cloudera

Scaling SQL-on-Hadoop for BI

Yanpei Chen

Alan Choi, Dileep Kumar, David Rorke, Silvius Rus, Justin Erickson

SQL-on-Hadoop

HIVE

MapReduce

OR

Spark

Spark SQL

SQL Analytics and BI

Batch Processing

Spark developers

cloudera

Business Intelligence (BI)

- Data discovery
  - Need flexibility to have more data readily available without upfront modeling
  - Need flexibility to analyze existing data sets in different ways
- Dashboards
  - Need to unify different data sources
  - Need finer-grain details in source data vs. planned summarized extracts
- Reporting
  - Familiar workload from established relational databases

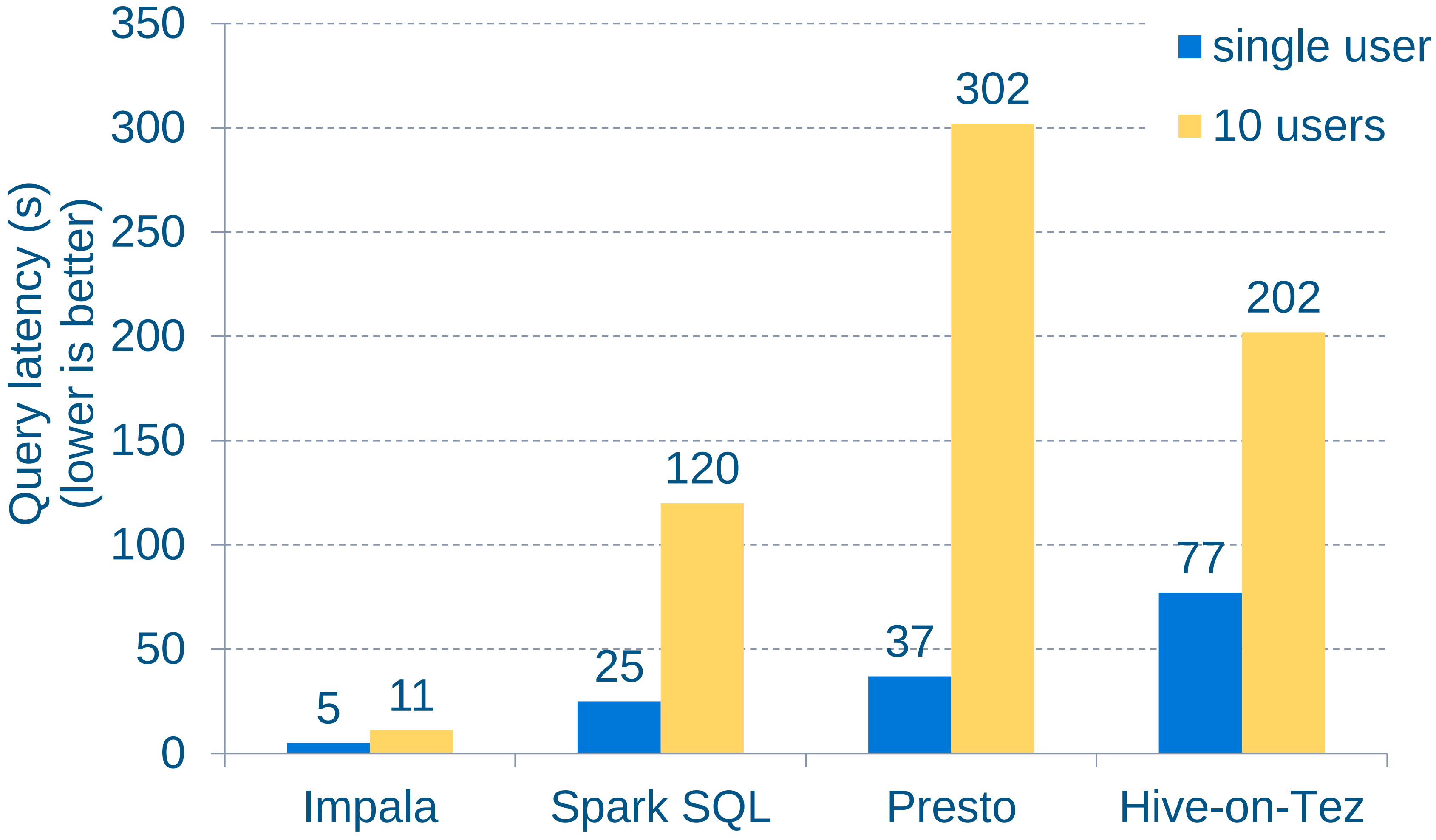
What BI needs from SQL-on-Hadoop

- Integration with existing BI tools
- Operate on native Hadoop data, no costly extract-transform-load (ETL)
  - Common native file formats
  - Common security
  - Common data management
  - Common resource management
- Interactive query latency for many users

What it means to “support more users”

- Can SQL-on-Hadoop maintain interactive latency as we add users?
- How many users can a particular cluster support?
- What is the system throughput?
- Can you “just add more machines” to increase the throughput and the number of users supported?

Prior results do not address these questions



- Sept. 2014
- Impala 1.4.0
- Spark SQL 1.1
- Presto 0.74
- Hive-on-Tez Stinger final phase
- From Cloudera Developer blog –

  [New Benchmarks for SQL-on-Hadoop: Impala 1.4 Widens the Performance Gap]

cloudera

Focus of this talk – Impala

Scaling is a multidimensional problem

Scale is measured by

- Query latency
- Query throughput
- Users supported
- Cluster size
- Hardware utilization

Desired scaling behavior

Efficient system

- Query throughput maximizes at a saturation point where some cluster hardware resource is fully utilized.

High performance

- At saturation point, query latency is low and throughput is high.

Scalable in the number of users

- Adding users after saturation leads to constant throughput and proportionally increasing latency.

Scalable in cluster size

- Adding hardware to the system leads to proportionally increasing saturation throughput and the number of users supported at a given latency.

cloudera

How do we tackle this complexity?

