

Spark SQL: Another 16x faster after Tungsten

SPARC processor has dramatic advantages over x86 on Apache Spark

Brad Carlile
Sr. Director
Solution Architecture Engineering SAE
Room 311, Wednesday, Feb 8, 2:40 pm–2:55 pm
February 7, 2017

Additional info on Proof points:
<http://blogs.oracle.com/bestperf>



Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

SPARC DAX in Apache Spark is Proof-of-concept and not product

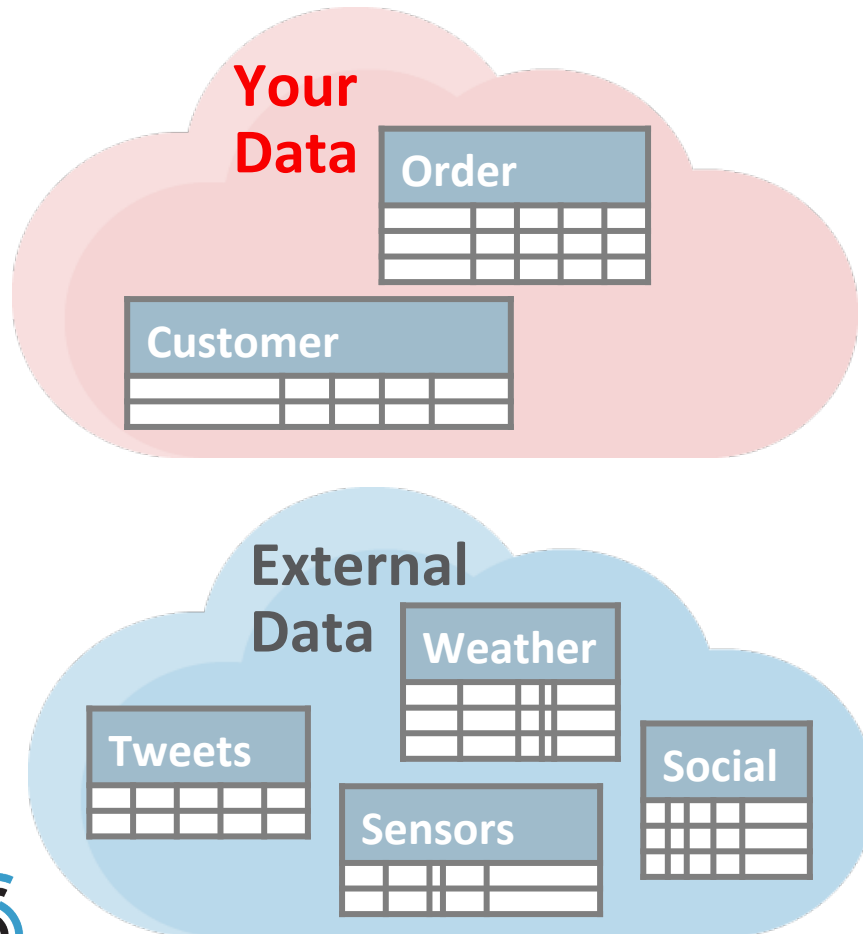
Spark is Exciting – *Lots of software innovation*

Spark's Whole Ecosystem **totally appeals to the performance expert in me**

- Great Ecosystem for Analytics
 - Spark forms a continuum with other technologies (Kafka, Solr, ...)
 - Lots of Oracle's customer using Spark
 - Spark is fast & scalable because of clean design
 - Spark's in-memory focus
 - Spark speaks SQL & DataFrames
 - Perfect marriage for Data Scientists & **Hardware Acceleration**
- *...but what is the status of processor performance?*

Modern Analytics is about Your Data & External Data

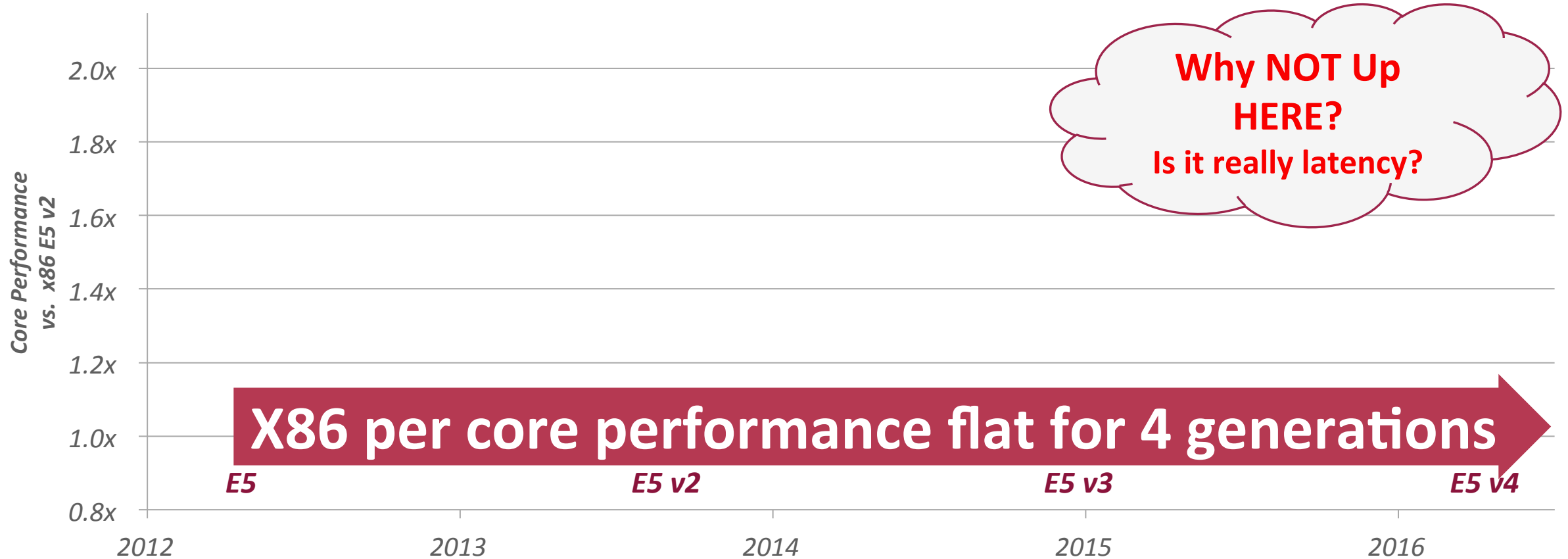
Better information when you can analyze all relevant data



- Analytics is now your data **PLUS** external data
 - Oracle Database In-Memory Fastest on SPARC
 - Continuously analyzing same data in different ways – **Fastest if store data in-memory**
- Big Data can be used to describe external data
 - Social networks, public data: sentiments, weather, traffic, events, ratings, trends, IoT sensors, behaviors,...
 - Lots of ETL/filtering needed to find useful data
 - Big Data SQL

IS THERE COMPUTE BOTTLENECK? REALLY?

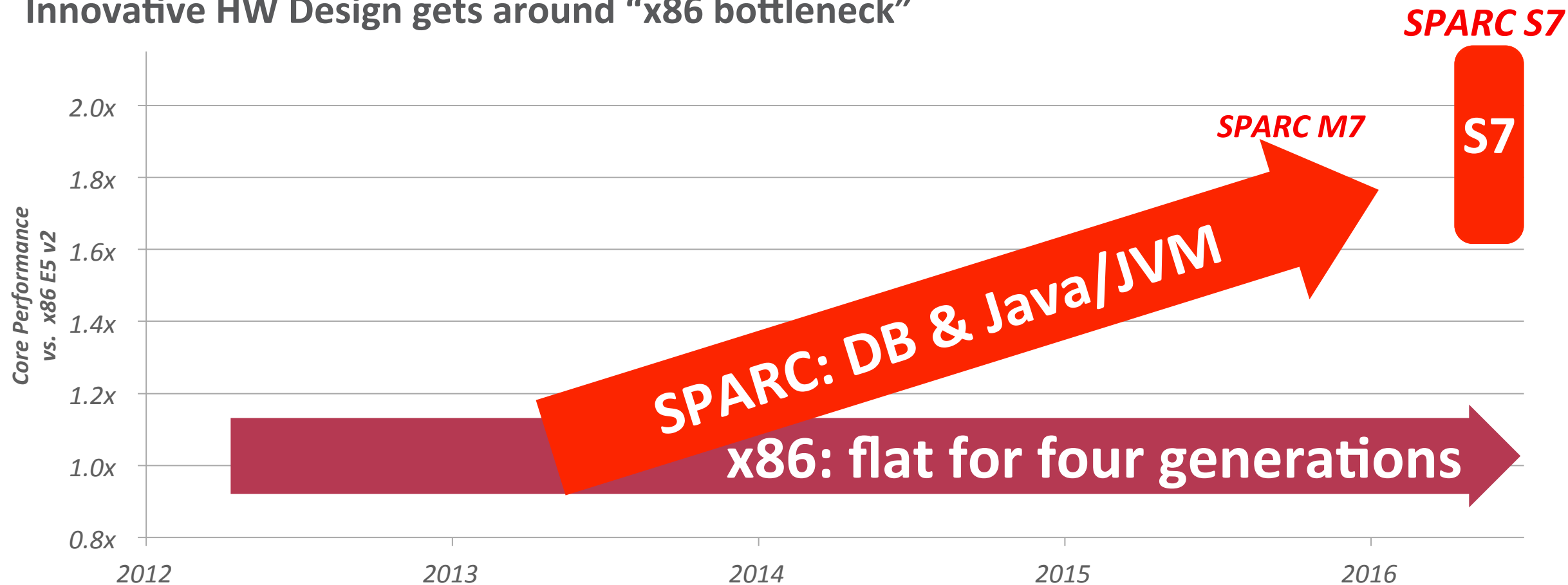
x86 Innovations in Per Core Performance has Stalled



MIT Tech Review, "Chip maker Intel has signaled a slowing of Moore's Law, ... which played a role in just about every major advance in engineering and technology for decades."

