SPARK SUMMIT 2017

Deep Dive Into Spark Multi-User Performance

Mikhail Genkin
Performance Architect and Optimization Lead, IBM Spectrum Computing

Michael Feiman (presenter)
Principal Software Architect, IBM Spectrum Computing

Peter Lankford (presenter)
Founder & Director, STAC

Outline

- Introduction
- Factors affecting multi-user performance
- Spark-RM integration architecture
- Analyzing multi-user performance with the Spark Multi-user Benchmark
- Break
- Data and insights
- Q&A
- References

Disclaimer

IBM, the IBM logo, ibm.com, IBM System Storage, IBM Spectrum Storage, IBM Spectrum Control, IBM Spectrum Protect, IBM Spectrum Archive, IBM Spectrum Virtualize, IBM Spectrum Scale, IBM Spectrum Accelerate, Softlayer, and XIV are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

- The following are trademarks or registered trademarks of other companies.
- Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
- IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.
- Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.
- Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
- Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
- ITIL is a Registered Trade Mark of AXELOS Limited.
- UNIX is a registered trademark of The Open Group in the United States and other countries.

* All other products may be trademarks or registered trademarks of their respective companies.

- **STAC and all STAC names are trademarks or registered trademarks of the Securities Technology Analysis Center, LLC.**

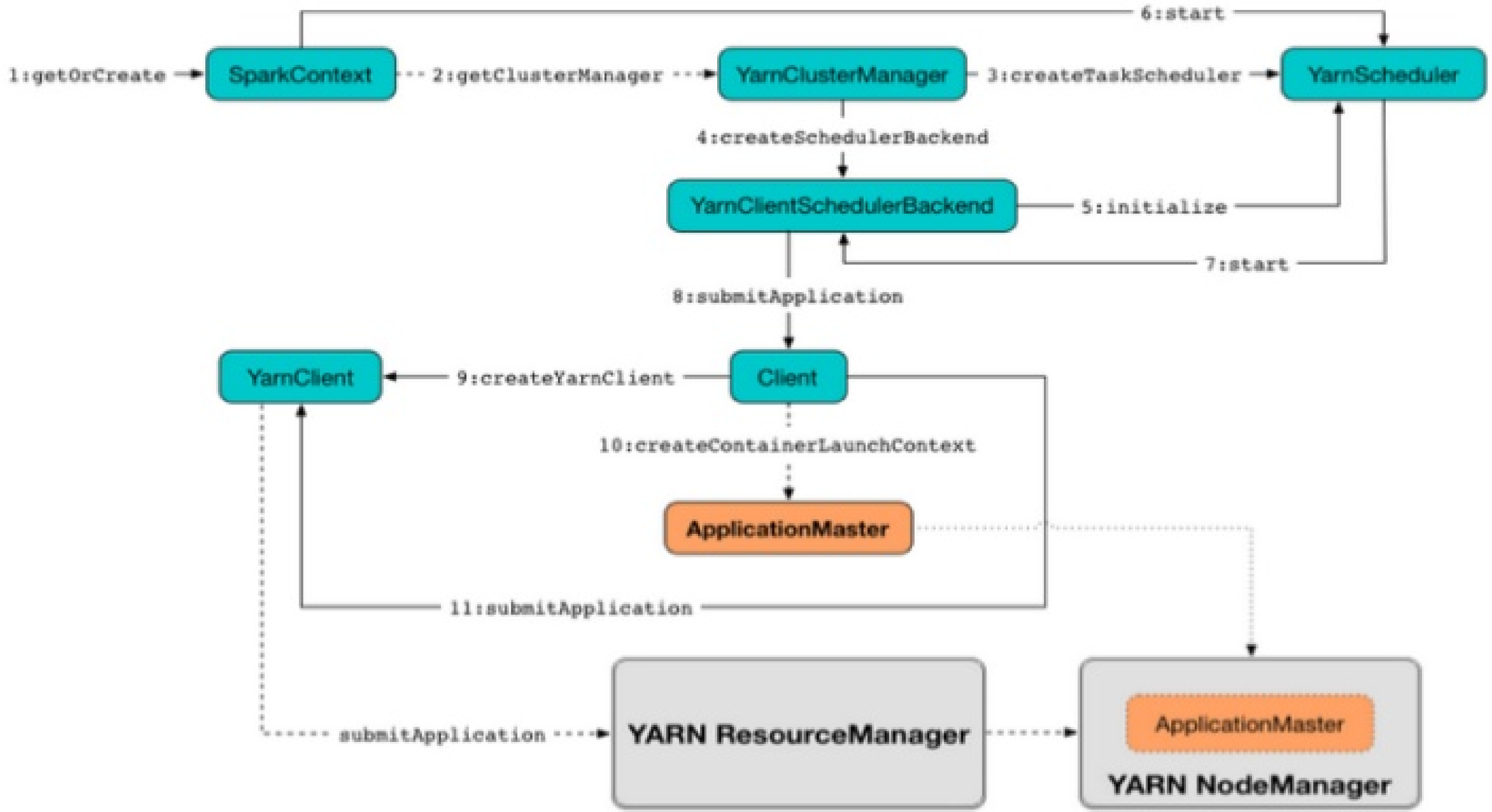
Introduction

- Most Spark production deployments are multi-user or multi-tenant
- Understanding factors that affect performance of Spark applications under multi-user or multi-tenant conditions is very important
- In this session we focus on Spark performance with YARN, Mesos and IBM Spectrum Conductor with Spark
- IBM contracted STAC to perform and independent evaluation of 3 common Spark resource manager configurations:
  - Spark 2.0.2 (dev) + YARN 2.7.3
  - Spark 2.0.1 + Mesos 1.0.1
  - IBM Spectrum Conductor with Spark 2.1.0.1 (includes Spark 2.0.1)
- IBM views this as a first step in providing more data-driven community engagement on Spark resource management

Factors Affecting Multi-User/Tenant Performance

- Resource manager configuration and behavior
  - Resource manager controls:
    - how many CPUs and how much memory a given app gets
    - how many Spark applications will run concurrently
    - can have order-of-magnitude impact on a given application performance
- Spark configuration and tuning
  - Some Spark tunables, such as ***spark.executor.memory*** affect how many Spark executors and applications will run concurrently
- Number of concurrently running users and applications
  - More concurrent applications means less resources per application and slower performance
- Cluster infrastructure
  - Capacity and presence or absence of bottlenecks
- Data scale used by the applications
  - More data usually results in slower performance
- Type of processing performed by the app
  - Spark SQL vs. machine learning for example
- Others

Spark-Resource Manager Integration



From: https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/yarn/

Spark-RM Integration Architecture

- Interplay between Spark and Resource Manager configurations
- Spark determines:
  - Executor memory
  - Executor CPU
- Resource manager determines (directly and indirectly):
  - How many CPUs and how much memory each application gets initially
  - Whether applications can release or gain resources during execution (dynamic allocation)
  - Whether resources can be taken from one application to boost another (pre-emption)
  - How many Spark executors will be started concurrently
  - How many Spark tasks will execute in parallel

Spark Multi-User Benchmark (SMB)

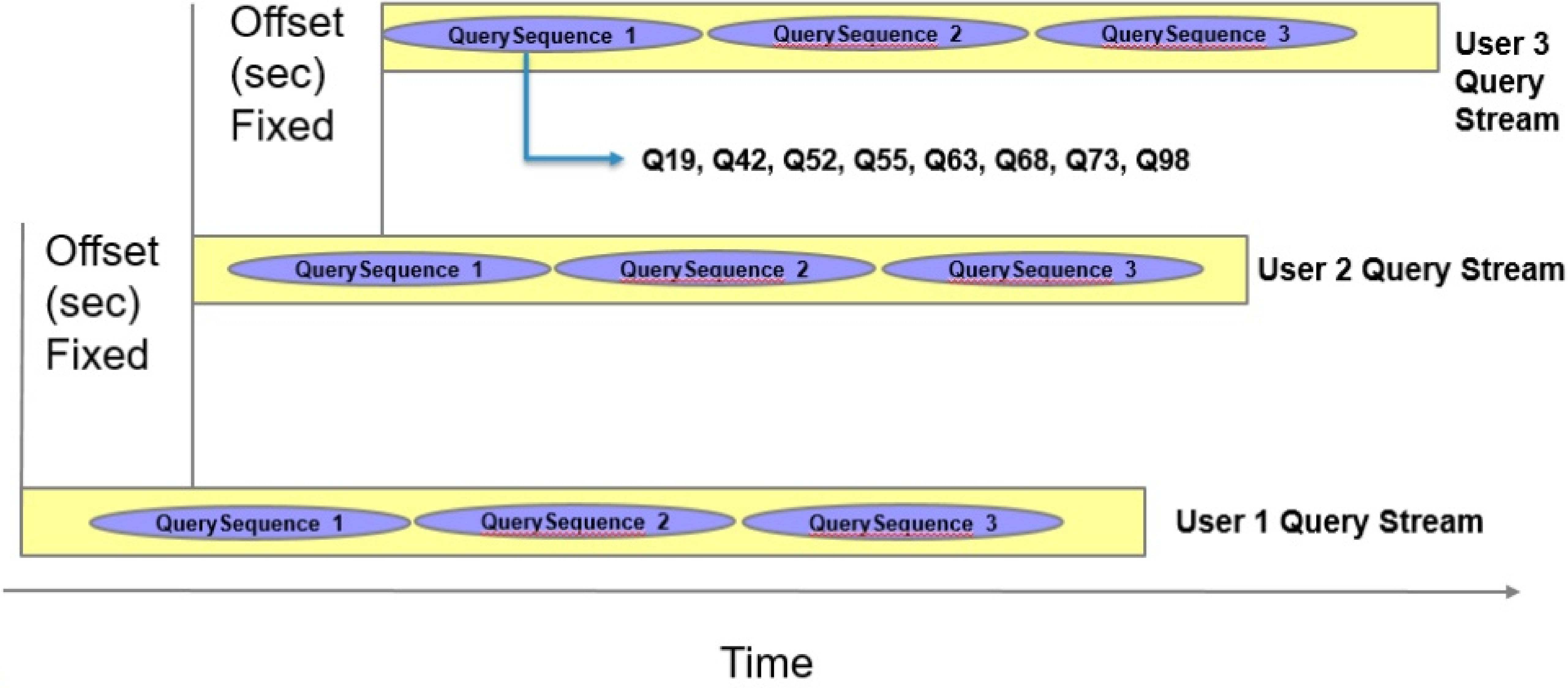
- Open-source benchmark: www.github.com/IBM/SparkMultiuserBenchmark
  - Initially developed at IBM
- SMB is designed to measure performance under very realistic workload conditions
  - Simulates steady-state, non-steady state job-arrival patterns
  - Mix of TPC-DS-inspired queries and machine learning jobs
  - Short-running synchronous interactive queries executed in parallel by multiple users
  - Longer-running batch jobs and queries executed in parallel by multiple users
  - Mixed workloads – interactive + batch – executed in parallel by multiple users
  - Multi-tenant workloads – interactive + batch – executed in parallel by multiple users with different weighting or QoS constraints
- How it can be used:
  - Tune and optimize your Spark applications and cluster hardware infrastructure
  - Study resource manager behavior and efficiency