- Goal: Investigate performance across 5, 10, 20, 40, 80 nodes
- Cluster: 80 nodes
- Workload: TPC-DS
  - Heavy reporting and modeling, not good match with real workload
  - Open and commonly known
  - Hence, hand-pick a few queries that more closely match BI

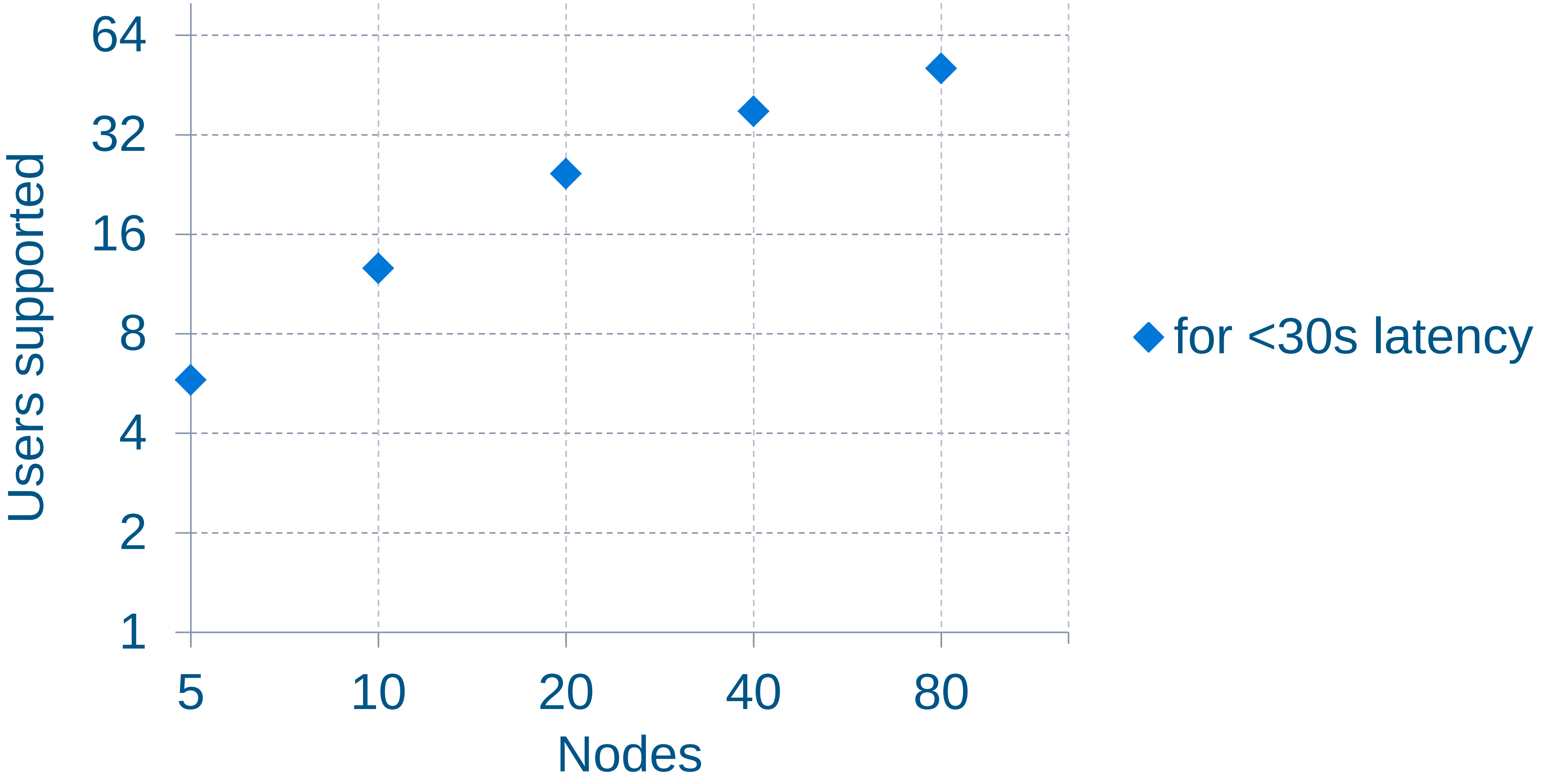
cloudera

Details: Test setup

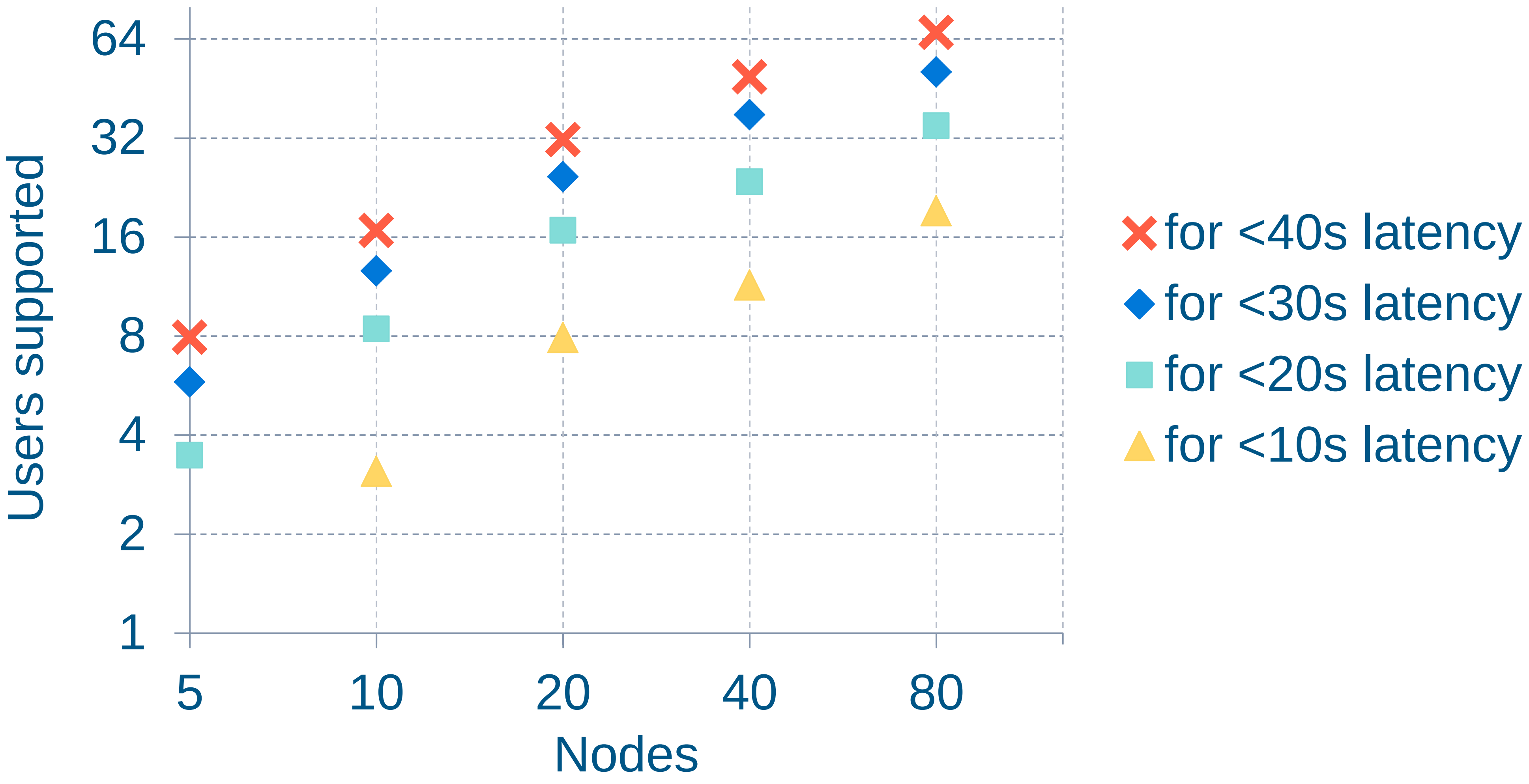
- 80 nodes, each 2 socket Intel Xeon E5-2630L 2.00GHz, 64GB RAM, 12x 2TB disks, 10Gbps Ethernet
- CDH5.3.3 with Impala 2.1.3, parquet + snappy data format, 50GB mem limit
- TPC-DS at 15000 scale factor (15TB), queries 19, 42, 52, 55, 63, 68, 73, 98
  - Runs a number of concurrent query streams
  - Each query stream models a concurrent user
  - Each query stream runs all queries, in different randomized order
  - No pause between queries
  - 80.5 GB data touched per query on average after partition pruning
  - THESE ARE BIG QUERIES AND NOT INFLATED NUMBERS!!!

