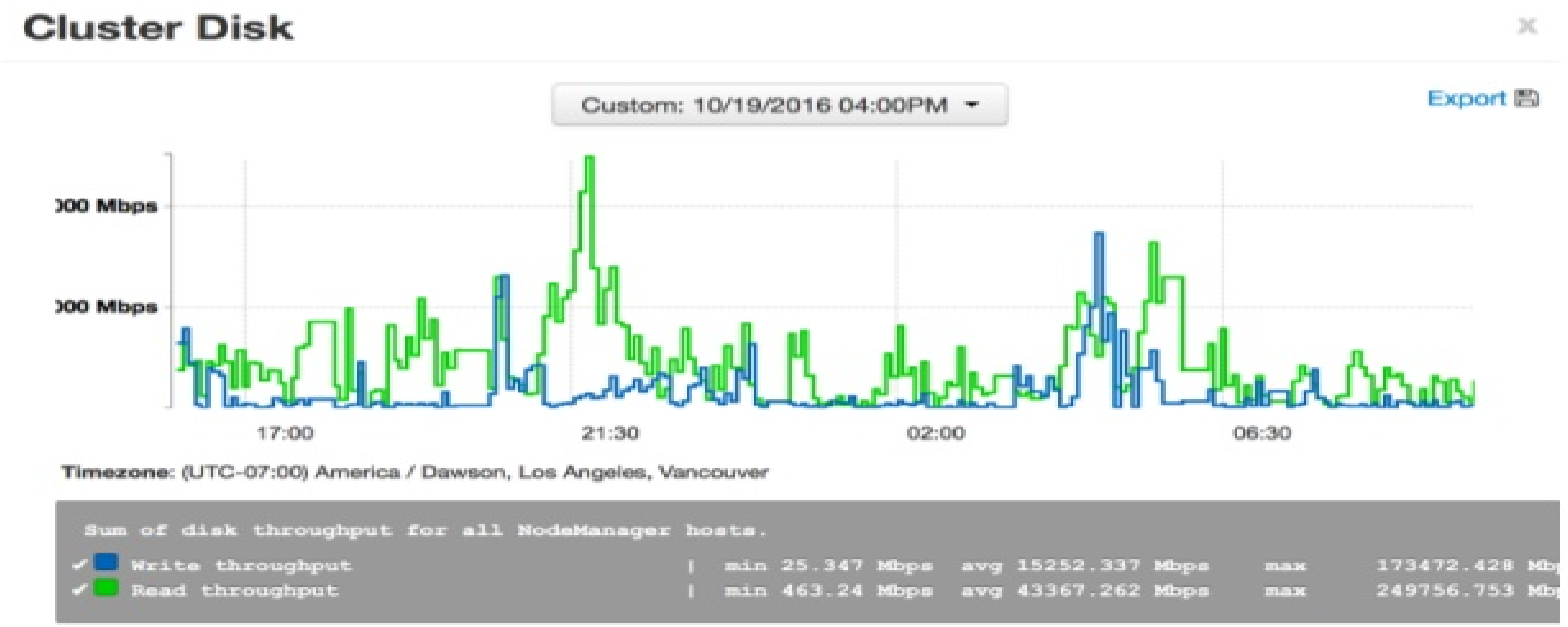
```
✓ 🟦 Out._avg    | min    0 B    avg   260.7 MB    max   618.3 MB
✓ 🟩 In._avg     | min    0 B    avg   260.8 MB    max   621.8 MB
```

A typical data node from Ambari UI

TPC-DS Throughput (I/O)

SSDs I/O during level-load

- Peak writes = 21.7 GB/s
- Peak reads = 31.2 GB/s
- Avg writes = 1.9 GB/s
- Avg reads = 5.4 GB/s



Timezone: (UTC-07:00) America / Dawson, Los Angeles, Vancouver

```
Sum of disk throughput for all NodeManager hosts.
✓ 🟦 Write throughput       |  min 25.347 Mbps   avg 15252.337 Mbps    max     173472.428 Mb
✓ 🟩 Read throughput        |  min 463.24 Mbps   avg 43367.262 Mbps    max     249756.753 Mb
```

SPARK SUMMIT EUROPE 2016

Summary

- Spark SQL 2.0 is a huge step forward. Spark SQL 2.1 seems stable.
  - There are some valuable JIRAs in the queue
- Together, IBM, Intel, Lenovo and Mellanox constructed a "State of the Art" system to test out the hypothesis of a new Spark Cluster topology using latest Spark
- TPC-DS has proven itself to be a valuable and challenging workload
- The initial TPC-DS "Power Test" (single user) did well
  - Completing 92% (91 of 99 queries)
- The initial TPC-DS "Throughput Test" with 4 concurrent users shows good scalability
- There are definitely areas that can and need to improve
  - Great opportunities for the Spark community ☺
- Additional results will be forthcoming as we have time to run additional experiments.

Thank-you!