SMB-1 and SMB-2

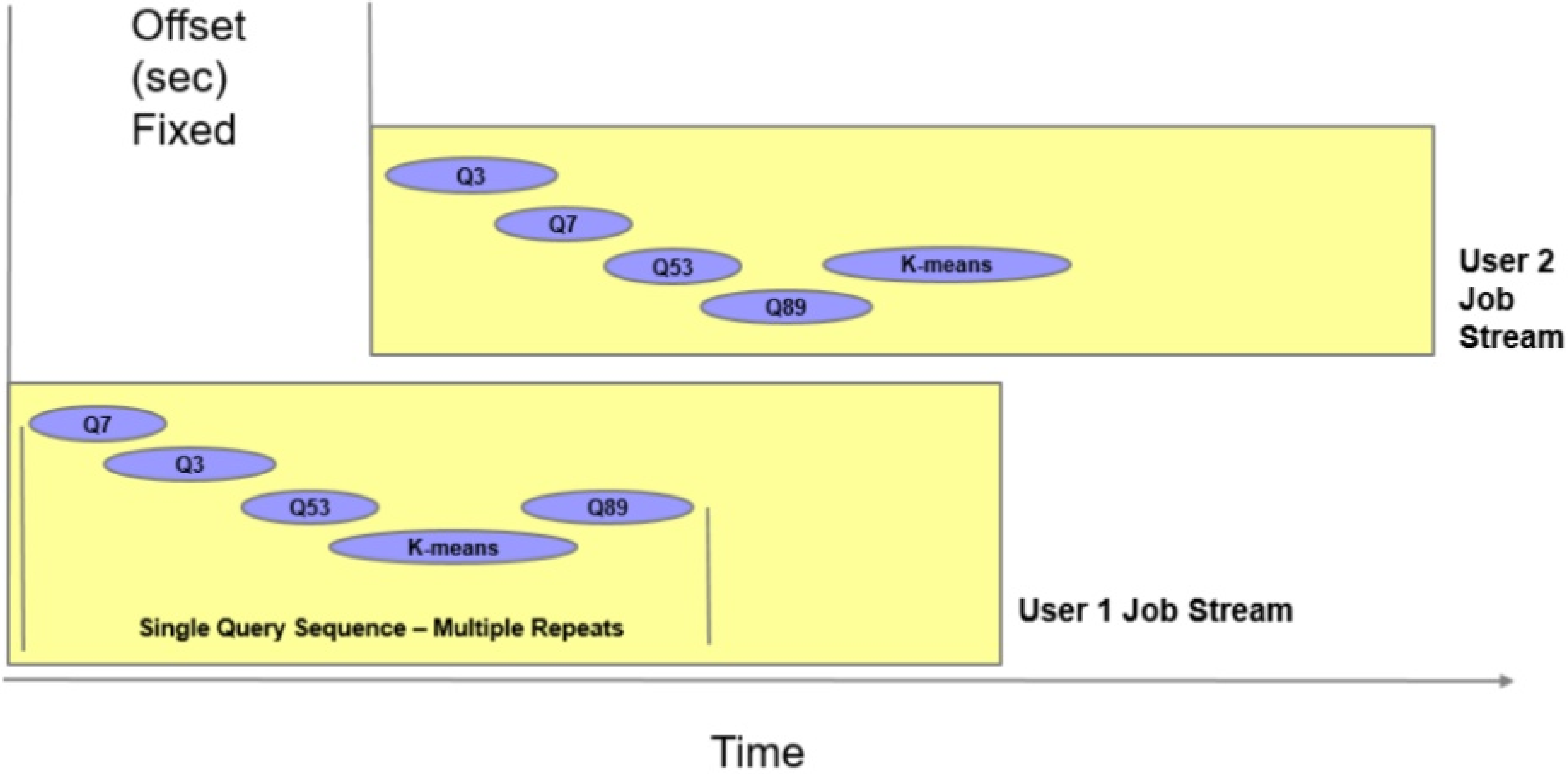
- Workload patterns:
  1. Synchronous interactive workload
     - SMB-1 and SMB-2
     - Short running queries – 5 to 20 sec duration
     - TPC-DS-inspired queries
  2. Asynchronous batch workload
     - SMB-2
     - Longer running queries – 30 sec to 5 min duration
     - TPC-DS-inspired queries
     - K-means machine learning jobs

- Use cases:
  1. Synchronous interactive multi-user
     - SMB-1 and SMB-2
     - Workload pattern 1 only
     - All users have the same weight and QoS
  2. Asynchronous batch multi-user
     - SMB-2 only
     - Workload pattern 2 only
     - All users have the same weight and QoS
  3. Mixed multi-user
     - SMB-2 only
     - Workload mix:
       - 50% interactive users
       - 50% batch users
     - Same weight for all users
  4. Mixed multi-tenant
     - SMB-2 only
     - Workload mix:
       - 50% interactive users
       - 50% batch users
     - Interactive has priority
     - Interactive can use up to 70% of resources
     - Batch can use 100% of resources when interactive is not running
     - Batch can use not more than 30% when interactive is running

SMB-2 Synchronous Interactive Workload Pattern

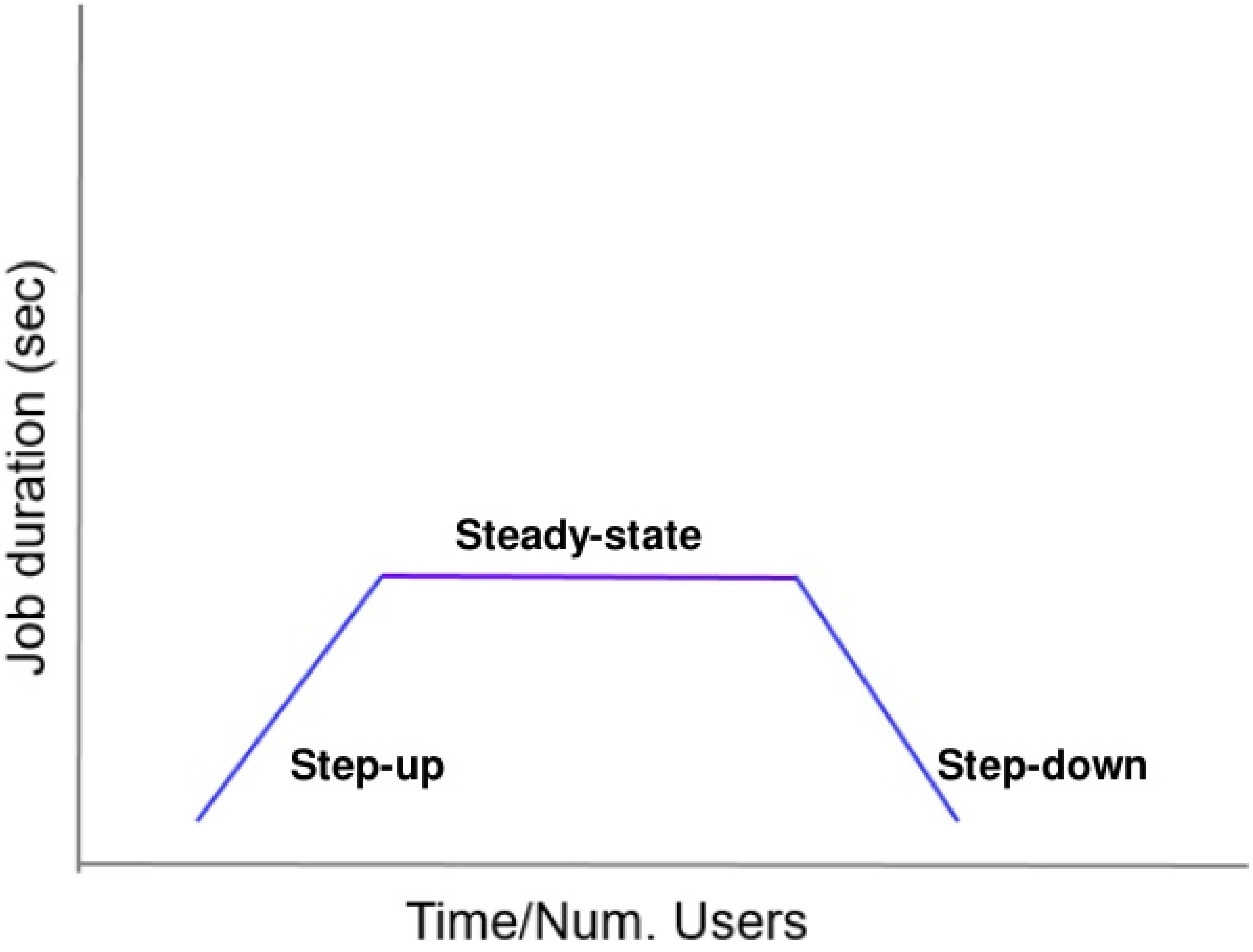


SMB-2 Asynchronous Batch Workload Pattern



Interpreting SMB Data

- Coarse metrics
  - Total test duration
  - Combined query throughput (qph)
  - Average weighted Relative Standard Deviation (RSD)
    - RSD = Std.Dev./Avg. *100
    - Measures variability
- Detailed per-query metrics
  - Query RSD
  - Avg. query duration
  - Standard deviation for query durations
  - Plot of query durations for all queries vs. test duration (pattern shown on the right)