Results …

Users supported

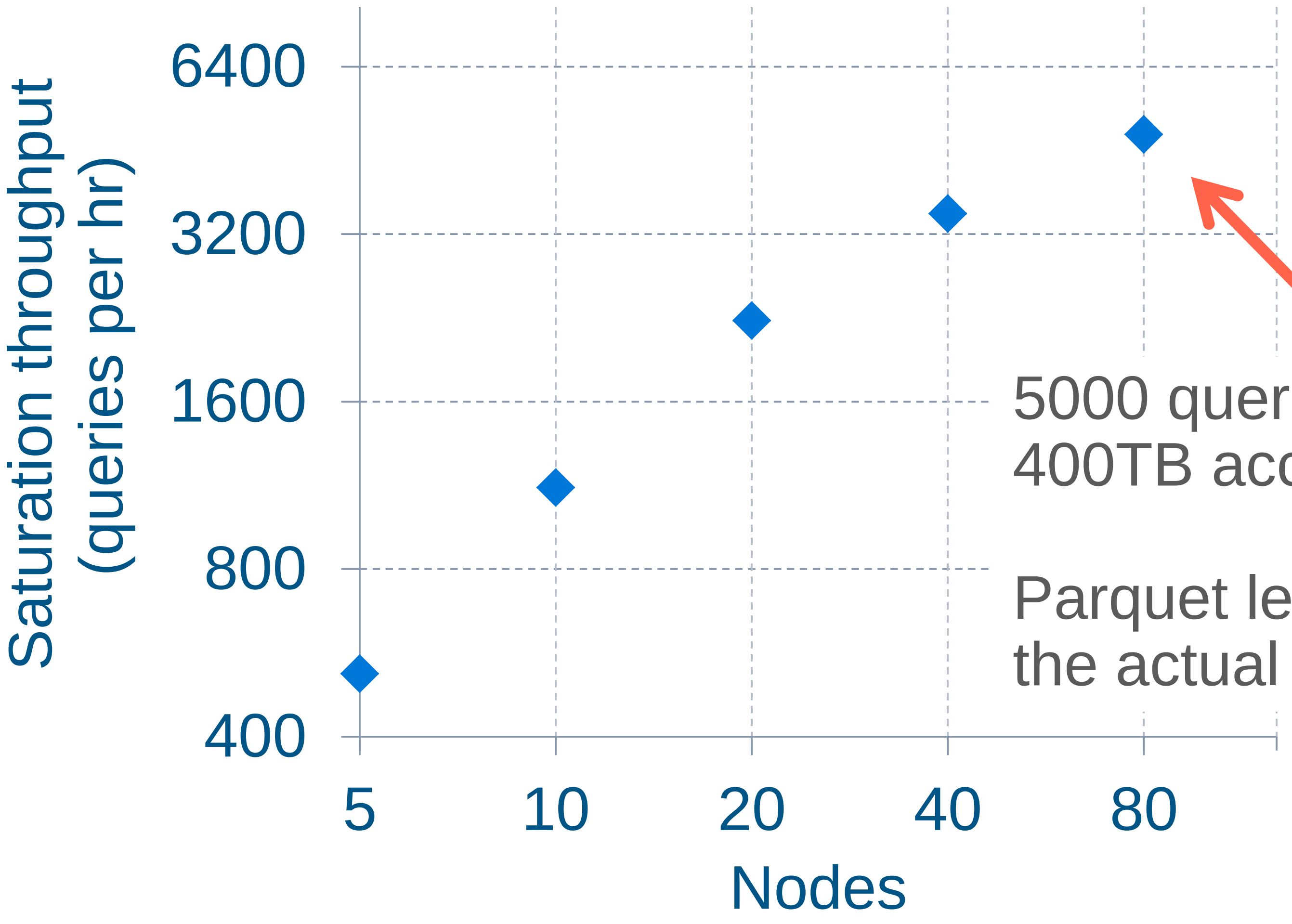


Users supported



cloudera

Cluster saturation throughput



5000 queries per hr =
400TB accessed per hr!!!