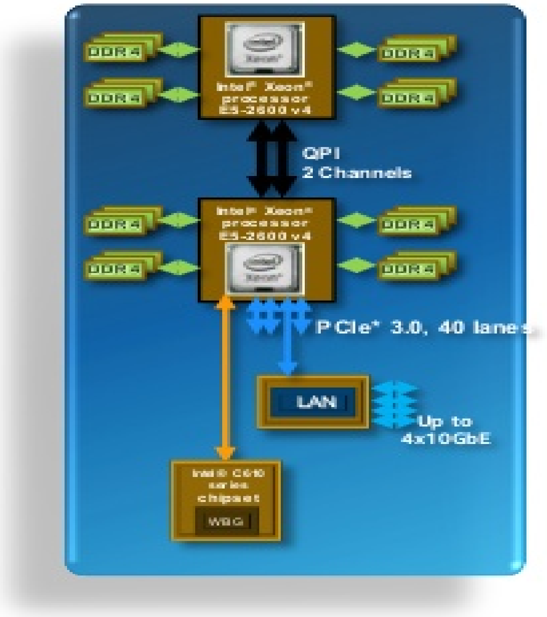
schiefer@ca.ibm.com

SPARK SUMMIT EUROPE 2016

F1 Spark SQL Platform

- Designed to address nearly every "SQL on Hadoop" customer requirement
  - Standard SQL – handles the SQL you use today
  - Fast Query Performance – excellent SQL performance
  - Handles Data Volume – handles big data requirements, easy to scale
- Platform component partners
  - Lenovo – servers cases, mother boards, memory, drive controllers, power supplies
  - Intel – CPUs, NVMe PCIe flash storage
  - Mellanox - ethernet switches, adapters, DAC cables, transceivers
  - IBM – racks, power, cooling, operation, maintenance and performance testing
- "State of the Art" Spark Cluster
  - Latest
    - Lenovo Servers
    - Intel Broadwell CPUs
    - Mellanox 100Gb Ethernet network
    - High-Bandwidth Flash Storage

Intel® Xeon® Processor E5 v4


DDR4
DDR4
DDR4
DDR4
Intel® Xeon® processor E5-2600 v4
QPI
2 Channels
DDR4
DDR4
DDR4
DDR4
Intel® Xeon® processor E5-2600 v4
PCIe* 3.0, 40 lanes
LAN
Up to 4x10GbE
Intel® C610 series chipset
WBG

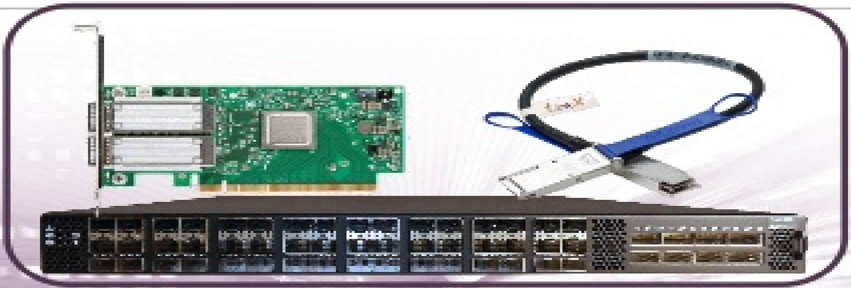
Broadwell CPU	Up to 22 cores; new ISA (Gather improvements, Security, Virtualization and Platform Quality of Service) TDP: Up to 145W (Server); 160W (Workstation only)
Socket	Socket-R3
Scalability	2S capability
Memory	4xDDR4 channels 1600 (3 DPC), 1866 (3 DPC), 2133 (2 DPC), 2400 (1 DPC)
	RDIMM, LRDIMM, 3DS LRDIMM
QPI	2xQPI 1.1 channels 6.4, 8.0, 9.6 GT/s
PCIe*	PCIe* 3.0 (2.5, 5, 8 GT/s) PCIe Extensions: Dual Cast, Atomics
	40xPCIe* 3.0
Wellsburg PCH	DMI2 – 4 lanes; Up to 6xUSB3, 8x USB2 ports, 10xSATA3 ports; GbE MAC (+ External PHY)
LAN	Fortville (40GbE), Springville (1GbE)
Firmware	Servers: Intel® Server Platform Services (SPS) Workstations: ME 9.x

Intel NVMe SSD Drives @2TB

DC P3600 Series SSD 2.5 inch and AIC

Consistently high IOPS and throughput
Sustained low latency
Thermal throttling and monitoring
Power loss protection capacitor self-test

Mellanox 100Gb Ethernet Network



Mellanox TECHNOLOGIES

Connect. Accelerate. Outperform.

ConnectX™-4 Adapters
Advanced Hardware Offloads
More CPU Cores for Analytics

Spectrum™ Switches
Predictable, low long-tail latency
Guaranteed SLA for SQL Engines

LinkX™ Cables
Lowest Bit Error Rate: $1E^{-15}$
Highest Data Integrity

Interesting Facts/Figures

- How long to generate 100TB of data with Map/Reduce? ~7 hours
- How long to ”load the data? (convert to Parquet) ~18 hours
- What was the compression ratio? (Raw vs Parquet file size) 100TB->31TB
- How much disk space is there? 448TB (8x2TB x 28 machines)
- How much free space is there for temp? 267 TB
- How short is the shortest query? 14 seconds
- How long is the longest query (that finished)? 2,586 seconds (Q80)
- What the total row counts for 100TB database? 556 Billion
- What is the row count of the largest table? 288 Billion