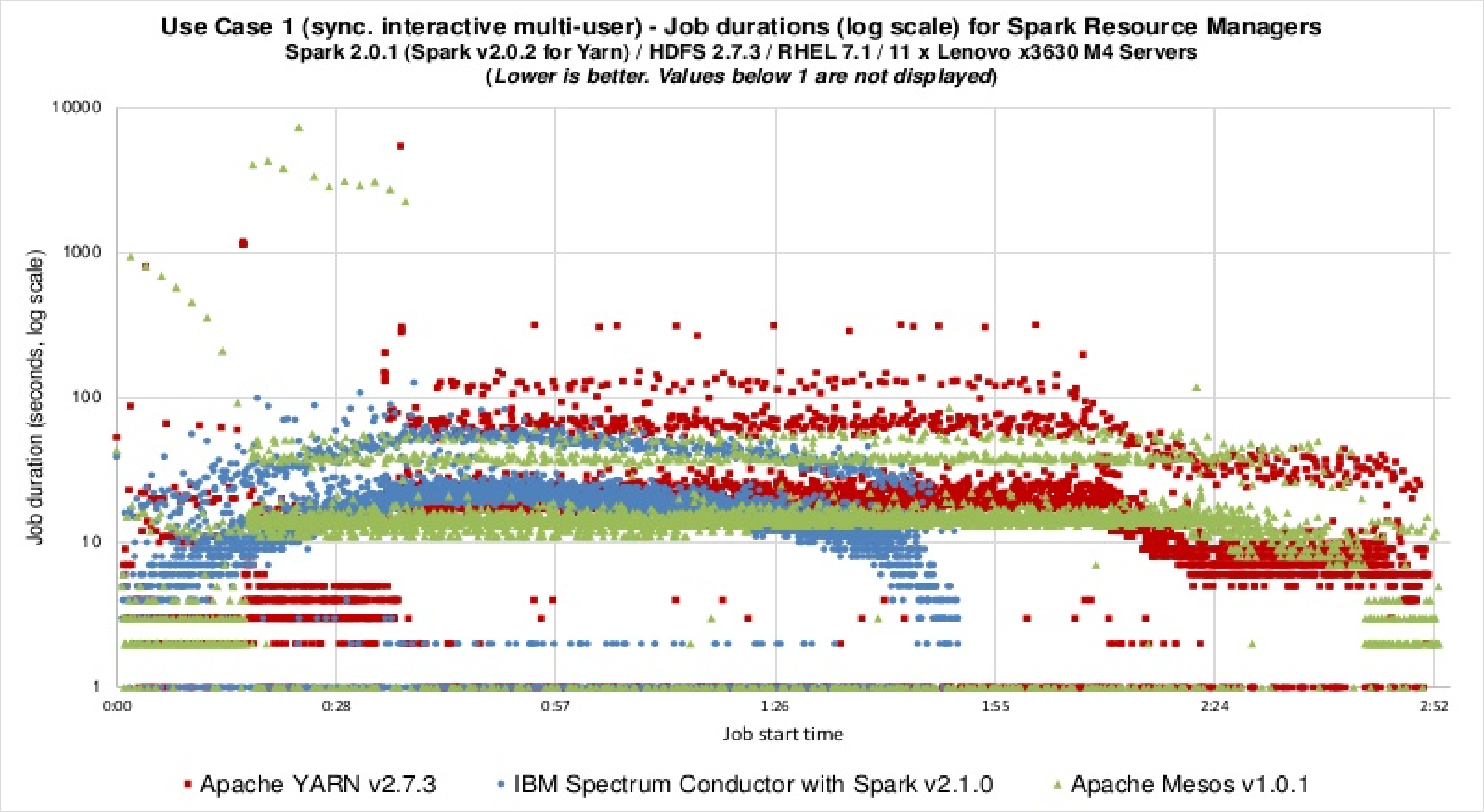
Break – 10 min

Part 2 – Data and Insights

So what is STAC®?

- STAC facilitates the STAC Benchmark Council:
  - ~300 financial firms and ~50 tech vendors
  - Establishes standard technology benchmarks & promotes dialog
  - Big data, fast data, fast compute, big compute
- STAC performs independent benchmark audits

Visualization – Use Case 1

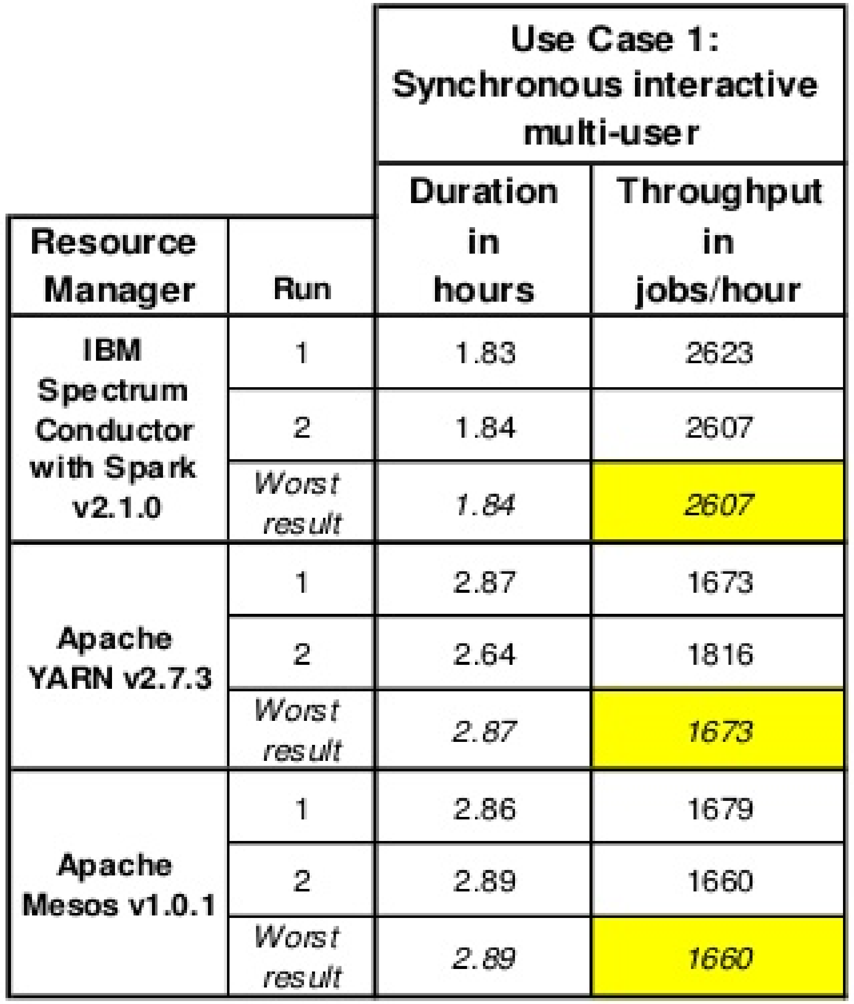


Use Case 1 (sync. interactive multi-user) - Job durations (log scale) for Spark Resource Managers

Spark 2.0.1 (Spark v2.0.2 for Yarn) / HDFS 2.7.3 / RHEL 7.1 / 11 x Lenovo x3630 M4 Servers

(*Lower is better. Values below 1 are not displayed*)

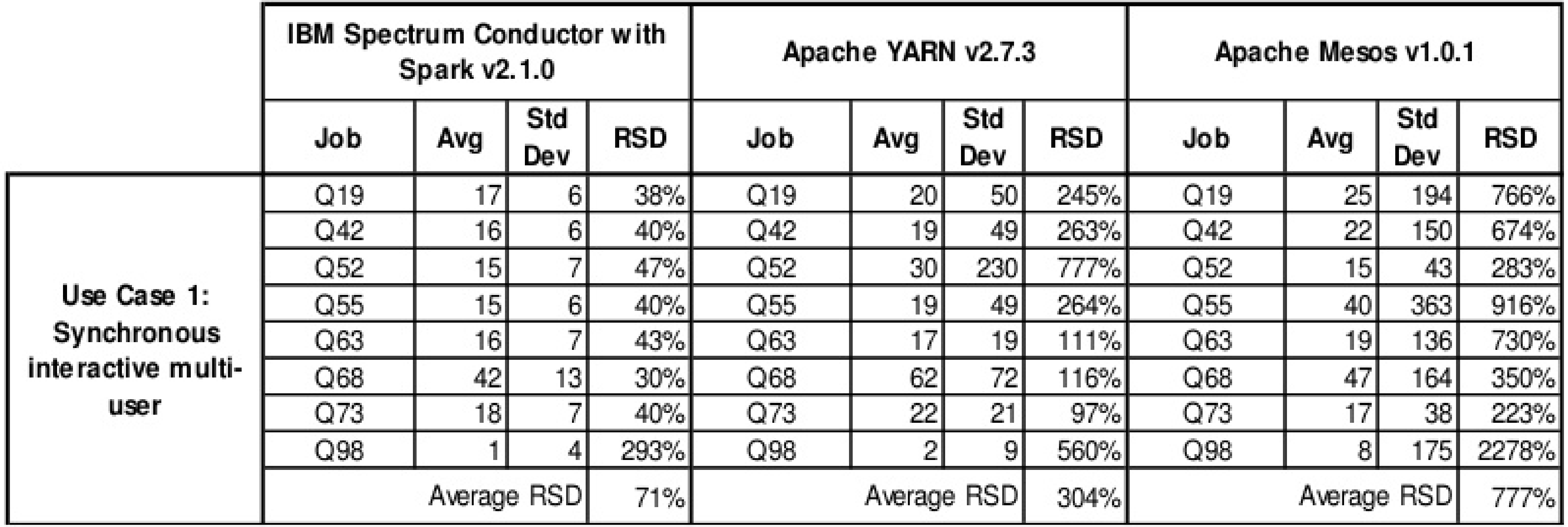
Throughput – Use Case 1



		Use Case 1: Synchronous interactive multi-user	
Resource Manager	Run	Duration in hours	Throughput in jobs/hour
IBM Spectrum Conductor with Spark v2.1.0	1	1.83	2623
	2	1.84	2607
	Worst result	*1.84*	*2607*
Apache YARN v2.7.3	1	2.87	1673
	2	2.64	1816
	Worst result	*2.87*	*1673*
Apache Mesos v1.0.1	1	2.86	1679
	2	2.89	1660
	Worst result	*2.89*	*1660*