Parquet lets us scan only
the actual columns needed

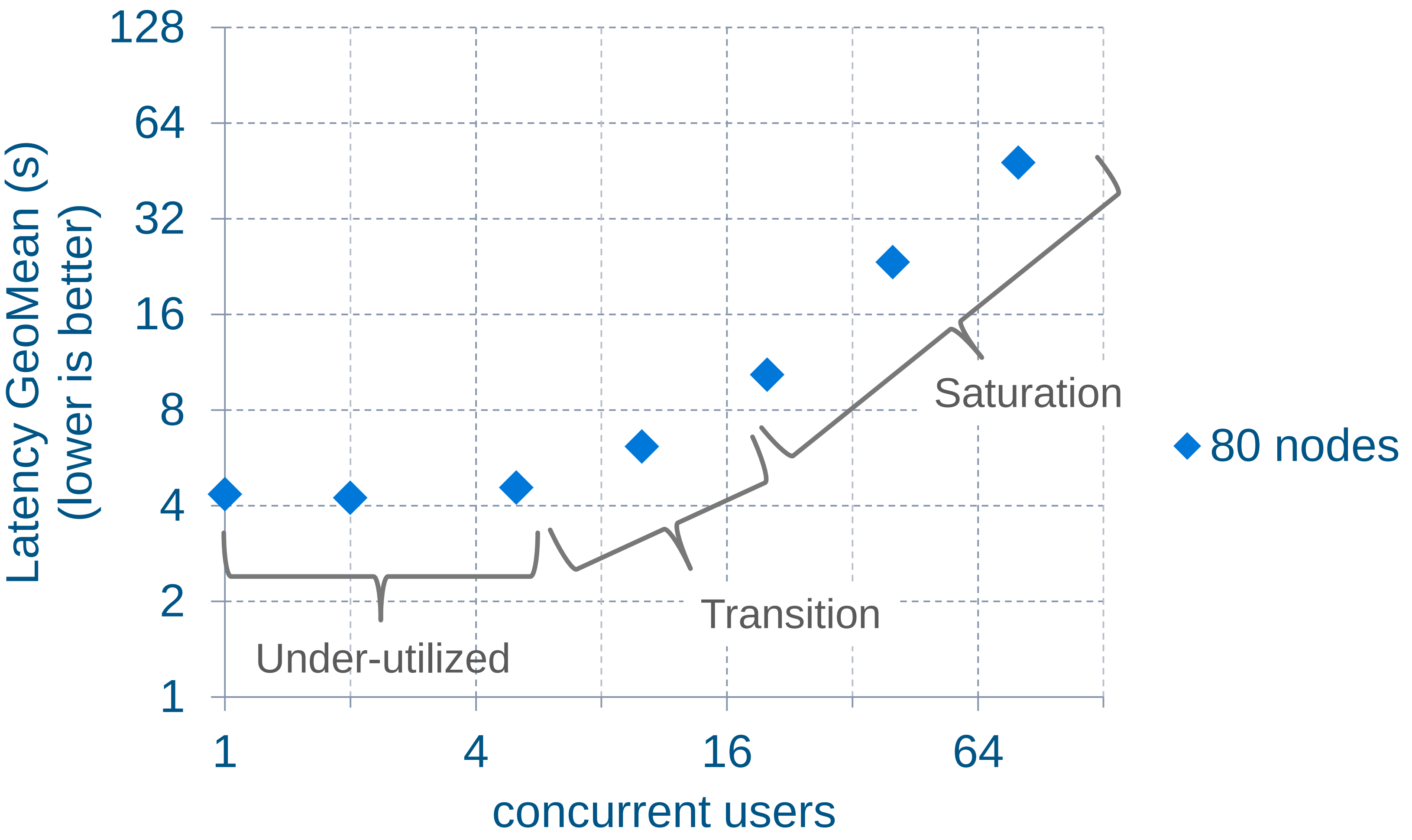
cloudera

Highlight result

To maintain latency (SLA) while adding users, just add more nodes!

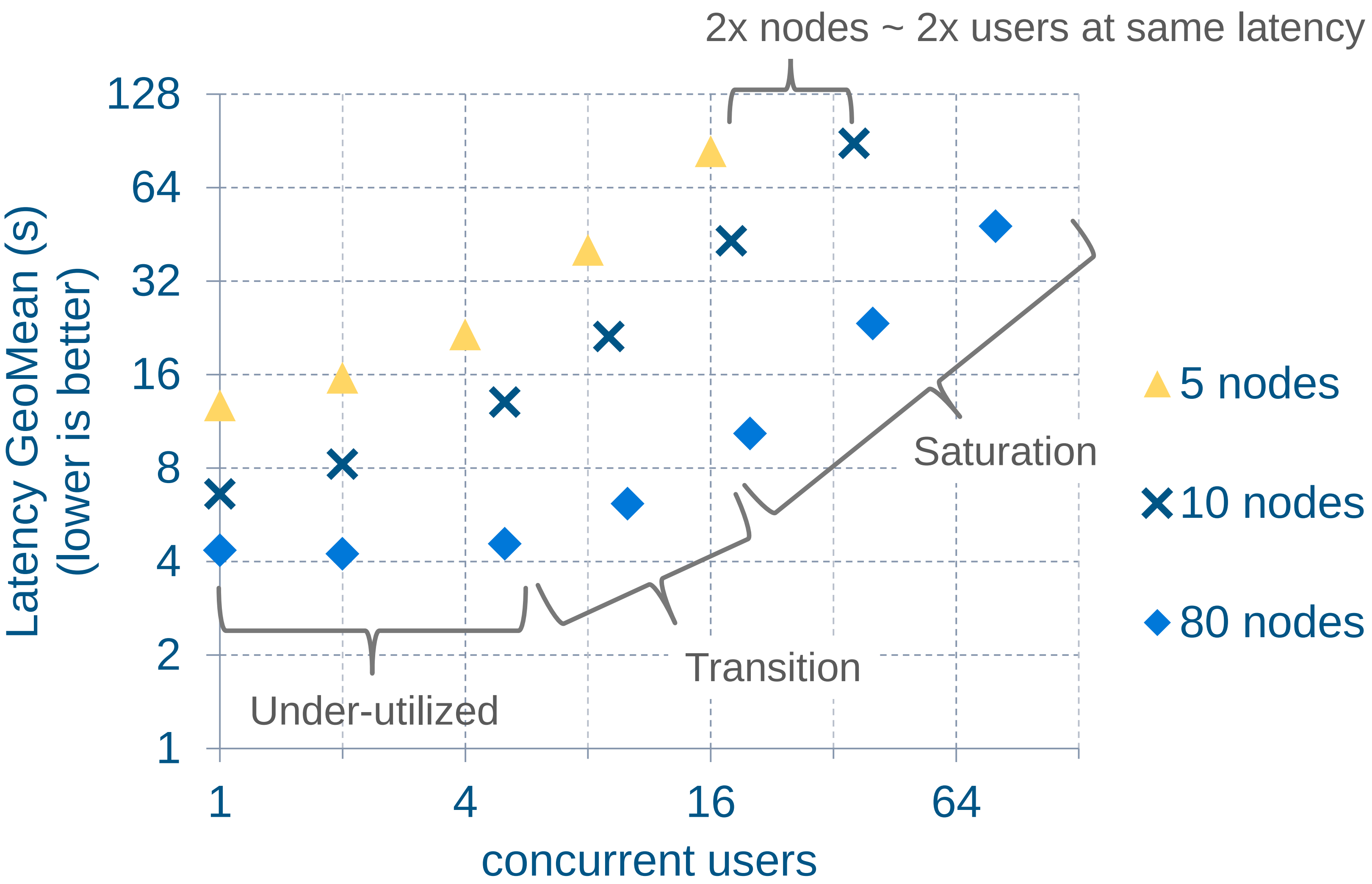
Scalable in cluster size and number of users! Yay!!!