SPARC's Innovations are Continually Increasing Core Performance

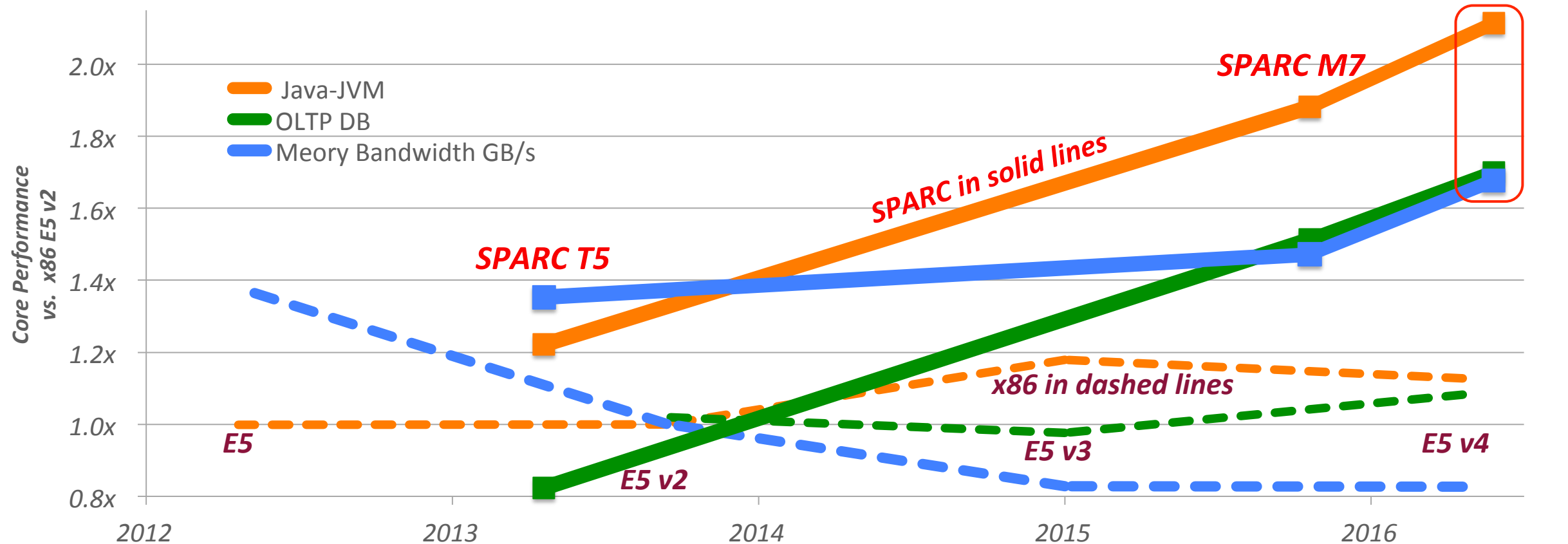
Innovative HW Design gets around "x86 bottleneck"



- SPARC has faster cores for Database, Java, Apps, ...
- SPARC has more cores per chip which also drives efficiency

SPARC S7 is 1.6x to 2.1x faster than x86

Innovative HW Design gets around “x86 bottleneck”, x86 per core perf stalled



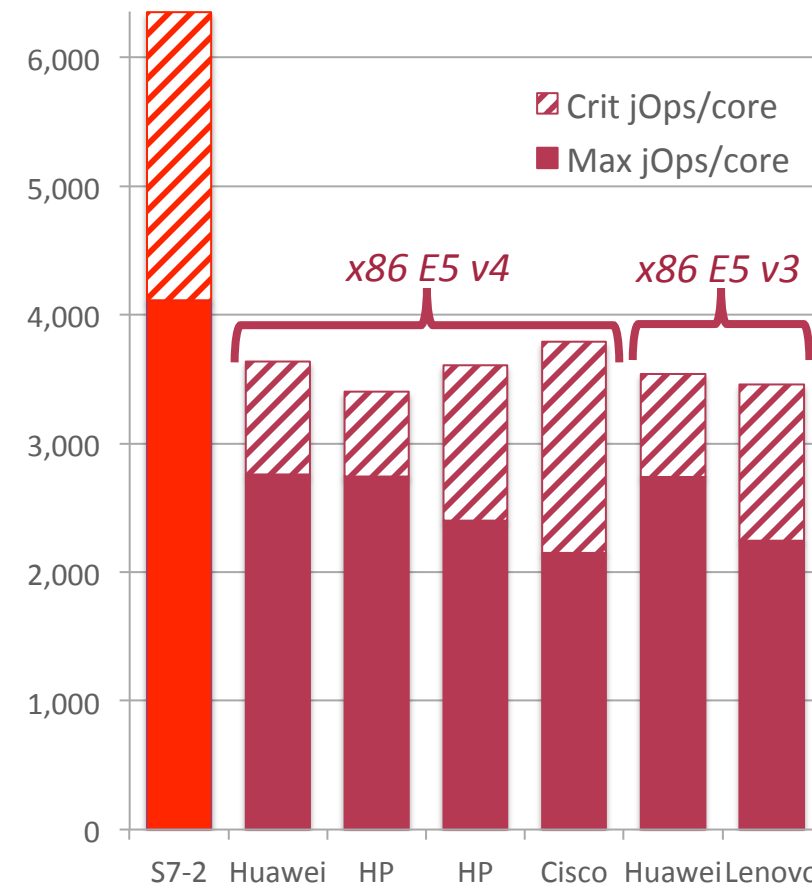
It's more important how you use transistors,
than the number transistors you make (Moore's Law)

SPECjbb2015 MultiJVM SPARC S7

SPARC S7 core 1.5x to 1.9x faster than x86 E5 v4 core on Max jOps/core

	Processor	chip, core	Max jOPS	Crit jOPS
➔ SPARC S7-2	4.27GHz SPARC S7	2,16	65,790	35,812
Huawei RH2288Hv3	2.2GHz E5 v4 22c	2,44	121,381	38,595
Cisco C220	2.2GHz E5 v4 22c	2,44	94,667	71,951
Huawei RH2288Hv3	2.3GHz E5 v3 18c	2,36	98,673	28,824
Lenovo x240 M5	2.3GHz E5 v3 18c	2,36	80,889	43,654

- Innovations in SPARC processor innovations improve Java and JVM performance
 - x86 performance is *flat* generation-to-generation
 - Trade off between tuning for Max jOps or Crit jOps on x86



Java(JVM) is the Language of the Cloud & FOSS

Java & the underlying JVM is the design target for SPARC



Big Data & DevOps	Description	Written to
Oracle Fusion	Enterprise Apps	Java
Apache Spark	In-memory Analytics	JVM
Apache Cassandra	NoSQL database	Java
Apache Hadoop	Disk to disk Map Reduce	Java
Apache HDFS	filesystem	Java
Apache Lucene	Doc text Indexing	Java
Apache Solr	text/doc/web search	Java
Jersey/Grizzly (REST)	REST Interface	Java
Apache Hive	SQL on HDFS	Java
Neo4j	Graph	Java
Scala	Language	JVM

Big Data & DevOps	Description	Written to
Akka (REST)	REST for Big Data	JVM
Apache Accumulo	key/value store	Java
Apache Flink	Batch &Stream Processing	Java(JVM)
Apache Kafka	Streaming Message Broker	JVM
Apache log4j	Log event manager	Java
Apache Samza	Stream processing	JVM
Apache Storm	Stream processing	Java & Clojure
Apache Yarn	Cluster job manager	Java
Apache Zookeeper	Config & group services	Java
H2O.ai	ML Apps libraries	Java, Python,R
Apache Hbase	NoSQL database	Java

JVM = Java Virtual Machine - Runtime

Cassandra NoSQL 2.2.6 (Yahoo Cloud Serving Benchmark)

SPARC core is 2.0x faster than E5 v3 core



Cassandra	Chip, core	Processor	Ave Ops/s	K Ops/s per core	SPARC per core Advantage
➡ SPARC S7-2	2,16	4.27 SPARC S7	69,858	4.4K	2.0x
E5 v3 Haswell	2,36	2.3 E5 v3	90,184	2.5K	1.0x

- SPARC S7 *2.0x faster per core* than E5 v3 Haswell
- SPARC's Java/JVM advantages benefit Cassandra performance
 - YCSB – Yahoo Cloud Serving Benchmark – 300M
 - Mixed Load Workload A (50% Read/50% Update)
 - More info at: <http://blogs.oracle.com/bestperf>

Oracle Database also Optimized for Analytic Queries

Apache Spark needs to implement these optimizations!