- IBM had 56% higher throughput than YARN and 57% higher than Mesos
- YARN and Mesos were nearly the same

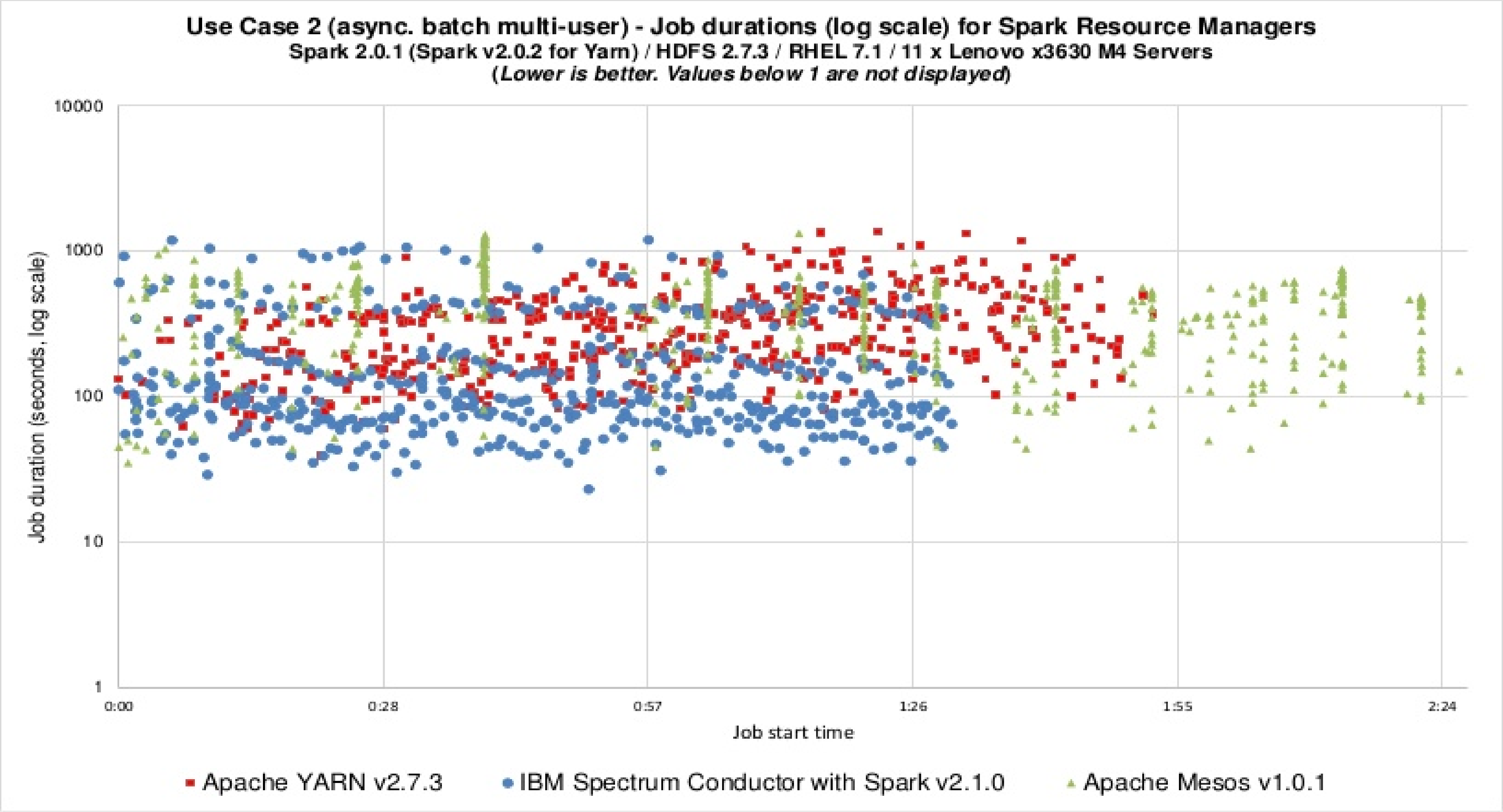
Variability – Use Case 1



	IBM Spectrum Conductor with Spark v2.1.0				Apache YARN v2.7.3				Apache Mesos v1.0.1			
	Job	Avg	Std Dev	RSD	Job	Avg	Std Dev	RSD	Job	Avg	Std Dev	RSD
Use Case 1: Synchronous interactive multi-user	Q19	17	6	38%	Q19	20	50	245%	Q19	25	194	766%
	Q42	16	6	40%	Q42	19	49	263%	Q42	22	150	674%
	Q52	15	7	47%	Q52	30	230	777%	Q52	15	43	283%
	Q55	15	6	40%	Q55	19	49	264%	Q55	40	363	916%
	Q63	16	7	43%	Q63	17	19	111%	Q63	19	136	730%
	Q68	42	13	30%	Q68	62	72	116%	Q68	47	164	350%
	Q73	18	7	40%	Q73	22	21	97%	Q73	17	38	223%
	Q98	1	4	293%	Q98	2	9	560%	Q98	8	175	2278%
			Average RSD	71%			Average RSD	304%			Average RSD	777%

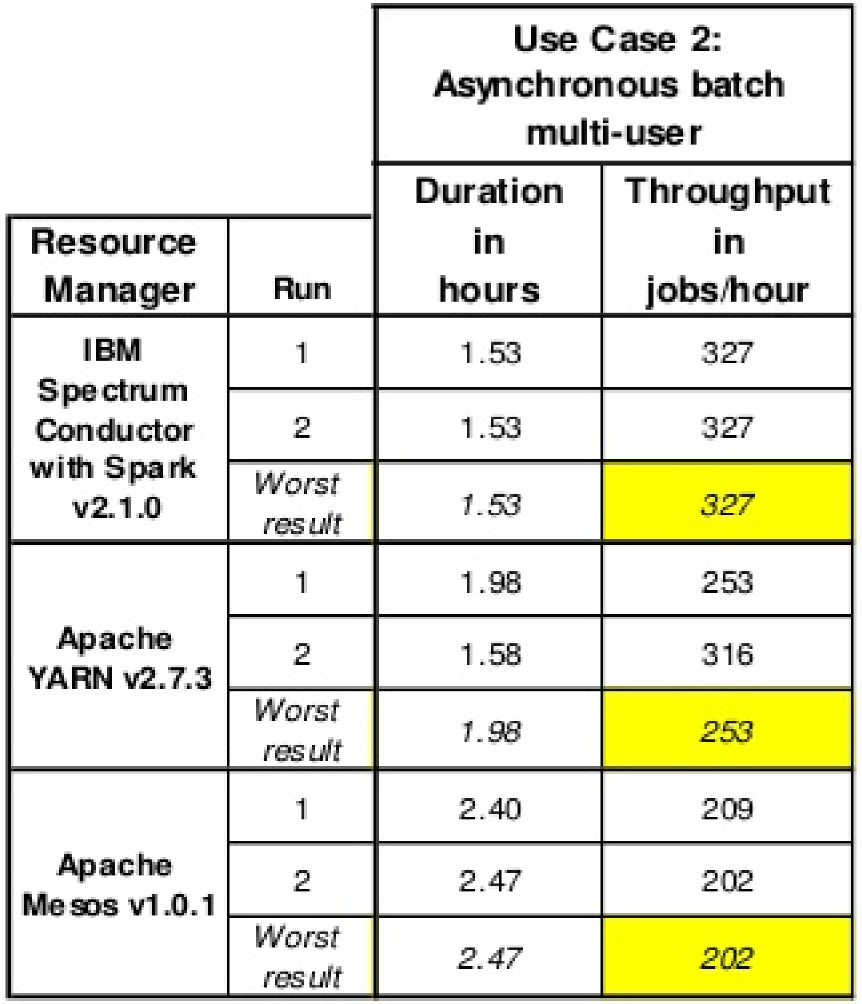
- IBM had 4.3x lower average RSD than YARN and 10.9x lower than Mesos
- YARN had 2.6x lower average RSD than Mesos

Visualization – Use Case 2



SPARK SUMMIT 2017

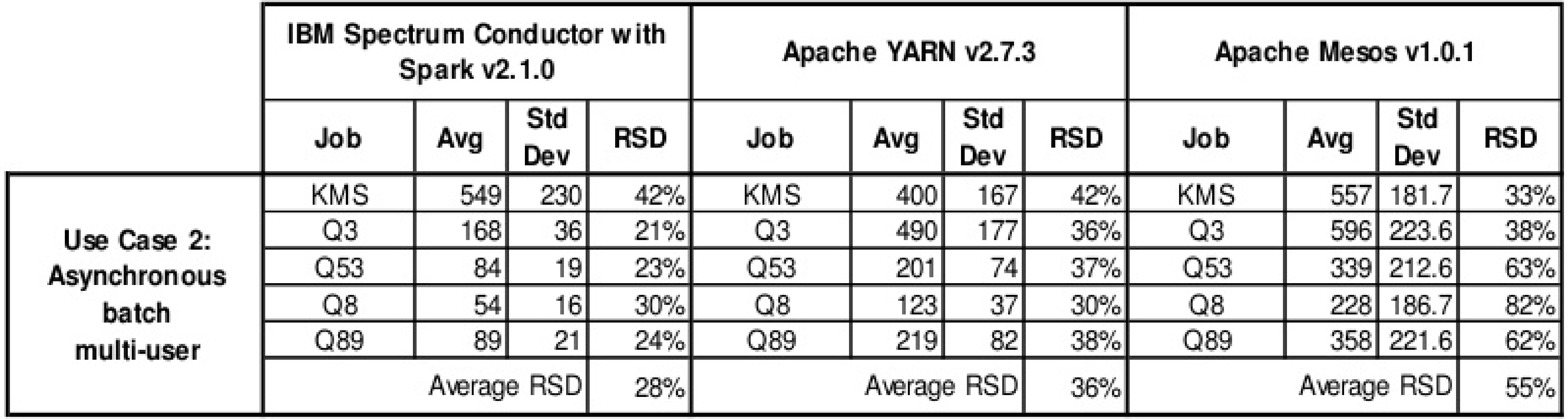
Throughput – Use Case 2



		Use Case 2: Asynchronous batch multi-user	
Resource Manager	Run	Duration in hours	Throughput in jobs/hour
IBM Spectrum Conductor with Spark v2.1.0	1	1.53	327
	2	1.53	327
	Worst result	*1.53*	*327*
Apache YARN v2.7.3	1	1.98	253
	2	1.58	316
	Worst result	*1.98*	*253*
Apache Mesos v1.0.1	1	2.40	209
	2	2.47	202
	Worst result	*2.47*	*202*