Details: Cluster behavior as we add users

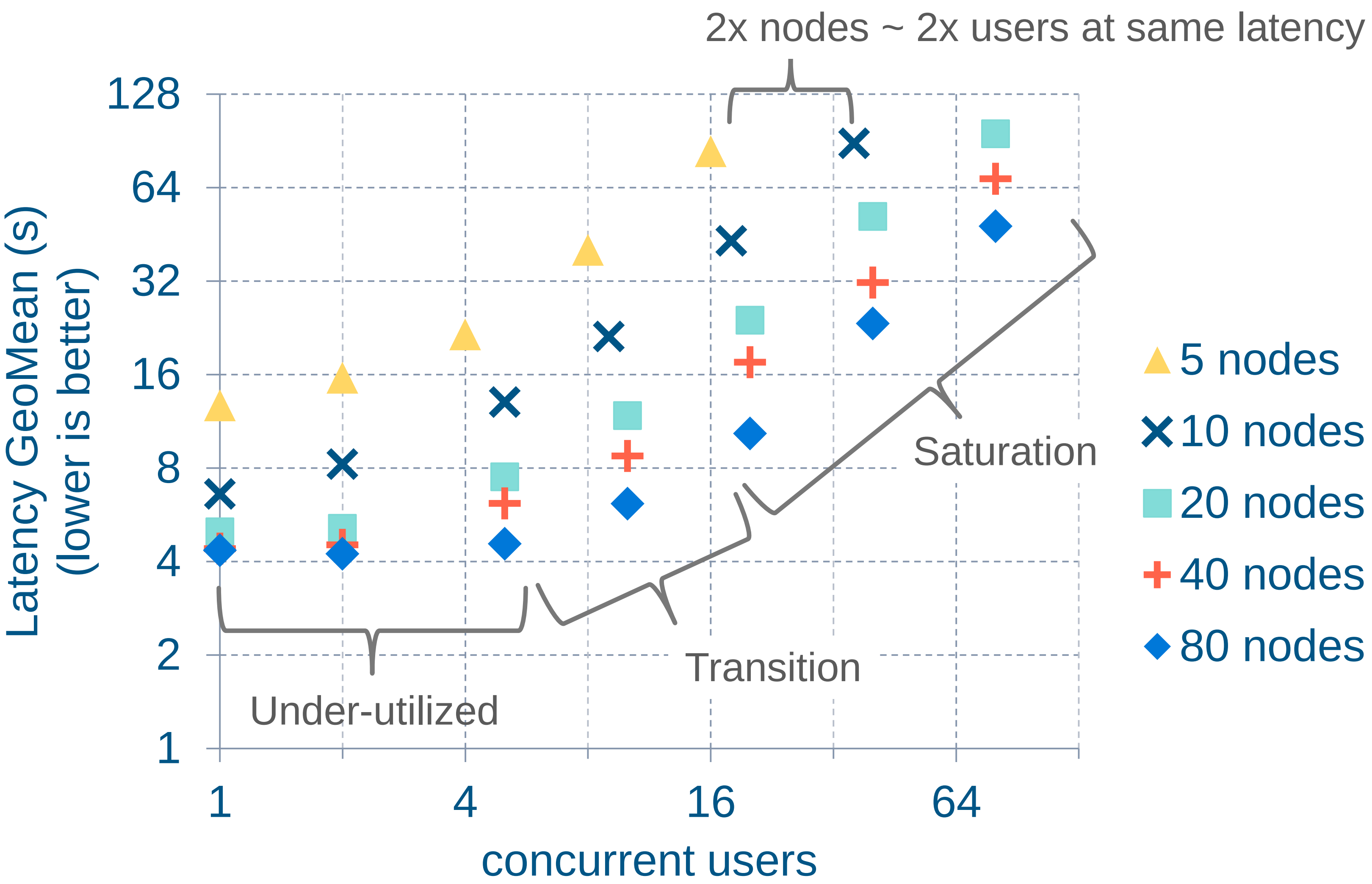


cloudera

Details: Cluster behavior as we add users

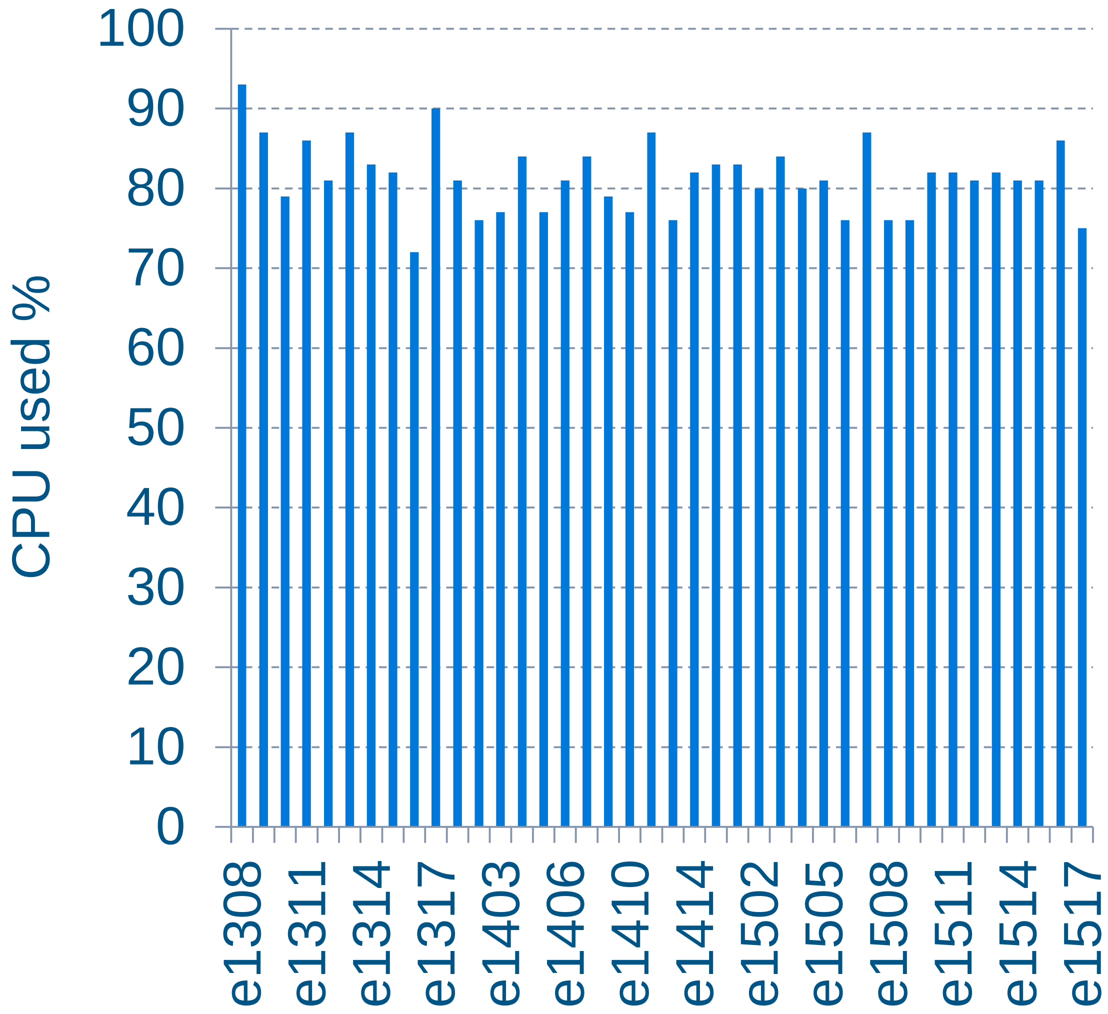