SQL optimized for analytic queries

- Column Store is optimal for Analytics
 - Analytics Looks across transactions at specific columns
 - Database columns are Machine Learning features
- Columns far more efficient when analyzing features (efficient bandwidth & CPU needs)
- Can exploit data characteristics >20x benefit
 - Storage optimizations – data repeatedly analyzed
 - Vector processing of Dictionary-encoded compares needs to be added to in Spark persisting DataFrames in memory
 - layered compression means 10's TB of data fit into TB's of memory
 - Processing optimizations
 - Bloom Filter join processing, operator pushdown, metadata optimize

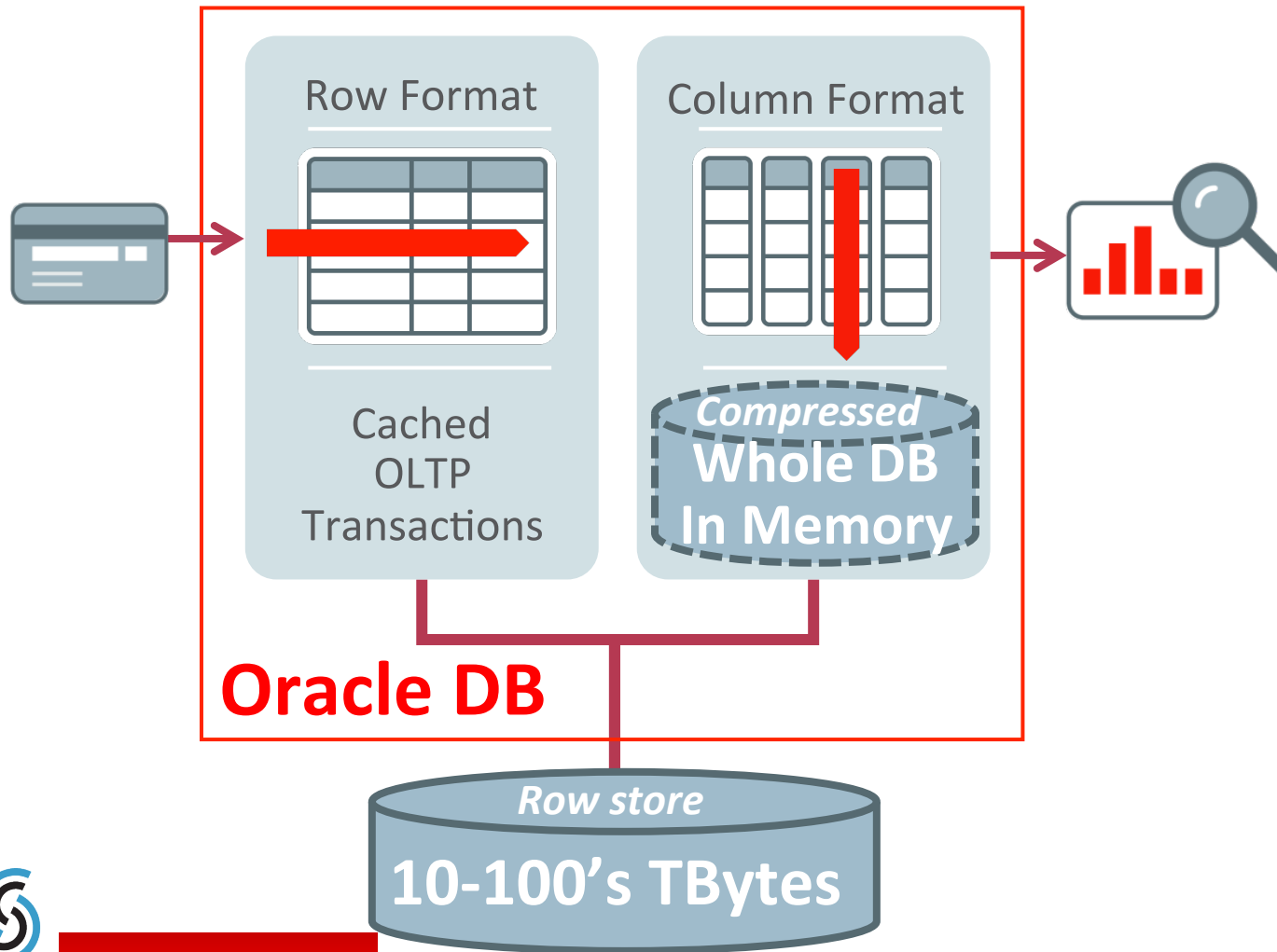
Big Feature Revolution

*# features collected is Exploding
10's to 100's*

Order	F1	F2	...	F150

Customer	F1	F2	...	F78	...	F100

Oracle Database In-Memory At Scale & SPARC DAX



- OLTP uses proven row format
- Analytics use in-memory Column
 - 10x faster analytics due to software
 - Columnar compression means huge databases can now fit in-memory
- Oracle database **stores BOTH** row and column formats
 - Simultaneously active and transactionally consistent
- **10x to 20x Faster SPARC DAX (Data Accelerator) HW innovation**