- IBM throughput was 30% higher than YARN and 62% higher than Mesos
- YARN throughput was 25% higher than Mesos

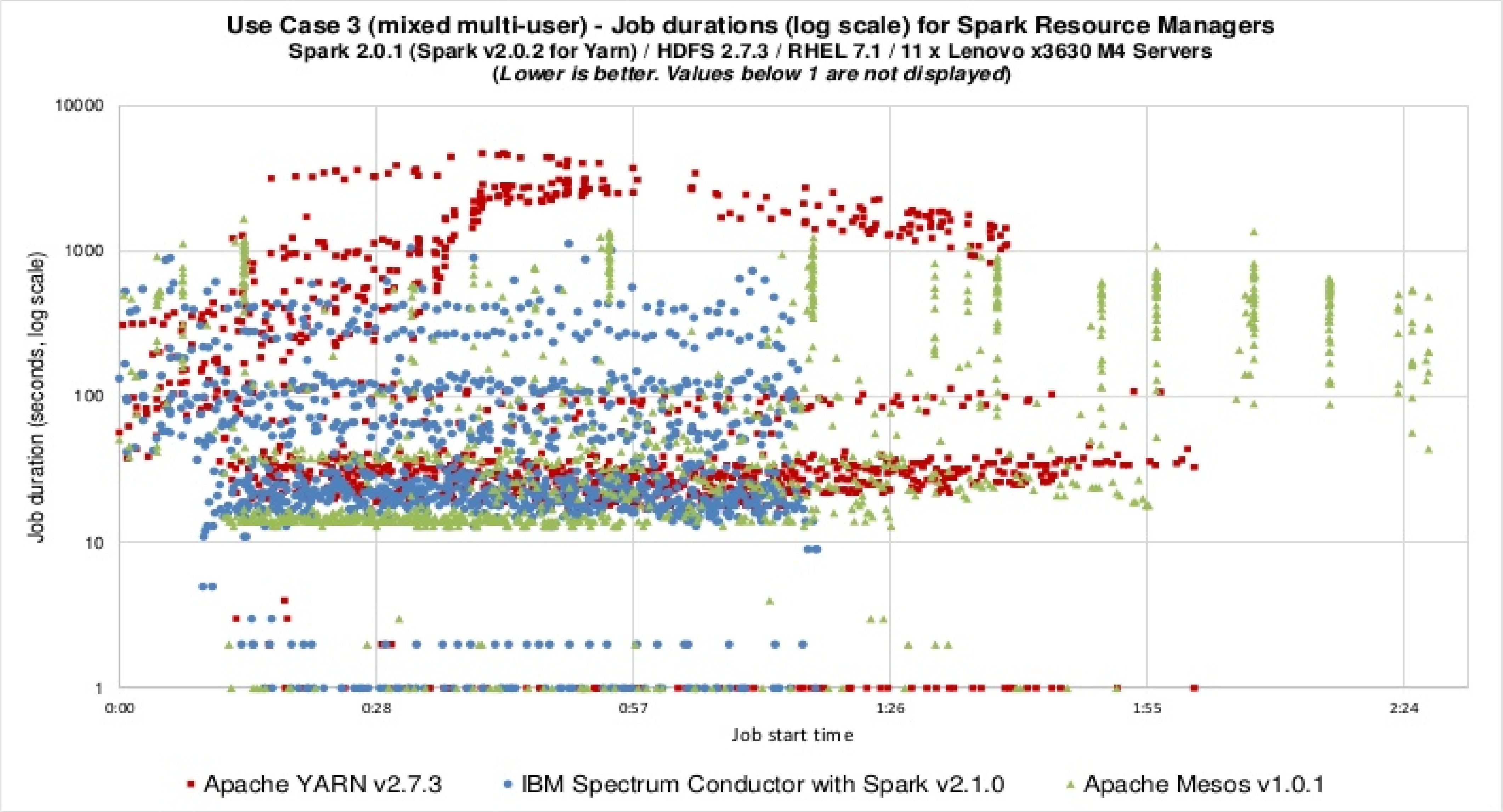
Variability – Use Case 2



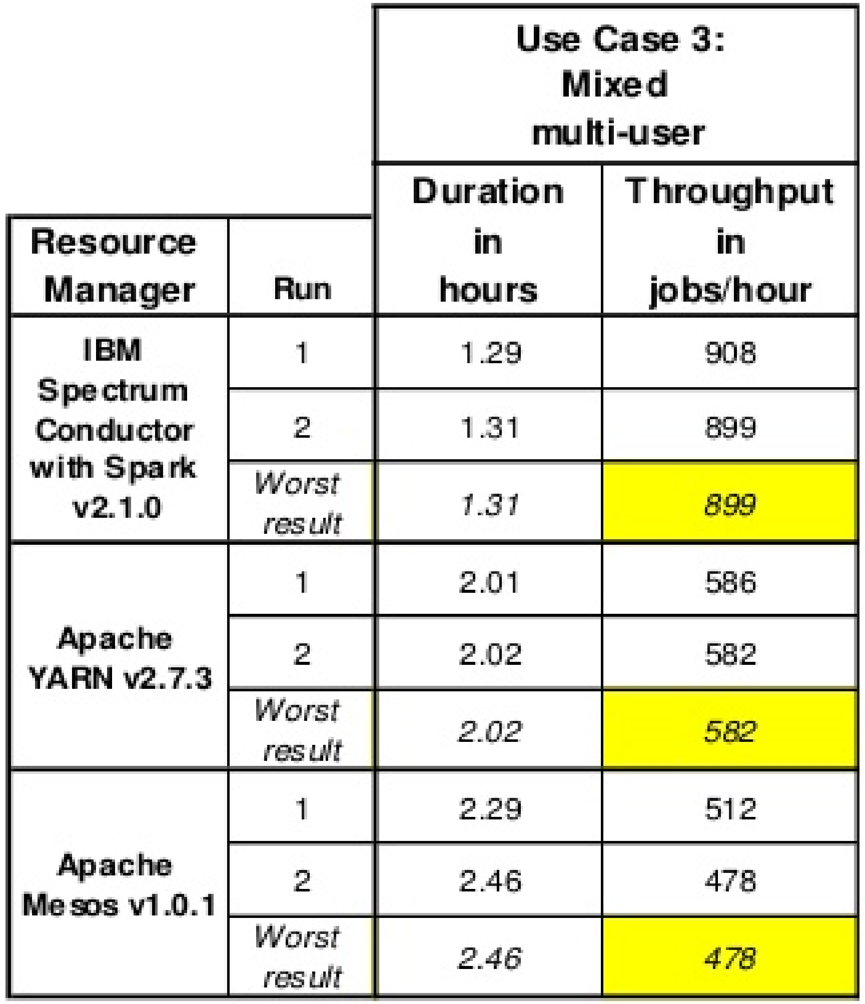
	IBM Spectrum Conductor with Spark v2.1.0				Apache YARN v2.7.3				Apache Mesos v1.0.1			
	Job	Avg	Std Dev	RSD	Job	Avg	Std Dev	RSD	Job	Avg	Std Dev	RSD
Use Case 2: Asynchronous batch multi-user	KMS	549	230	42%	KMS	400	167	42%	KMS	557	181.7	33%
	Q3	168	36	21%	Q3	490	177	36%	Q3	596	223.6	38%
	Q53	84	19	23%	Q53	201	74	37%	Q53	339	212.6	63%
	Q8	54	16	30%	Q8	123	37	30%	Q8	228	186.7	82%
	Q89	89	21	24%	Q89	219	82	38%	Q89	358	221.6	62%
			Average RSD	28%			Average RSD	36%			Average RSD	55%

- IBM had 1.3x lower average RSD than YARN and 2.0x lower than Mesos
- YARN had 1.5x lower average RSD than Mesos