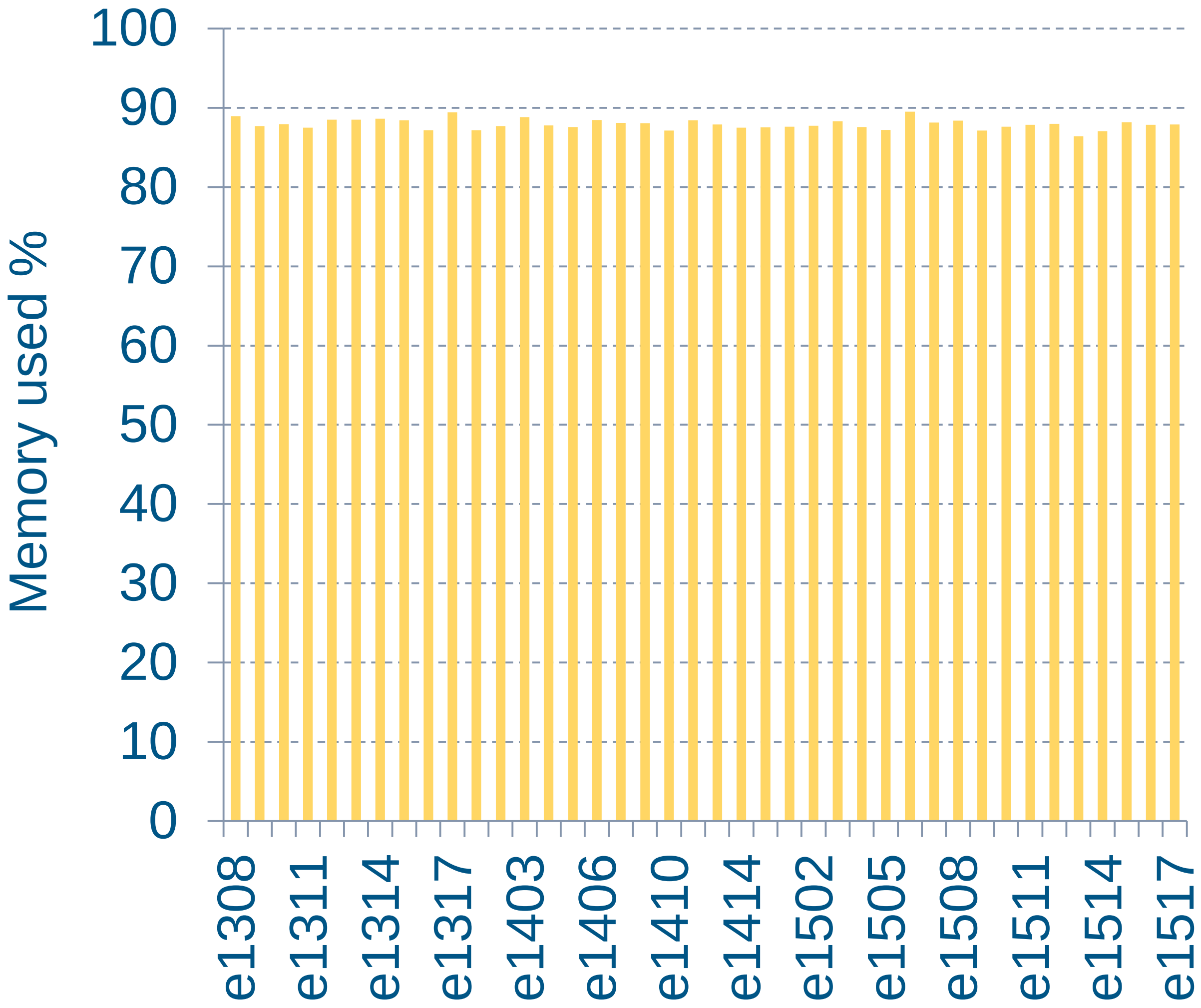
Details: Cluster behavior as we add users



cloudera

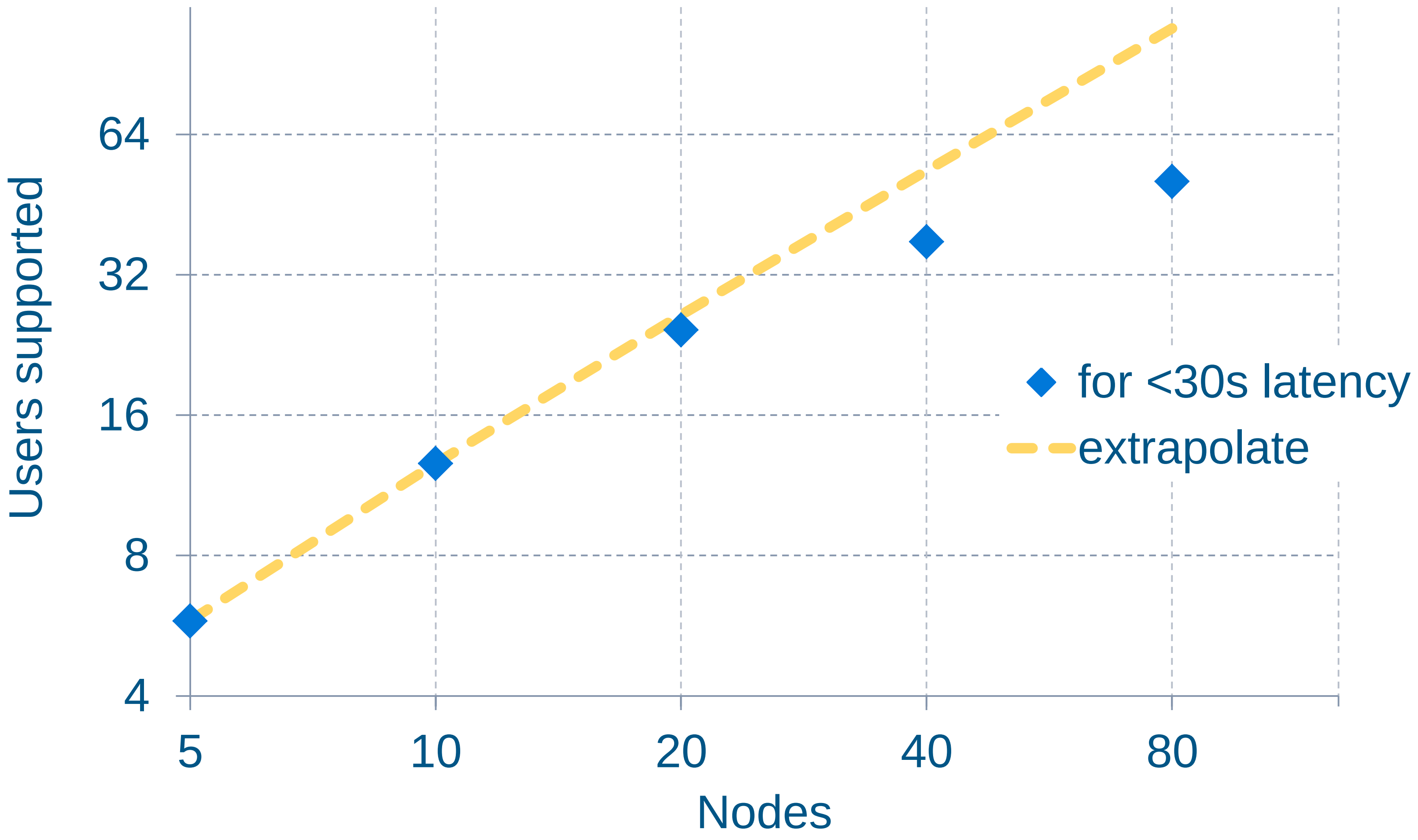
Details: Cluster is CPU and memory bound