SPARC Dramatically Faster In-Memory SQL Analytics in DB

SPARC is multigenerational lead in performance: **14 Billion-row/sec per core**

9.4x faster
SPARC core advantage

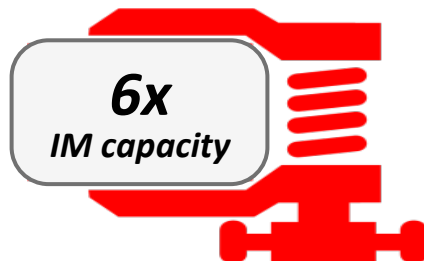
E5v4
x86

In-Memory SQL
Analytics

SPARC Offload

*DAX frees
Cores for other
processing*

6x Compression

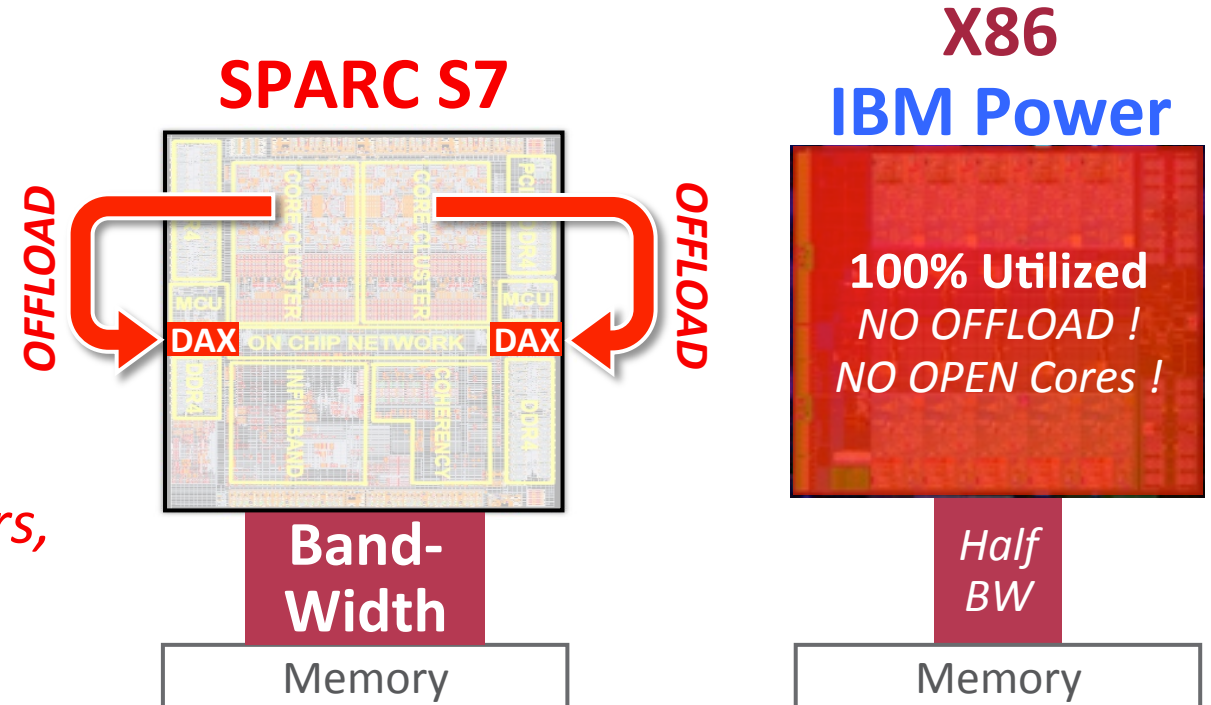


- **SPARC S7 9.4x faster per core than x86 E5 v4**
 - SPARC S7-2 422 query/m vs 2-chip x86 56 query/m
 - S7-2 (18-cores total), 2-chip x86 E5 v4 (20-cores total)
 - x86 per core performance flat for 4 generations
- DAX offloads In-Memory Scans
 - DAX offload allows cores to increase throughput
- 160GB in memory represents 1TB DB on disk
 - 6.2x compression with Oracle 12.2 In-memory

SPARC's Multi-generational Leapfrog in Performance

Radical Innovation: Integrated Offload offers 10x faster performance!

- Integrated Offload
 - Data Analytics Acceleration (DAX)
 - Encryption & Security
- *It's more important how you use transistors, than Moore's Law (the number transistors you make)*



Other vendors focused on Detached GPUs

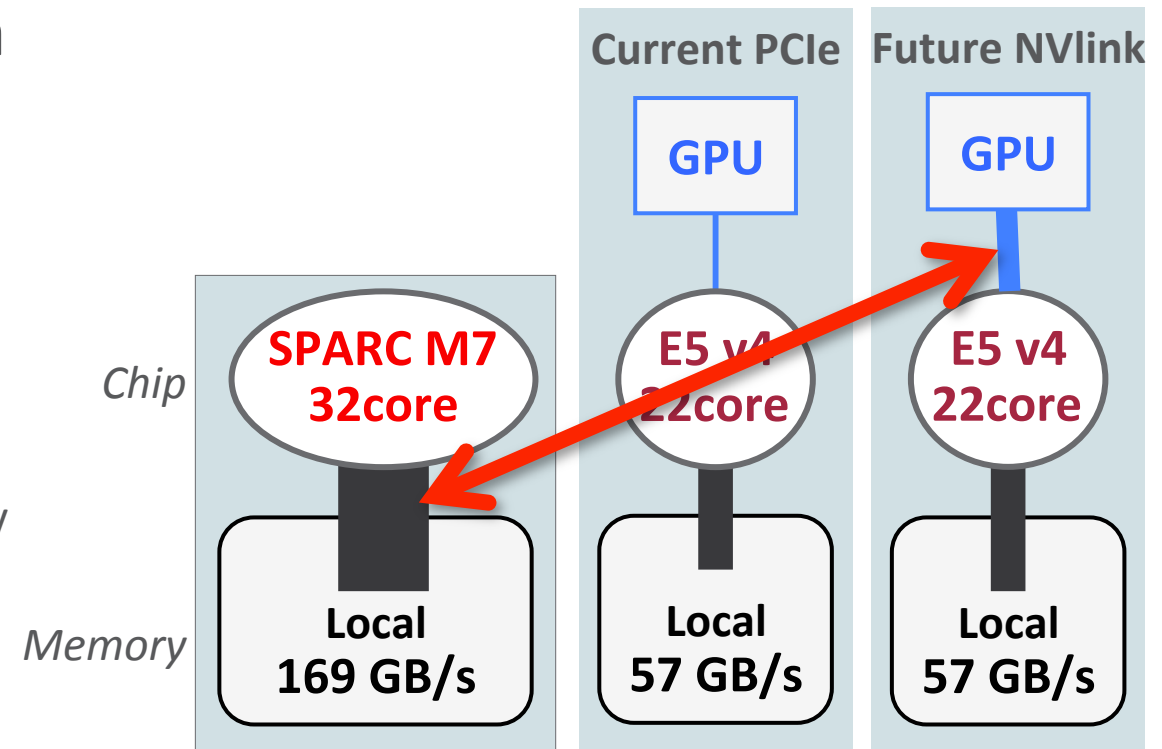
- *Detached GPUs are poorly designed for SQL*
- *Slow GPU interconnection robs performance*

Graphic Processing Units (GPUs)

GPUs Only Help Very Compute Intensive Algorithms

Complex ML/Statistics often not fast on GPUs... “Can only compute as fast as move data”

- Detached GPUs have limited bandwidth
 - GPUs fit for lots of simple graphics
 - Graphics have tiny data movement:
Coords & textures in, video frame out
 - Big Data ML much more demanding:
GPUs have limited applications
 - ML Learning & Scoring
 - Some ML Learning can be compute intensive, but only done initially to create Model
 - Nvidia Math Libraries only have BLAS3 routines
 - BLAS2 and BLAS1 do NOT have compute intensity therefore are not in library
 - Complicated ML has mix of BLAS1, BLAS2, BLAS3



Bandwidth lines to scale
GPU compute often stalled by lack of bandwidth

SQL & the Many Flavors of DSL – All potentials for DAX

SQL can be written in Domain Specific Languages (aka Language Integrated Queries)

- **SQL:**

- *SELECT count(*) from person WHERE citizen.age > 18*

- **Apache Spark SQL (DSL)**

- *val voters : Int = citizen.filter(\$"citizen.age" > 18).count()*

*Apache Spark feeds both
formats into SQL optimizer
Joins can be written & optimized*

- **Java Streams**

- *int voters = arrayList.parallelStream().filter(citizen-> citizen.olderThan(18)).count();*

- **Apache Eclipse (Goldman Sachs Collections)**

- *int voters = fastList.count(citizen-> citizen.olderThan(18));*



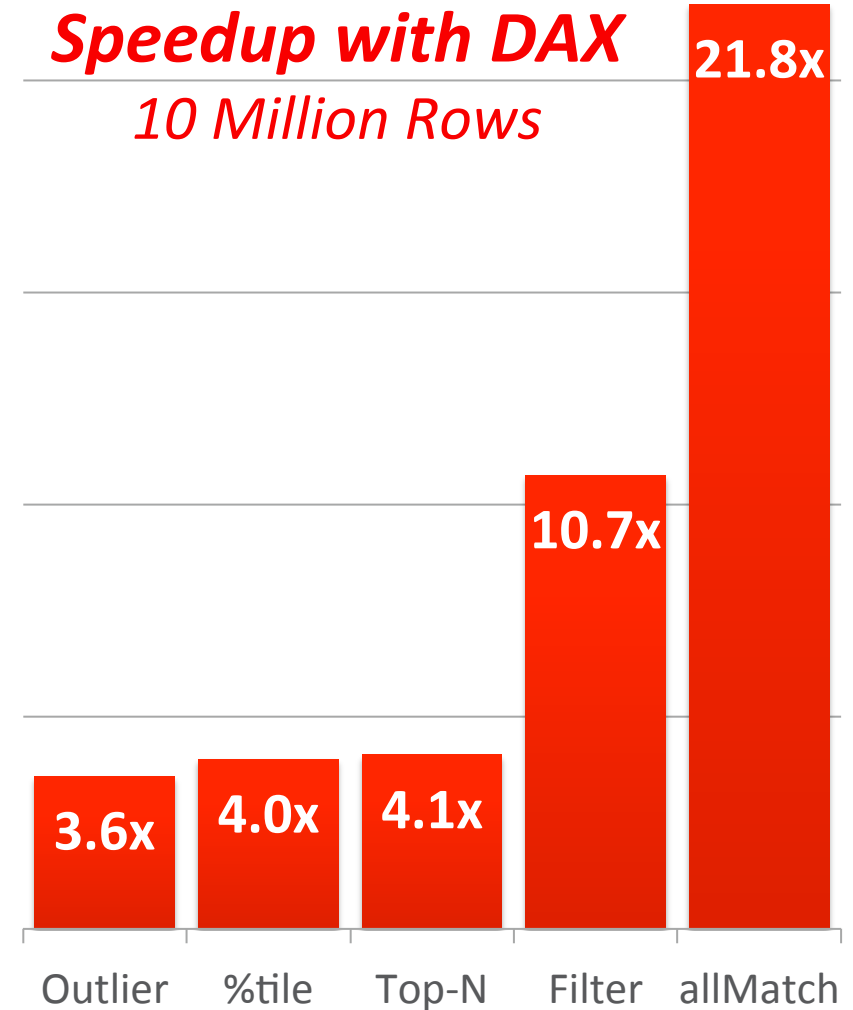
SPARC DAX Integrated with Java JDK8 Streams