Visualization – Use Case 3



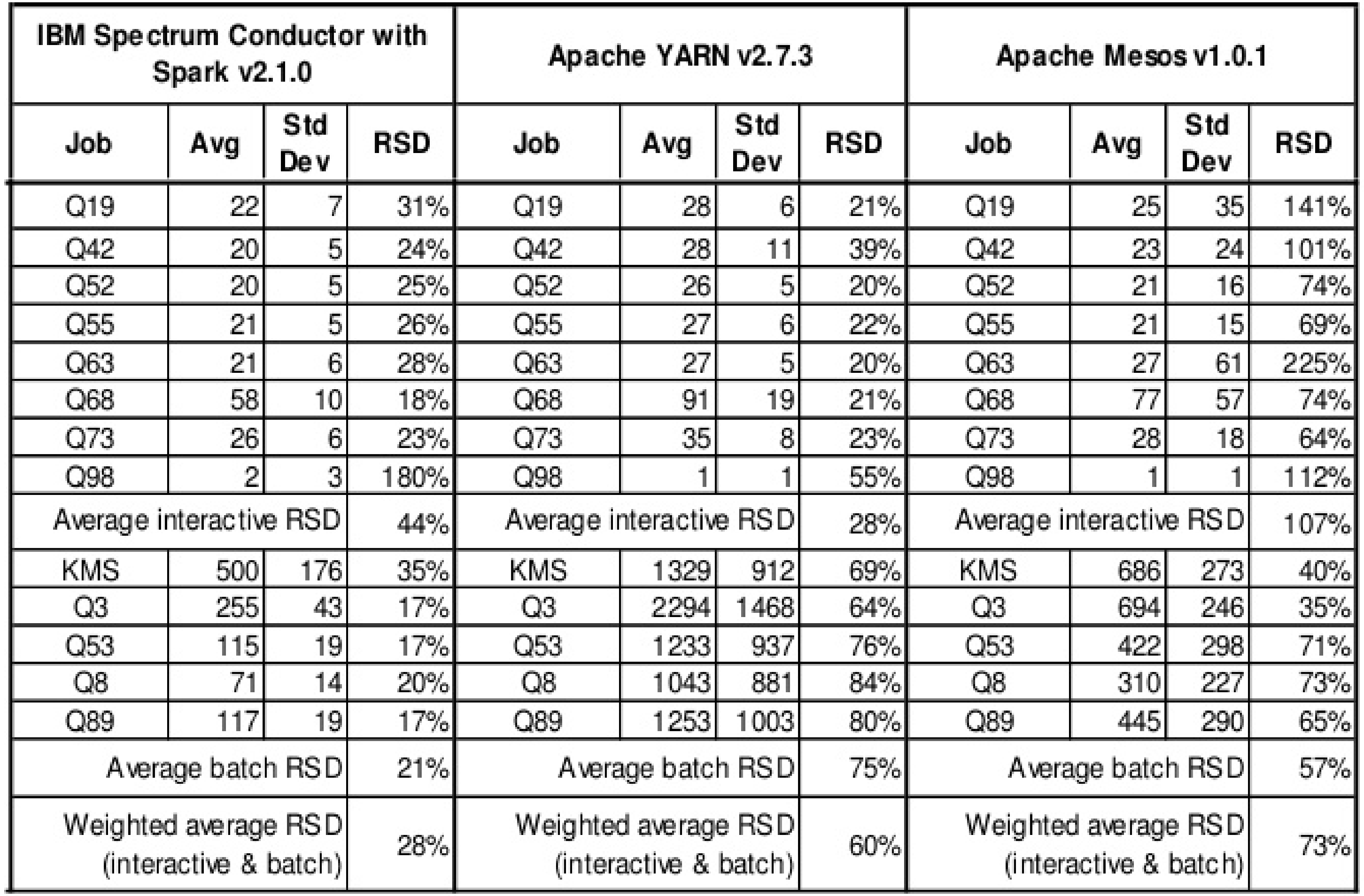
Throughput – Use Case 3



		Use Case 3: Mixed multi-user	
Resource Manager	Run	Duration in hours	Throughput in jobs/hour
IBM Spectrum Conductor with Spark v2.1.0	1	1.29	908
	2	1.31	899
	Worst result	*1.31*	*899*
Apache YARN v2.7.3	1	2.01	586
	2	2.02	582
	Worst result	*2.02*	*582*
Apache Mesos v1.0.1	1	2.29	512
	2	2.46	478
	Worst result	*2.46*	*478*

- IBM throughput was 55% higher than YARN and 88% higher than Mesos
- YARN throughput was 22% higher than Mesos

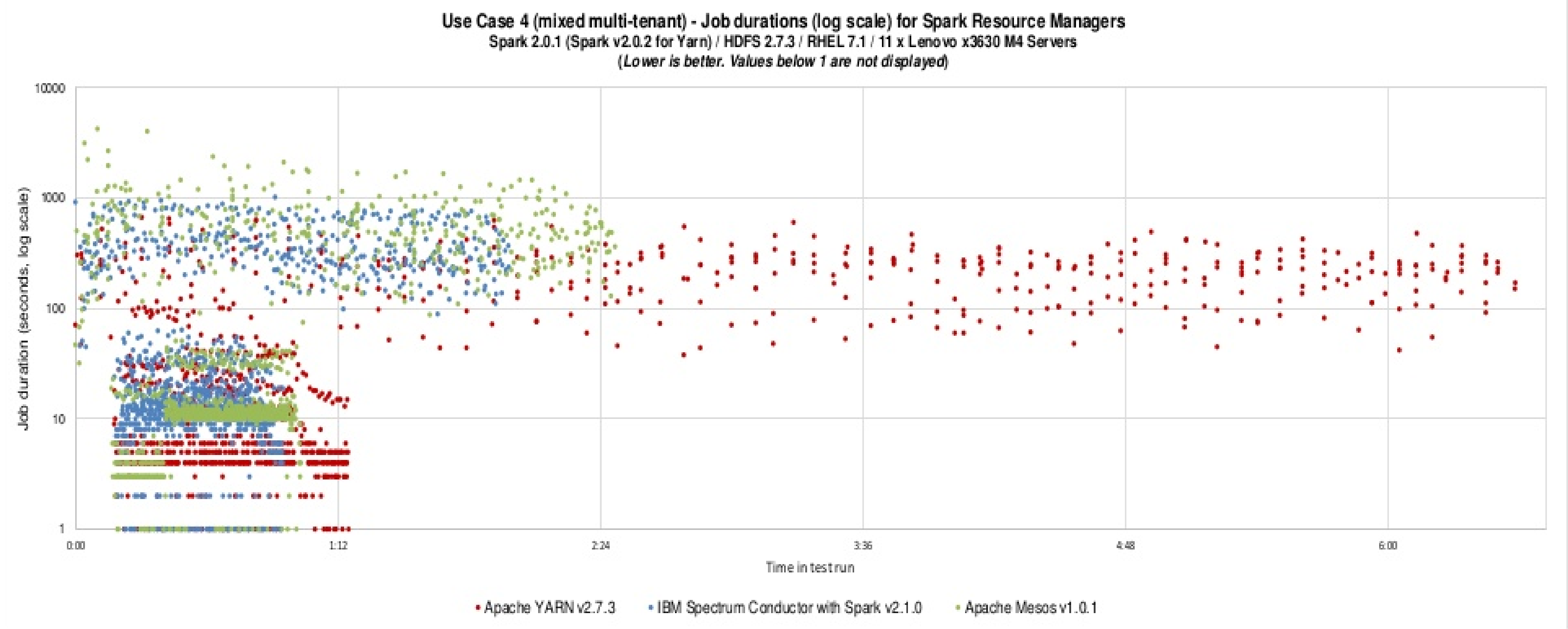
Variability – Use Case 3



IBM Spectrum Conductor with Spark v2.1.0				Apache YARN v2.7.3				Apache Mesos v1.0.1			
Job	Avg	Std Dev	RSD	Job	Avg	Std Dev	RSD	Job	Avg	Std Dev	RSD
Q19	22	7	31%	Q19	28	6	21%	Q19	25	35	141%
Q42	20	5	24%	Q42	28	11	39%	Q42	23	24	101%
Q52	20	5	25%	Q52	26	5	20%	Q52	21	16	74%
Q55	21	5	26%	Q55	27	6	22%	Q55	21	15	69%
Q63	21	6	28%	Q63	27	5	20%	Q63	27	61	225%
Q68	58	10	18%	Q68	91	19	21%	Q68	77	57	74%
Q73	26	6	23%	Q73	35	8	23%	Q73	28	18	64%
Q98	2	3	180%	Q98	1	1	55%	Q98	1	1	112%
Average interactive RSD			44%	Average interactive RSD			28%	Average interactive RSD			107%
KMS	500	176	35%	KMS	1329	912	69%	KMS	686	273	40%
Q3	255	43	17%	Q3	2294	1468	64%	Q3	694	246	35%
Q53	115	19	17%	Q53	1233	937	76%	Q53	422	298	71%
Q8	71	14	20%	Q8	1043	881	84%	Q8	310	227	73%
Q89	117	19	17%	Q89	1253	1003	80%	Q89	445	290	65%
Average batch RSD			21%	Average batch RSD			75%	Average batch RSD			57%
Weighted average RSD (interactive & batch)			28%	Weighted average RSD (interactive & batch)			60%	Weighted average RSD (interactive & batch)			73%

- IBM had 2.1x lower average RSD than YARN and 2.6x lower than Mesos
- YARN had 1.2x lower average RSD than Mesos

Visualization – Use Case 4

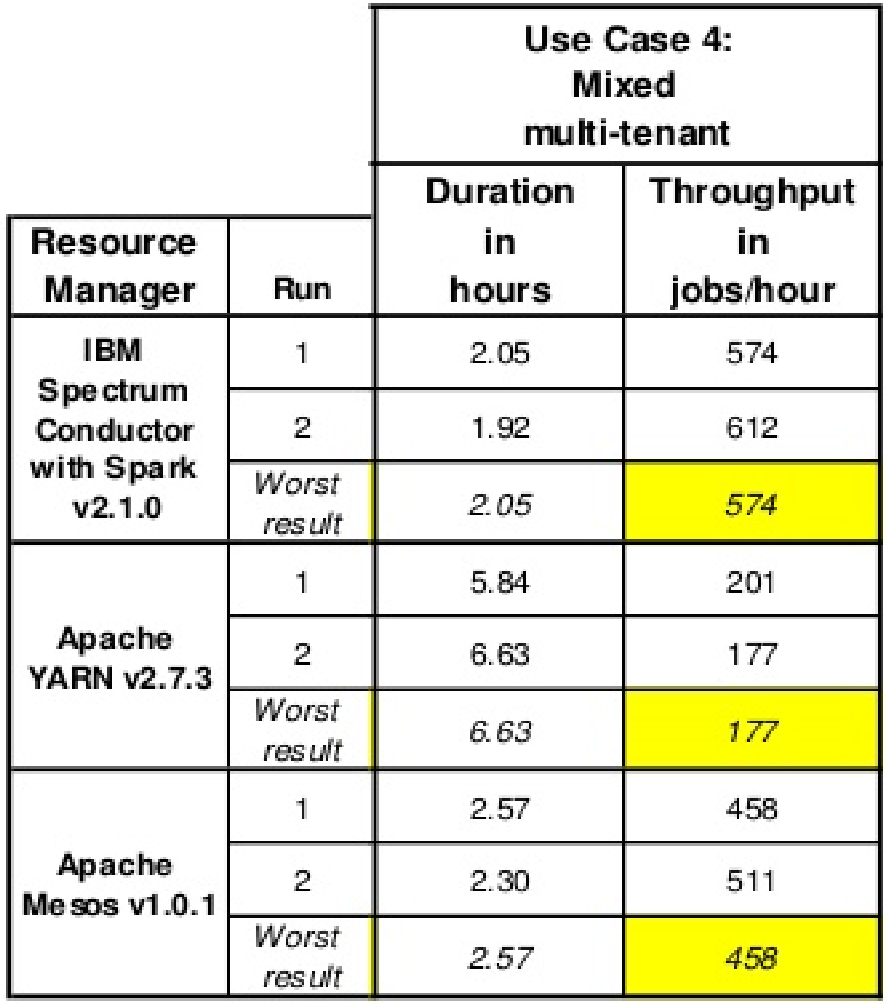


Use Case 4 (mixed multi-tenant) - Job durations (log scale) for Spark Resource Managers