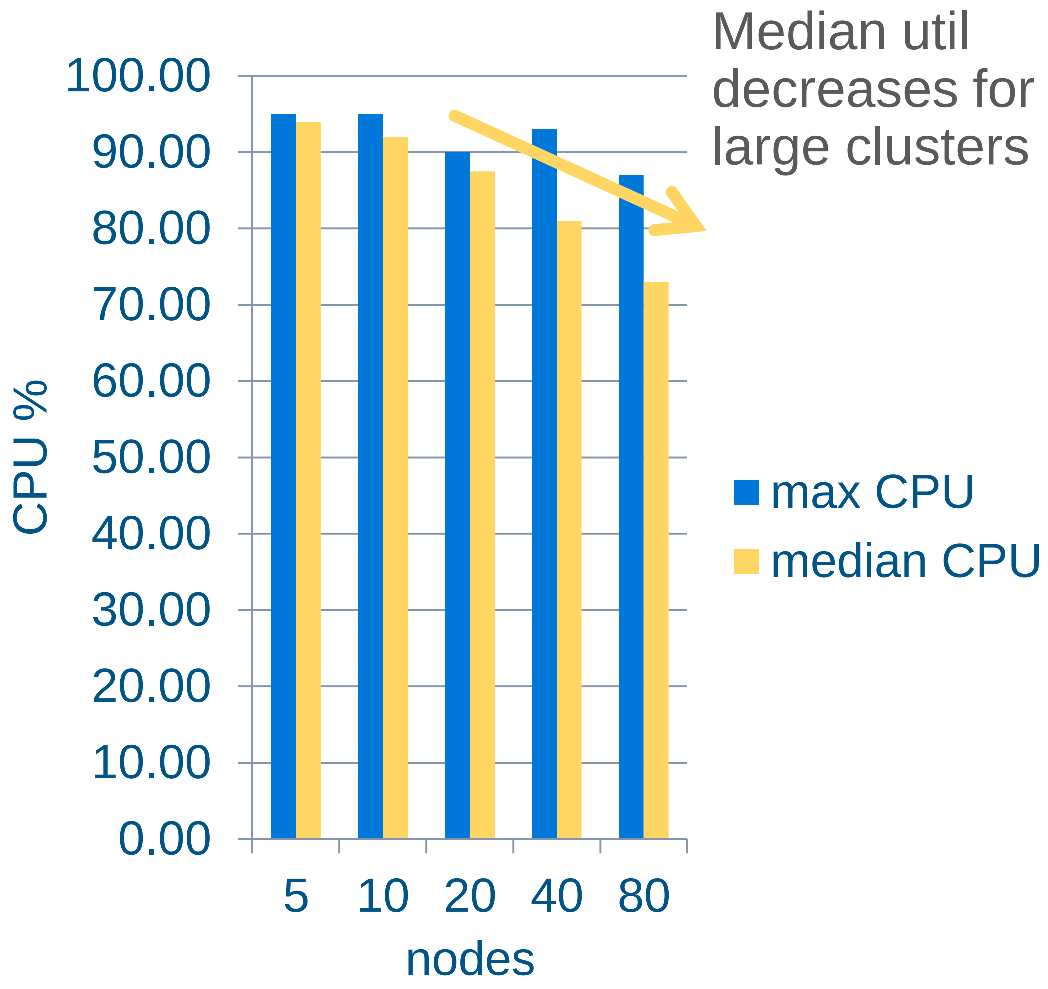
cloudera

Details: There is scaling overhead ☹



Details: Scaling overhead from workload skew

- Some partitions have more data than others
- Small cluster – more partitions per node, variations even out
- Large cluster – fewer partitions per node, waiting on slowest node
- Shows up as large CPU variations across nodes for large clusters



Details: Need to set admission ctrl

- Keeps cluster in efficient (not overwhelmed) state
- To configure:
  - Run your typical queries in stand-alone fashion
  - Find per-node peak memory use from query profile
  - Set admission control threshold to
    (RAM size – headroom for OS etc.) /
    (max per-node peak memory use across all queries) ×
    (safety factor < 1)
- This is a super conservative limit!!!

cloudera

Recap

To maintain latency (SLA) while adding users, just add more nodes!

Large clusters saturates at more users

Hardware is CPU and memory bound

Plan for scaling overhead from inherent workload skew

cloudera

Stepping back – What does this all mean?

Stepping back: SQL-on-Hadoop design

To fellow software/hardware engineers working on SQL-on-Hadoop:

- Concurrent query performance should receive additional focus
- Admission control an important design point
- SQL-on-Hadoop should prioritize CPU efficiency
- Hardware needs both memory and CPU capacity

Stepping back: Product implications

To customers running SQL-on-Hadoop: Select solution that

- Scales better to large clusters
- Achieves fast, interactive latency
- Makes efficient use of hardware - both CPU and memory
- Simplifies planning: add more users → add more nodes

cloudera

Thank you.