DAX accelerates Java Streams: 3.6x to 21.8x faster

- Java Streams provide SQL-style code in Java
 - Java Streams is a perfect fit for DAX acceleration
- DAX & Java Streams
 - Integer Stream filter, allMatch, anyMatch, noneMatch, map(ternary operator), toArray & count functions
 - Simple code changes to use DAX
 - `import com.Oracle.Stream` *instead of* `import java.util.Stream`
 - `DaxIntStream` *instead of* `IntStream`
- Public access: <http://SWiSdev.oracle.com/DAX>

Java Stream API:

```
filter2_data.parallelStream().filter(w->w.temp<100).count()
```



Apache Spark: SQL and DSL Both Optimized by Catalyst Optimizer

Ex: “Select all books by authors born after 1980 named ‘Paulo’ from books & authors”

SQL

```
SELECT *  
FROM author a  
JOIN book b ON a.id = b.author_id  
WHERE a.year_of_birth > 1980  
      AND a.first_name = 'Paulo'  
ORDER BY b.title
```

Query DSL (Domain Specific Language)

```
val joinDF =  
  author.as('a')  
    .join(book.as('b'), $"a.id" === $"b.author_id")  
    .filter($"a.year_of_birth" > 1980)  
    .filter($"a.first_name" = "Paulo")  
    .orderBy('b.title)
```

Catalyst Optimizer

Optimized Plan
(can I reorder various operations?)

Whole-Stage CodeGen
new in Spark 2.0, this is Project Tungsten

Apache Spark 2.1

Whole-stage CodeGen Performance

2-chip x86 E5 v3(Haswell), total 36 cores, 72 threads/VCPU

- Let's evaluation performance on Full Table Scan of 600M rows
 - Time to Scan = 0.16 sec ***Impressive: 104.2 Million Rows/sec per core***



By Evaluating Delivered Bandwidth, We can see we are still Not Yet Efficient

2-chip x86 E5 v3(Haswell), total 36 cores, 72 threads/VCPU

- Evaluation performance on Full Table Scan of 600M rows
 - Time to Scan = 0.16 sec ***Impressive: 104.2 Million Rows/sec per core***
- Going back to 1st principles: Scanning is about data movement
 - What is bandwidth of 2.1 in-memory Scan? **14 GB/s (Spark 2.0.1)**
 - What is the system memory bandwidth? **114 GB/s (Stream Triad)**
 - ***7.6x more bandwidth available!***
- Other Queries deliver less: “SELECT count(*) FROM lineorder WHERE lo_quantity BETWEEN 10 and 20”
 - ***19x more bandwidth available! -- BUT only delivering 6GB/s on the system***

After Spark 2.1: Another 10x in Performance Still Possible

Select count(*) from store_sales where ss_item_sk > 100 and ss_item_sk < 1000

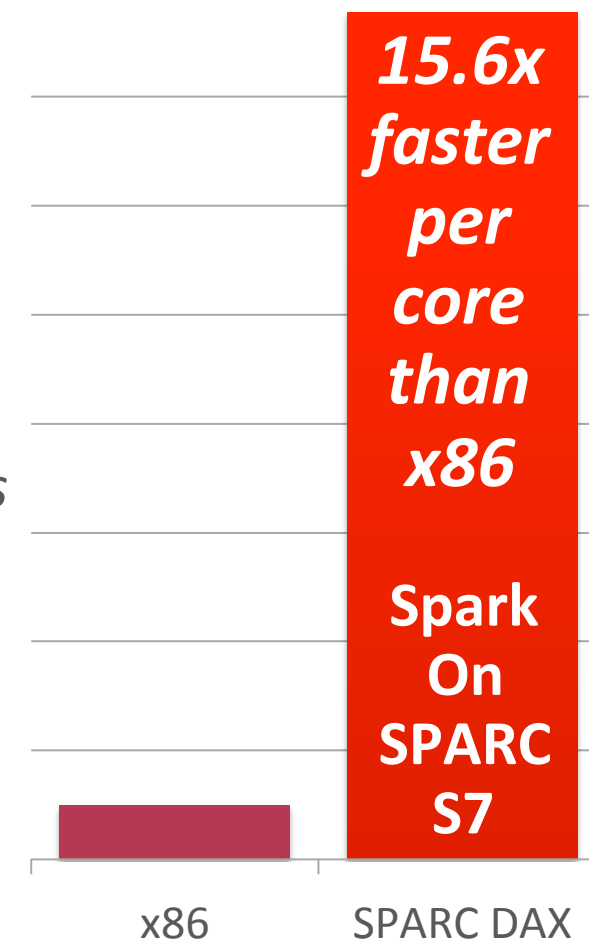
1. Volcano Iterator model: plan interpretation
 - *Open; Next data element; perform predicate; Close*
2. “College Freshman” Java : Whole-Stage CodeGen pipelined code 
 - *for (ss_item_sk in store_sales) {if (ss_item_sk > 100 and ss_item_sk < 1000) { count += 1}}*
3. Tuned true vectorization library: highly-tuned code operates on contiguous column
 - *vectorRangeFilter (n, VECTOR_OP_GT, 1000, VECTOR_OP_LT, 1000, store_sales, result, result_cnt)*
4. **Hardware Acceleration** further accelerates scanning 
 - *vectorRangeFilter (n, VECTOR_OP_GT, 1000, VECTOR_OP_LT, 1000, store_sales, result, result_cnt)*
 - x86 AVX2 (Graphics Instruction) – deliver 70 GB/s per chip
 - Oracle’s SPARC M7 processor – deliver 140 GB/s per chip



SPARC DAX Accelerating Apache Spark 2.1.0

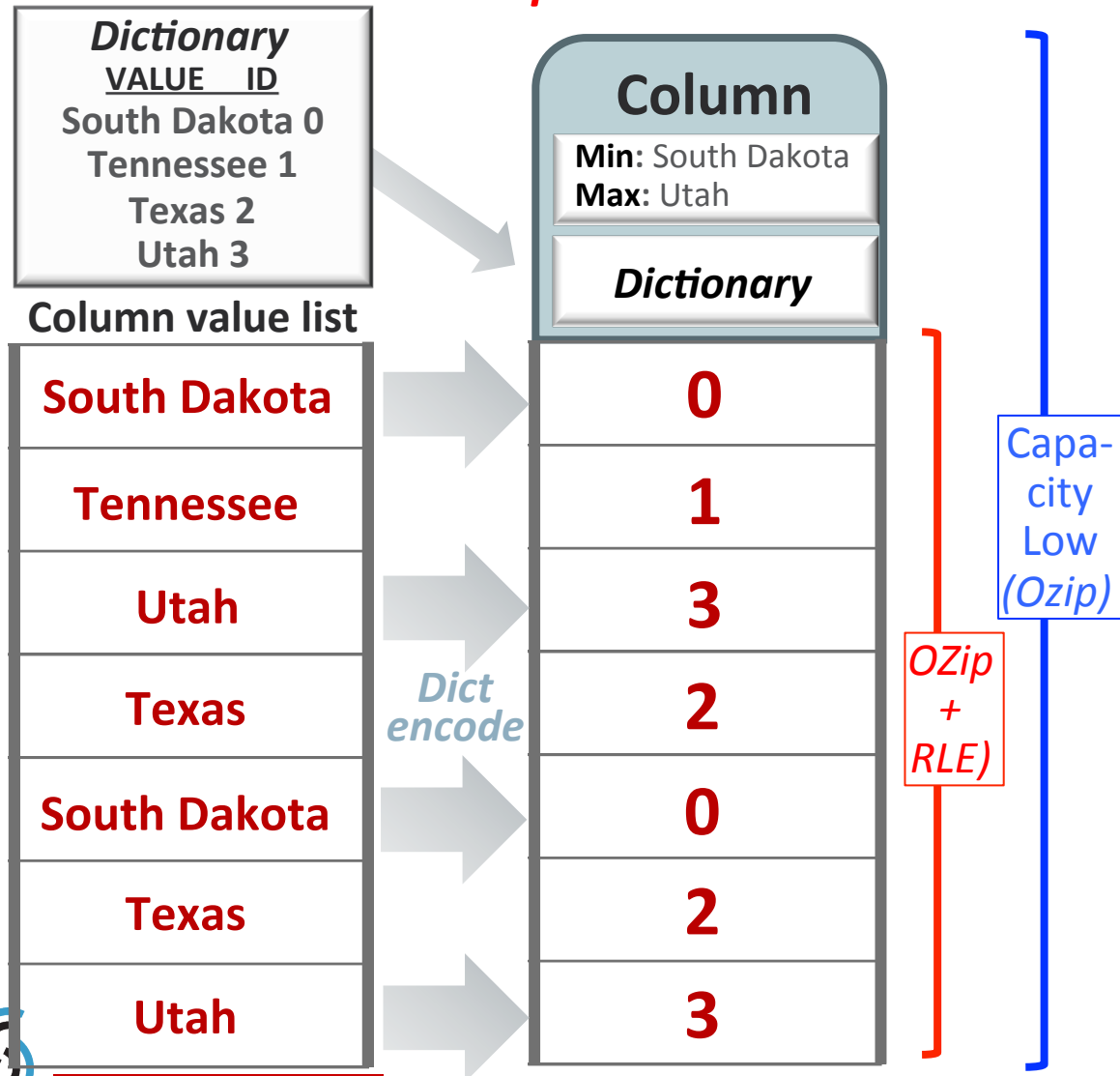
Spark SQL on SPARC S7-2 DAX **over 15.6x faster per core** than x86