Spark 2.0.1 (Spark v2.0.2 for Yarn) / HDFS 2.7.3 / RHEL 7.1 / 11 x Lenovo x3630 M4 Servers

(*Lower is better. Values below 1 are not displayed*)

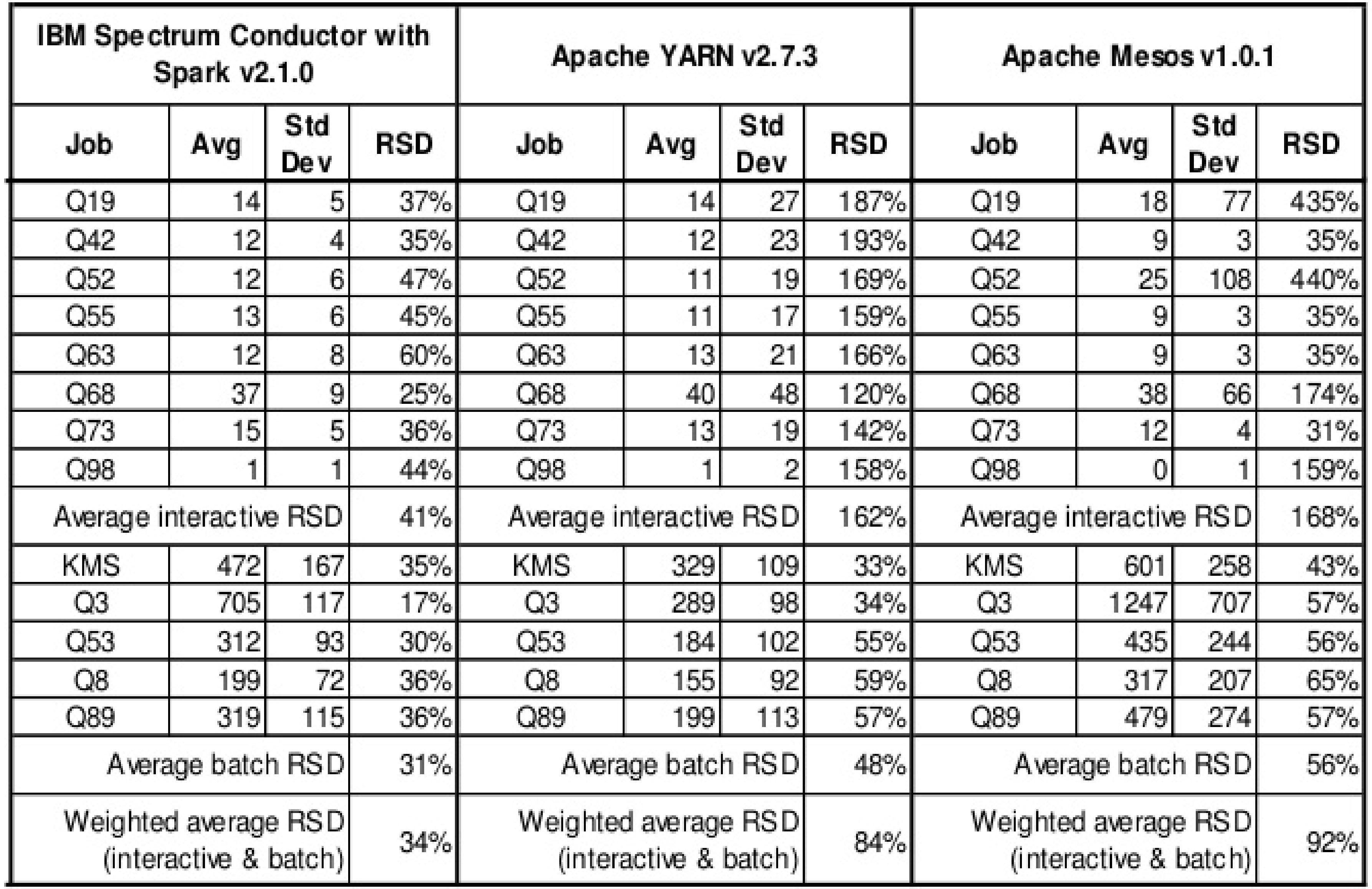
Throughput – Use Case 4



		Use Case 4: Mixed multi-tenant	
Resource Manager	Run	Duration in hours	Throughput in jobs/hour
IBM Spectrum Conductor with Spark v2.1.0	1	2.05	574
	2	1.92	612
	Worst result	*2.05*	*574*
Apache YARN v2.7.3	1	5.84	201
	2	6.63	177
	Worst result	*6.63*	*177*
Apache Mesos v1.0.1	1	2.57	458
	2	2.30	511
	Worst result	*2.57*	*458*

- IBM throughput was 224% higher than YARN and 25% higher than Mesos
- Mesos throughput was 158% higher than YARN

Variability – Use Case 4

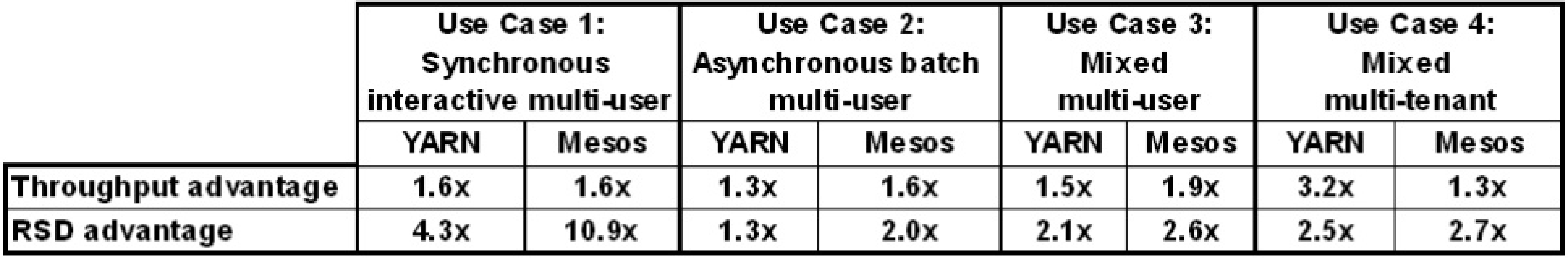


IBM Spectrum Conductor with Spark v2.1.0				Apache YARN v2.7.3				Apache Mesos v1.0.1			
Job	Avg	Std Dev	RSD	Job	Avg	Std Dev	RSD	Job	Avg	Std Dev	RSD
Q19	14	5	37%	Q19	14	27	187%	Q19	18	77	435%
Q42	12	4	35%	Q42	12	23	193%	Q42	9	3	35%
Q52	12	6	47%	Q52	11	19	169%	Q52	25	108	440%
Q55	13	6	45%	Q55	11	17	159%	Q55	9	3	35%
Q63	12	8	60%	Q63	13	21	166%	Q63	9	3	35%
Q68	37	9	25%	Q68	40	48	120%	Q68	38	66	174%
Q73	15	5	36%	Q73	13	19	142%	Q73	12	4	31%
Q98	1	1	44%	Q98	1	2	158%	Q98	0	1	159%
Average interactive RSD			41%	Average interactive RSD			162%	Average interactive RSD			168%
KMS	472	167	35%	KMS	329	109	33%	KMS	601	258	43%
Q3	705	117	17%	Q3	289	98	34%	Q3	1247	707	57%
Q53	312	93	30%	Q53	184	102	55%	Q53	435	244	56%
Q8	199	72	36%	Q8	155	92	59%	Q8	317	207	65%
Q89	319	115	36%	Q89	199	113	57%	Q89	479	274	57%
Average batch RSD			31%	Average batch RSD			48%	Average batch RSD			56%
Weighted average RSD (interactive & batch)			34%	Weighted average RSD (interactive & batch)			84%	Weighted average RSD (interactive & batch)			92%

- IBM had 2.5x lower average RSD than YARN and 2.7x lower than Mesos
- YARN had 1.1x lower average RSD than Mesos

Results Summary

Relative Advantage of IBM Spectrum Conductor for Spark
versus YARN and Mesos



	Use Case 1: Synchronous interactive multi-user		Use Case 2: Asynchronous batch multi-user		Use Case 3: Mixed multi-user		Use Case 4: Mixed multi-tenant	
	YARN	Mesos	YARN	Mesos	YARN	Mesos	YARN	Mesos
Throughput advantage	1.6x	1.6x	1.3x	1.6x	1.5x	1.9x	3.2x	1.3x
RSD advantage	4.3x	10.9x	1.3x	2.0x	2.1x	2.6x	2.5x	2.7x

IBM Spectrum Conductor with Spark v2.1.0 (as patched) / Apache YARN v2.7.3 / Apache Mesos v1.0.1

Why?