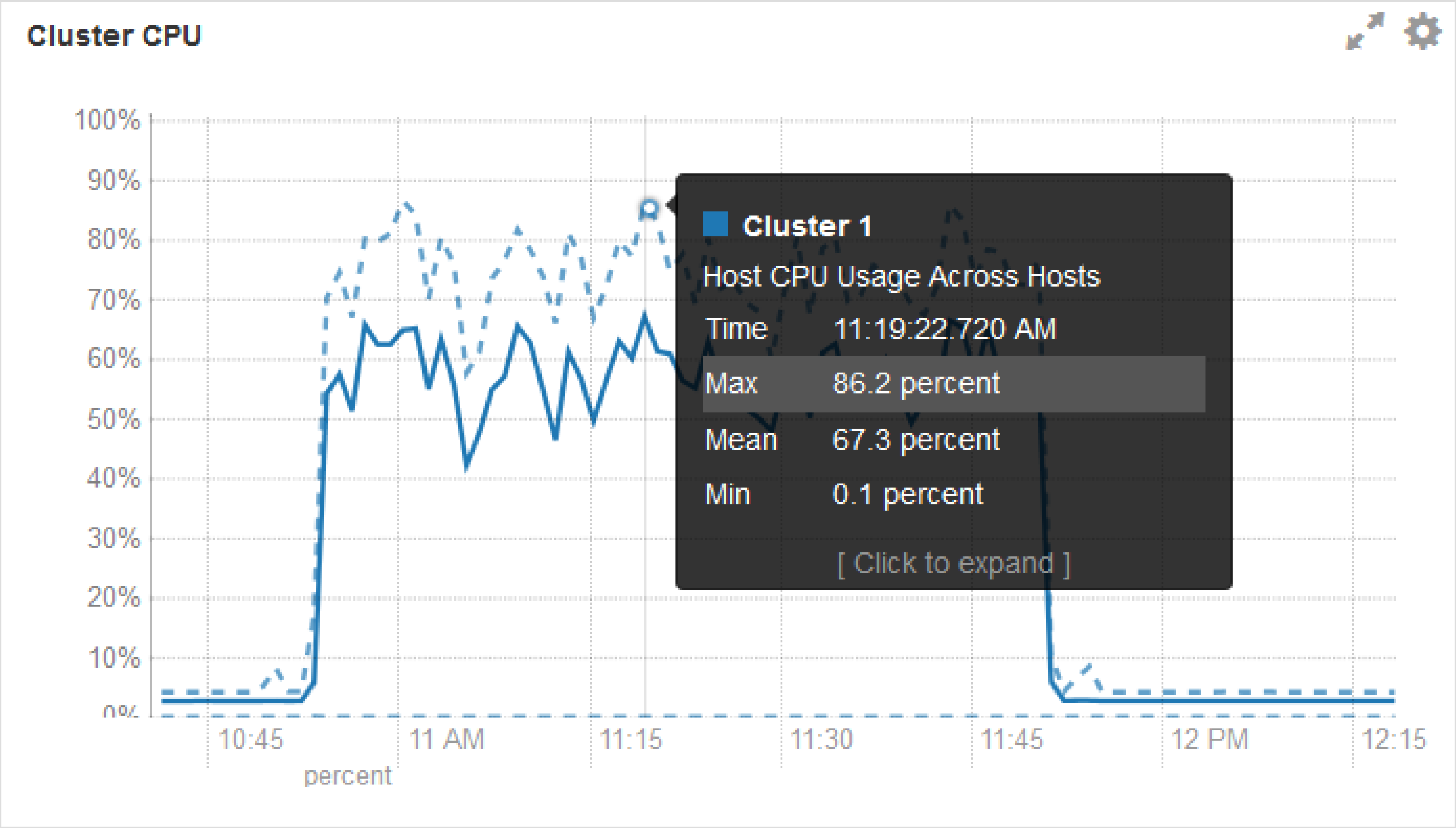
Digging deeper: User think time model

- Typical users run query, then pause to think
- Typical cluster has active and inactive users
- Tests so far assume all users are active and have no think time
- Adding think time + inactive users inflates “users supported”

- Verified that within some bounds, the inflated number is mathematically predictable

Details: Identifying skew – CM graphs

- Mean CPU far below max CPU for the cluster



cloudera

Details: Identifying skew – Query profiles

- Execution Summary shows avg time far below max time

Operator	#Hosts	Avg Time	Max Time	#Rows	Est. #Rows	Peak Mem	Est. Peak Mem	Detail
20:MERGING-EXCHANGE	1	2.921ms	2.921ms	301	100	0	-1.00 B	UNPARTITIONED
12:TOP-N	74	456.852us	9.721ms	301	100	12.00 KB	7.37 KB	
19:AGGREGATE	74	770.166ms	1s654ms	301	1.31B	6.55 MB	10.00 MB	FINALIZE
18:EXCHANGE	74	7.836ms	503.799ms	22.27K	1.31B	0	0	HASH(i_brand,i_brand_id,i_m...
11:AGGREGATE	74	1s038ms	2s641ms	22.27K	1.31B	13.93 MB	101.24 GB	
10:HASH JOIN	74	102.667ms	822.479ms	23.29M	1.31B	9.82 MB	0	INNER JOIN, BROADCAST
|--17:EXCHANGE	74	30.564us	277.386us	30	0	0	0	BROADCAST
| 05:SCAN HDFS	1	61.256ms	61.256ms	30	0	1.20 MB	48.00 MB	tpcds_15000_parquet.date_dim
09:HASH JOIN	74	149.101ms	673.251ms	23.29M	1.31B	8.94 MB	1.66 KB	INNER JOIN, BROADCAST
|--16:EXCHANGE	74	162.986us	2.701ms	62	62	0	0	BROADCAST
| 04:SCAN HDFS	1	8.678ms	8.678ms	62	62	62.00 KB	32.00 MB	tpcds_15000_parquet.store
08:HASH JOIN	74	1s321ms	3s869ms	23.55M	1.31B	838.04 MB	50.18 MB	INNER JOIN, BROADCAST
|--15:EXCHANGE	74	266.785ms	3s586ms	1.93M	1.93M	0	0	BROADCAST
| 03:SCAN HDFS	68	16.474ms	101.874ms	1.93M	1.93M	2.66 MB	32.00 MB	tpcds_15000_parquet.custome...
07:HASH JOIN	74	2s612ms	5s603ms	23.55M	1.31B	870.03 MB	32.31 MB	INNER JOIN, BROADCAST
|--14:EXCHANGE	74	205.152ms	348.638ms	3.85M	3.85M	0	0	BROADCAST
| 02:SCAN HDFS	60	23.557ms	372.937ms	3.85M	3.85M	5.88 MB	32.00 MB	tpcds_15000_parquet.customer
06:HASH JOIN	74	211.445ms	354.469ms	24.09M	1.31B	6.04 MB	24.71 KB	INNER JOIN, BROADCAST
|--13:EXCHANGE	74	51.653us	439.104us	591	305	0	0	BROADCAST
| 01:SCAN HDFS	1	129.547ms	129.547ms	591	305	1.32 MB	96.00 MB	tpcds_15000_parquet.item
00:SCAN HDFS	74	8s671ms	11s711ms	198.14M	1.31B	84.49 MB	352.00 MB	tpcds_15000_parquet.store_s...

Digging deeper: Complex queries

- Mismatch between TPC-DS and real-life BI workloads
- BI: Dashboards, data discovery
- TPC-DS: Heavy reporting, modeling

- Verified added query complexity gives similar qualitative behavior