- *Proof-of-Concept Prototype on SPARC S7:*
Apache Spark 2.1 POC: DAX & in-memory columnar
 - 2-chip SPARC S7 DAX: **9.8 Billion rows/sec** on 2-predicate scan
 - SPARC S7-2(16 total cores) 0.061 sec
 - 2-chip x86 (20 total cores) 0.760 sec *latest generation E5 v4 Broadwell*
 - *These advantages are over and above Tungsten's improvements*
- SPARC DAX offloads critical functions
 - SQL: filtering, dictionary encoding, 1 & 2 predicate, join processing, additional compression, etc.
- Libdax open API: <http://SWiSdev.oracle.com/DAX>



In-Memory Columnar Two-level Compression: Oracle DB

Same technique needs to be added to Apache Spark



- Efficient In Memory Columnar Table
- Dictionary encoding huge compression
 - 50 US only need 6 bits (<1 byte) vs. “South Dakota” needs 12 characters or 192 unicode bits = 32x smaller
 - Ozip/RLE compression is then applied on top of dictionary encoding
- Innovations that Apache Spark still needs
 - *Directly scan dictionary encoded data !*
 - *Range scans (2-predicate) in one step*
 - *Ozip decompression without writing to memory*
 - Save Min & Max for scan elimination
 - Can use dictionary for “featurization” of data for ML

libdax Open API for Free & Open Source Software (FOSS)

Libdax designed for key scan & dictionary operations to accelerate a variety of software

- Designed to accelerate wide variety of Oracle & FOSS software
 - Examples:
 - Oracle DB In-memory **9.5x faster** with DAX
 - Apache Spark SQL **over 16x faster** with DAX
 - ActiveViam **6x to 8x faster** with DAX
 - Java Streams **3.6x to 21.8x faster** with DAX
- Open API for libdax – sign up for free systems to actually develop/try code
 - Published sample codes
 - <https://swisdev.oracle.com>

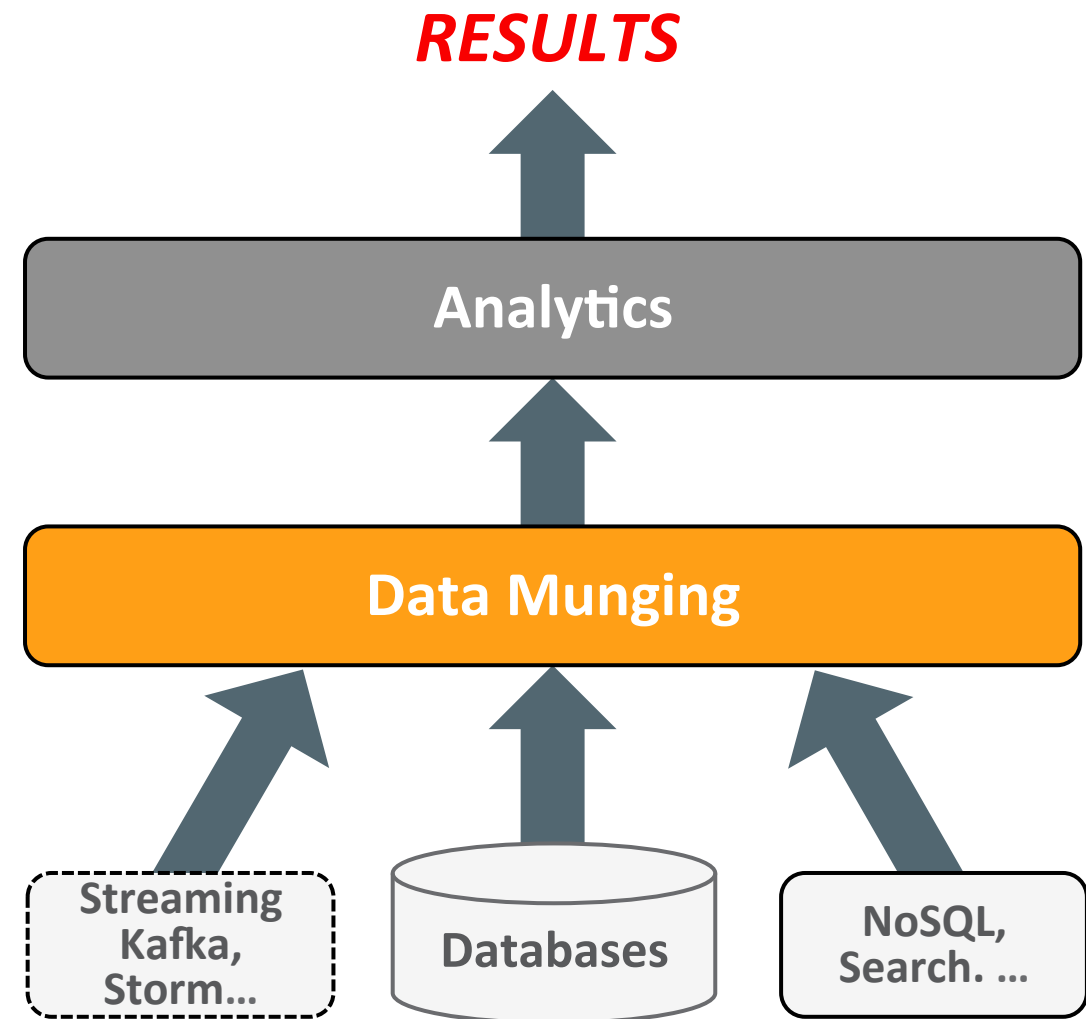
SPARC Processor is fastest for Statistics & ML (Machine Learning)

SPARC is Fastest for Spark Applications
SPARC DAX can dramatically accelerate Spark SQL

The Basic Analytics Flow

A lot of time spent in Data Munging

- Data can come from many sources
 - Databases, NoSQL, csv, feeds...
- We need to prepare it
 - Data Munging of all sorts!
- Analyze the data
 - Find the “right way” to analyze it
 - ML, Graph, SQL...