- Dynamic allocation/de-allocation
  - The resource manager should allow applications to both release and gain resources during execution, based on scheduling demand from the Spark driver
  - IBM Spectrum Conductor with Spark supports both dynamic allocation and dynamic de-allocation
  - YARN supports dynamic allocation but appeared to have issues with de-allocation (or re-allocation)
  - Mesos supports both dynamic allocation and dynamic de-allocation but appeared to have issues with the offer mechanism and long-running jobs
- Pre-emption
  - Pre-emption allows for more even resource distribution among applications, higher throughput and lower variation in performance
  - IBM Spectrum Conductor with Spark includes a highly configurable and intelligent pre-emption implementation
  - YARN supports pre-emption but it hurt rather than helped when we tried it
  - The version of Mesos we used did not support pre-emption

Next steps

- Please provide feedback/contribute to the SMB benchmarks
- Please use the SMB benchmarks and share what you find

References

Spark Multi-User Benchmark (SMB):
www.github.com/IBM/SparkMultiuserBenchmark

STAC Report:
www.STACresearch.com/news/2017/05/19/IBM170405

SMB-2 Blog:
https://developer.ibm.com/code/2017/05/24/introducing-spark-multiuser-benchmark-version-2/

Results Blog:
https://www.ibm.com/developerworks/community/blogs/281605c9-7369-46dc-ad03-70d9ad377480/entry/Understanding_Multiuser_Spark_Application_Performance_Differences?lang=en

Spark-related materials at STAC:
www.STACresearch.com/spark

Q&A

SPARK SUMMIT 2017

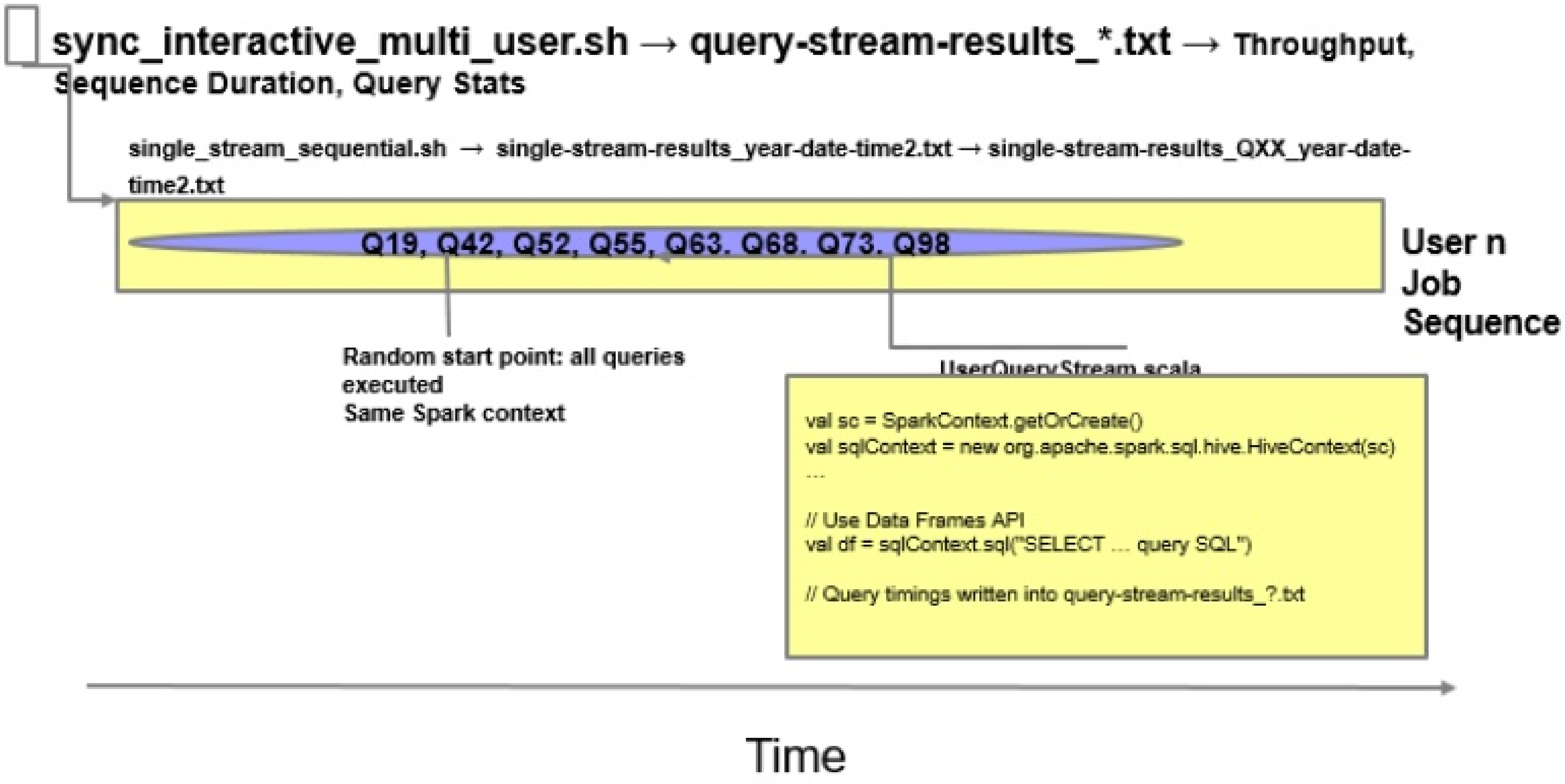
Thank You.

Mikhail Genkin: genkin@ca.ibm.com

Peter Lankford: peter.lankford@STACresearch.com

Back-Up

SMB-2 Sync-Interactive Workload Implementation



sync_interactive_multi_user.sh → query-stream-results_*.txt → Throughput, Sequence Duration, Query Stats

single_stream_sequential.sh → single-stream-results_year-date-time2.txt → single-stream-results_QXX_year-date-time2.txt

Random start point: all queries executed
Same Spark context

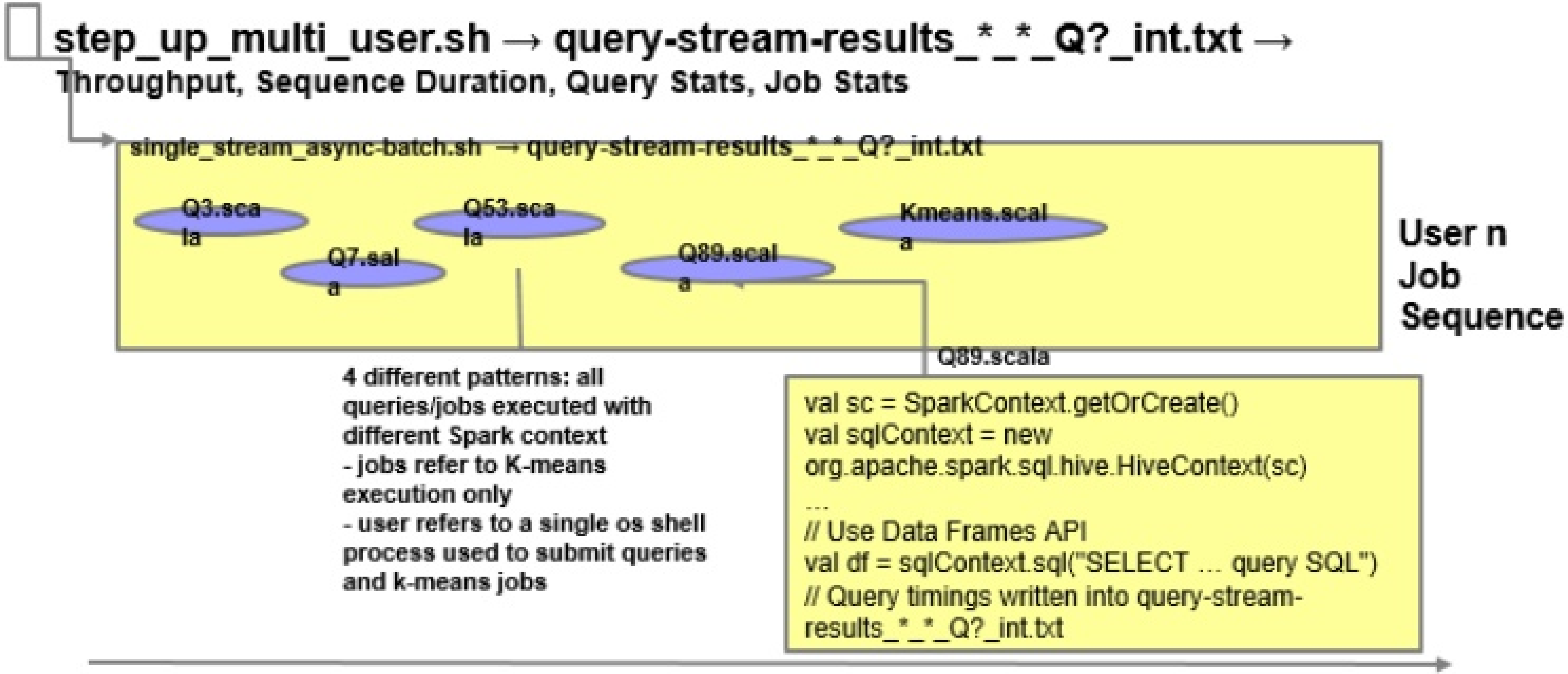
UserQueryStream.scala

```
val sc = SparkContext.getOrCreate()
val sqlContext = new org.apache.spark.sql.hive.HiveContext(sc)
....

// Use Data Frames API
val df = sqlContext.sql("SELECT .... query SQL")

// Query timings written into query-stream-results_?.txt
```

SMB-2 Asynchronous-Batch Workload Implementation



step_up_multi_user.sh → query-stream-results_*_*_Q?_int.txt → Throughput, Sequence Duration, Query Stats, Job Stats

single_stream_async-batch.sh → query-stream-results_*_*_Q?_int.txt

4 different patterns: all queries/jobs executed with different Spark context
- jobs refer to K-means execution only
- user refers to a single os shell process used to submit queries and k-means jobs

```
val sc = SparkContext.getOrCreate()
val sqlContext = new
org.apache.spark.sql.hive.HiveContext(sc)
...
// Use Data Frames API
val df = sqlContext.sql("SELECT … query SQL")
// Query timings written into query-stream-
results_*_*_Q?_int.txt
```

What was STAC’s role?

- Provide feedback on the benchmark specs
  - Note: This was not a STAC Benchmark™
- Confirm conformance of tests to the specs
- Inspect the system configurations
- Oversee the test execution
- Analyze the results
- Produce the STAC Report™ and detailed STAC Configuration Disclosure

Limitations (as per STAC Report)

- Sensitivity to tuning
- Sensitivity to workload
- Difficulty of configuring Use Case 4
- Mesos/YARN/Spark configuration puzzles
- Others