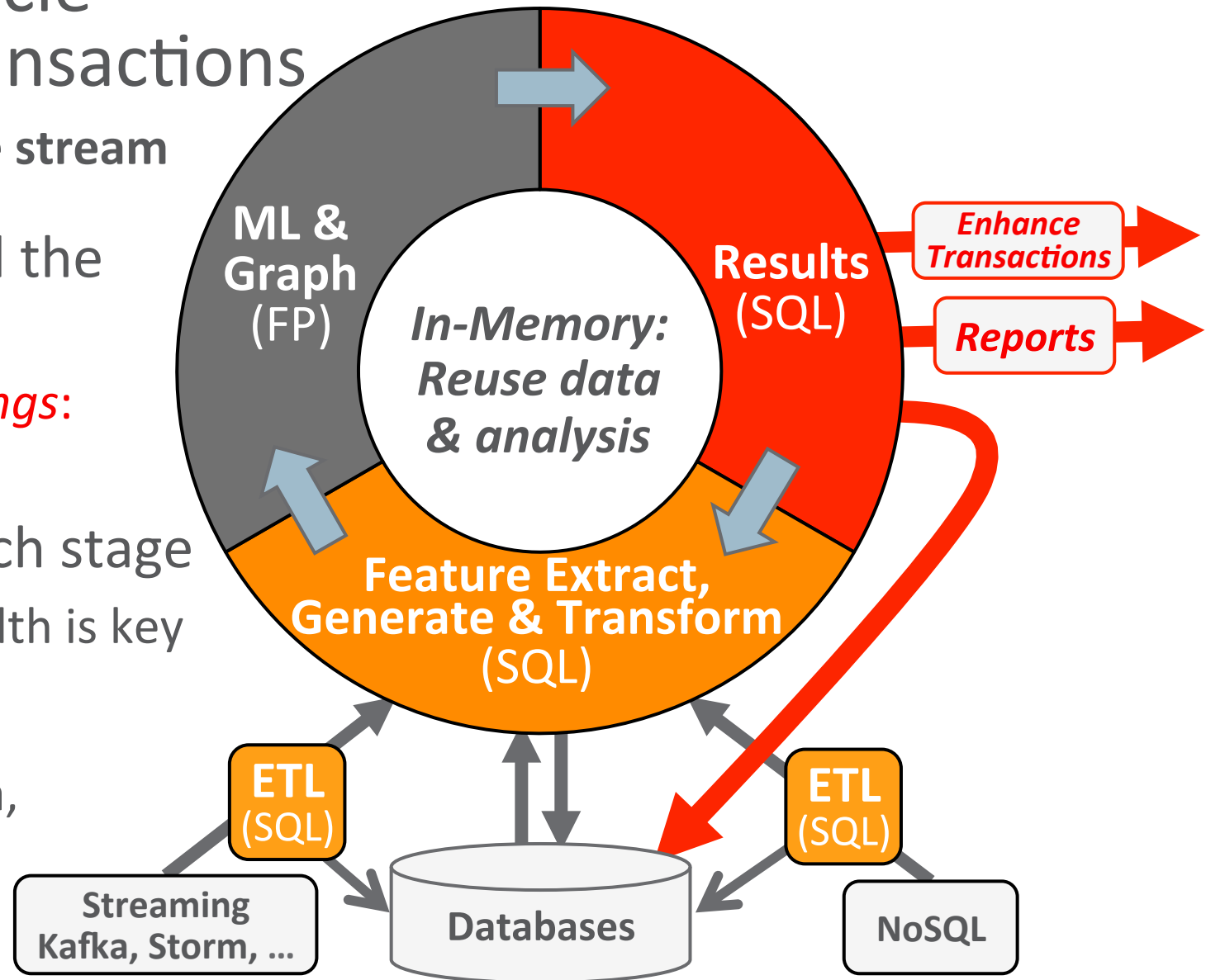


Continuous Analytics Cycle

Results Often Enrich Transactions

Analytics is more than one pipeline stream

- Continuous iterations around the data analytics wheel
 - Save, catalog, and re-use *all things*: data, SQL, code, and analytics
- In-memory advantages at each stage
 - SPARC's DAX & leading bandwidth is key
- Many sources of data
 - Internal proprietary, public data, external streaming, archives



Computational Graph Algorithms: PageRank & Single-Source Shortest Path (SSSP)



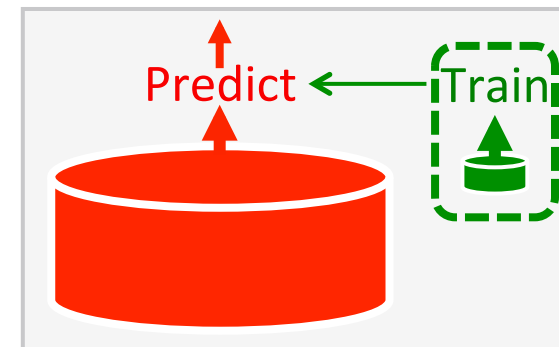
Graph: SPARC M7 up to 1.5x faster per core than x86

Graph Algorithm	Workload Size	4-chip X86 E5 v3	4-chip SPARC T7-4	SPARC per chip Advantage	SPARC per core Advantage
SSSP Bellman-Ford	448M vertices, 17.2B edges	39.2s	14.7s	2.7x	1.5x
	233M vertices, 8.6B edges	21.3s	8.5s	2.5x	1.4x
PageRank	448M vertices, 17.2B edges	136.7s	62.6s	2.2x	1.2x
	233M vertices, 8.6B edges	72.1s	27.6s	2.6x	1.5x

- Graph computations accelerated by SPARC's memory bandwidth
 - Bellman-Ford/SSSP (single-source shortest path) – optimal route or connection
 - PageRank - measuring website importance

Oracle Database Advanced Analytics Option Machine Learning on SPARC

ML training SPARC M7 up to 2.0x faster per core than x86

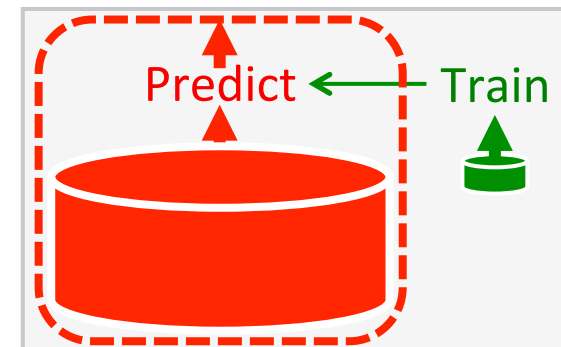


	Training: <i>Creating Model from data</i>	Attri- butes	X5-4 4-chip	T7-4 4-chip	SPARC per chip Advantage	SPARC per core Advantage
Supervised	SVM IPM Solver	900	1442s	404s	3.6x	2.0x
	GLM Classification	900	331s	154s	2.1x	1.2x
	SVM SGD Solver	9000	157s	84s	1.9x	1.1x
	GLM Regression	900	78s	55s	1.4x	0.8x
Cluster Model	Expectation Maximization	9000	763s	455s	1.7x	0.9x
	K-Means	9000	232s	161s	1.4x	0.8x

- Oracle Advanced Analytics in Oracle Database 12.2
 - SPARC M7 faster per core on training 64-bit floating point intensive
 - In-memory 640 million records, Airline On-time dataset

Oracle Database Advanced Analytics Option Machine Learning on SPARC

ML prediction SPARC 3.0x - 4.8x faster per core than x86



	Prediction <i>Using model on 1B records</i>	Attri- butes	X5-4 4-chip	T7-4 4-chip	SPARC per chip Advantage	SPARC per core Advantage
Supervised	SVM IPM Solver	900	206s	24s	8.6x	4.8x
	GLM Regression	900	166s	25s	6.6x	3.7x
	GLM Classification	900	156s	25s	6.2x	3.5x
	SVM SGD Solver	9000	132s	24s	5.5x	3.1x
Cluster Model	K-Means	9000	222s	35s	6.3x	3.6x
	Expectation Maximization	9000	243s	40s	6.1x	3.4x

- Oracle Advanced Analytics in Oracle Database 12.2
 - SPARC M7 much faster per core on scoring bandwidth-intensive
 - In-memory 1 Billion records, Airline On-time dataset

Spark ML DataFrame Algorithms use BLAS1 & some BLAS2

Lower compute intensity of 0.66 to 2.0, **BLAS1/BLAS2** limits ML performance, **BLAS3** faster

- ALS(Alternating Least Squares matrix factorization)
 - **BLAS2: _SPR, _TPSV**
 - **BLAS1: _AXPY, _DOT, _SCAL, _NRM2**
- Logistic regression classification
 - **BLAS2: _GEMV**
 - **BLAS1: _DOT, _SCAL**
- Generalized linear regression
 - **BLAS1: _DOT**
- Gradient-boosted tree regression
 - **BLAS1: _DOT**
- GraphX SVD++
 - **BLAS1: _AXPY, _DOT, _SCAL**
- Neural Net Multi-layer Perceptron
 - **BLAS3: _GEMM**
 - **BLAS2: _GEMV**

“One can only compute as fast as one can move data”

Max Flops = Compute-intensity * (Memory-BW/bytes-element)

Compute-Intensity = Flops/element

Single FP = 4bytes/element

Double FP = 8bytes/element

HINT: Spark Developers need to write sub-block matrix implementations to get more **BLAS3** usage in Spark ML ...Why?

BLAS3 are more efficient, because they use less memory bandwidth and reduce number of method calls

Fastest LAPACK solvers use **BLAS3**

GPUs Only Accelerate Some Compute Kernels

Detached GPU memory bandwidth is bottleneck for most computations

Routine Level	Operation	Flops	Mem	Compute Intensity	Tesla K80 peak	Max Possible Gflops @12GB/s block=100 PCIe	GPU % peak	Max Possible Gflops @80GB/s block=100 Nvlink (future)	GPU % peak
BLAS1 (vector)	$y = ax + y$	$2n$	$3n$	$2/3$	935 Gflops	1 Gflops	0.1%	7 Gflops	0.7%
BLAS2 (matrix-vector)	$y = Ax + y$	$2n^2 + 2n$	n^2	2	935 Gflops	3 Gflops	0.3%	20 Gflops	2.1%
BLAS3 (matrix-matrix)	$C = AB + C$	$2n^3$	$4n^2$	$n/2$	935 Gflops	75 Gflops	8.0%	500 Gflops	53.5%
FFT	$C = \text{FFT}(A)$	$n \log(n)$	$2n$	$\log(n)/2$	935 Gflops	239 Gflops 1M 1D-FFT	25%	935 Gflops 1M 1D-FFT	Near peak

Only 6 BLAS3 routine types in Nvidia Library: gemm, syrkm, syr2km, trsm, trmm, symm
Can write codes using BLAS2/1 in GPUs but resulting compute intensity must be large to accelerate

SPARC Advantages on Analytics Across the Spectrum

Key underlying technologies provide dramatic performance advantages

SQL DAX Offload	Machine Learning	Graph & Spatial	NoSQL
<ul style="list-style-type: none">• <i>Oracle Database</i>• <i>Big Data with DAX</i>	<ul style="list-style-type: none">• <i>Spark MLlib</i>• <i>Advanced Analytics</i>	<ul style="list-style-type: none">• <i>Spark GraphX</i>• <i>Oracle Spatial&Graph</i>	<ul style="list-style-type: none">• <i>Cassandra NoSQL</i>• <i>Oracle NoSQL</i>
up to 10x faster per core	up to 5x faster per core	up to 1.5x faster per core	up to 1.9x faster per core

SPARC DAX offload acceleration
Oracle Developer Perflib math libraries

SPARC's Java & JVM advantages

Required Benchmark Disclosure Statement

Must be in SPARC S7 & M7 Presentations with Benchmark Results

- Additional Info: <http://blogs.oracle.com/bestperf>
- Copyright 2016, Oracle &/or its affiliates. All rights reserved. Oracle&Java are registered trademarks of Oracle &/or its affiliates. Other names may be trademarks of their respective owners
- SPEC and the benchmark name SPECjEnterprise are registered trademarks of the Standard Performance Evaluation Corporation. Results from www.spec.org as of 7/6/2016. SPARC S7-2, 14,400.78 SPECjEnterprise2010 EjOPS (unsecure); SPARC S7-2, 14,121.47 SPECjEnterprise2010 EjOPS (secure) Oracle Server X6-2, 27,803.39 SPECjEnterprise2010 EjOPS (unsecure); IBM Power S824, 22,543.34 SPECjEnterprise2010 EjOPS (unsecure); IBM x3650 M5, 19,282.14 SPECjEnterprise2010 EjOPS (unsecure).
- SPEC and the benchmark name SPECjbb are registered trademarks of Standard Performance Evaluation Corporation (SPEC). Results from <http://www.spec.org> as of 6/29/2016. SPARC S7-2 (16-core) 65,790 SPECjbb2015-MultiJVM max-jOPS, 35,812 SPECjbb2015-MultiJVM critical-jOPS; IBM Power S812LC (10-core) 44,883 SPECjbb2015-MultiJVM max-jOPS, 13,032 SPECjbb2015-MultiJVM critical-jOPS; SPARC T7-1 (32-core) 120,603 SPECjbb2015-MultiJVM max-jOPS, 60,280 SPECjbb2015-MultiJVM critical-jOPS; Huawei RH2288H v3 (44-core) 121,381 SPECjbb2015-MultiJVM max-jOPS, 38,595 SPECjbb2015-MultiJVM critical-jOPS; HP ProLiant DL360 Gen9 (44-core) 120,674 SPECjbb2015-MultiJVM max-jOPS, 29,013 SPECjbb2015-MultiJVM critical-jOPS; HP ProLiant DL380 Gen9 (44-core) 105,690 SPECjbb2015-MultiJVM max-jOPS, 52,952 SPECjbb2015-MultiJVM critical-jOPS;; Cisco UCS C220 M4 (44-core) 94,667 SPECjbb2015-MultiJVM max-jOPS, 71,951 SPECjbb2015-MultiJVM critical-jOPS; Huawei RH2288H V3(36-core) 98,673 SPECjbb2015-MultiJVM max-jOPS, 28,824 SPECjbb2015-MultiJVM critical-jOPS; Lenovo x240 M5 (36-core) 80,889 SPECjbb2015-MultiJVM max-jOPS, 43,654 SPECjbb2015-MultiJVM critical-jOPS; SPARC T5-2 (32-core) 80,889 SPECjbb2015-MultiJVM max-jOPS, 37,422 SPECjbb2015-MultiJVM critical-jOPS; SPARC S7-2 (16-core) 66,612 SPECjbb2015-Distributed max-jOPS, 36,922 SPECjbb2015-Distributed critical-jOPS; HP ProLiant DL380 Gen9 (44-core) 120,674 SPECjbb2015-Distributed max-jOPS, 39,615 SPECjbb2015-Distributed critical-jOPS; HP ProLiant DL360 Gen9 (44-core) 106,337 SPECjbb2015-Distributed max-jOPS, 55,858 SPECjbb2015-Distributed critical-jOPS; HP ProLiant DL580 Gen9 (96-core) 219,406 SPECjbb2015-Distributed max-jOPS, 72,271 SPECjbb2015-Distributed critical-jOPS; Lenovo Flex System x3850 X6 (96-core) 194,068 SPECjbb2015-Distributed max-jOPS, 132,111 SPECjbb2015-Distributed critical-jOPS.
- SPEC and the benchmark names SPECfp and SPECint are registered trademarks of the Standard Performance Evaluation Corporation. Results as of October 25, 2015 from www.spec.org and this report. 1 chip results SPARC T7-1: 1200 SPECint_rate2006, 1120 SPECint_rate_base2006, 832 SPECfp_rate2006, 801 SPECfp_rate_base2006; SPARC T5-1B: 489 SPECint_rate2006, 440 SPECint_rate_base2006, 369 SPECfp_rate2006, 350 SPECfp_rate_base2006; Fujitsu SPARC M10-4S: 546 SPECint_rate2006, 479 SPECint_rate_base2006, 462 SPECfp_rate2006, 418 SPECfp_rate_base2006. IBM Power 710 Express: 289 SPECint_rate2006, 255 SPECint_rate_base2006, 248 SPECfp_rate2006, 229 SPECfp_rate_base2006; Fujitsu CELSIUS C740: 715 SPECint_rate2006, 693 SPECint_rate_base2006; NEC Express5800/R120f-1M: 474 SPECfp_rate2006, 460 SPECfp_rate_base2006.
- Two-tier SAP Sales and Distribution (SD) standard application benchmarks, SAP Enhancement Package 5 for SAP ERP 6.0 as of 5/16/16: SPARC M7-8, 8 processors / 256 cores / 2048 threads, SPARC M7, 4.133 GHz, 130000 SD Users, 713480 SAPs, Solaris 11, Oracle 12cSAP, Certification Number: 2016020, SPARC T7-2 (2 processors, 64 cores, 512 threads) 30,800 SAP SD users, 2 x 4.13 GHz SPARC M7, 1 TB memory, Oracle Database 12c, Oracle Solaris 11, Cert# 2015050. HPE Integrity Superdome X (16 processors, 288 cores, 576 threads) 100,000 SAP SD users, 16 x 2.5 GHz Intel Xeon Processor E7-8890 v3 4096 GB memory, SQL Server 2014, Windows Server 2012 R2 Datacenter Edition, Cert# 2016002. IBM Power System S824 (4 processors, 24 cores, 192 threads) 21,212 SAP SD users, 4 x 3.52 GHz POWER8, 512 GB memory, DB2 10.5, AIX 7, Cert#201401. Dell PowerEdge R730 (2 processors, 36 cores, 72 threads) 16,500 SAP SD users, 2 x 2.3 GHz Intel Xeon Processor E5-2699 v3 256 GB memory, SAP ASE 16, RHEL 7, Cert#2014033. HP ProLiant DL380 Gen9 (2 processors, 36 cores, 72 threads) 16,101 SAP SD users, 2 x 2.3 GHz Intel Xeon Processor E5-2699 v3 256 GB memory, SAP ASE 16, RHEL 6.5, Cert#2014032. SAP, R/3, reg TM of SAP AG in Germany and other countries. More info www.sap.com/benchmarks.

Backup Slides



Why Apache Spark is best on SPARC

Design innovations at every level

- SPARC's leadership in fast **JVM performance** speeds Scala
 - SPARC design focus on Java & JVM benefits all Apps that use JVM
- SPARC's huge lead in **delivered memory bandwidth**
 - SPARC accelerates Spark's in-memory design – Big data apps needs fast data movement
- SPARC DAX to provide huge **offload acceleration** in Apache Spark
 - Contributing code to Apache Spark SQL/DataFrames to use DAX
- SPARC's leading **floating-point** performance for Machine Learning
 - Designed for fast Floating Point (FP) on sparse & advanced algorithm

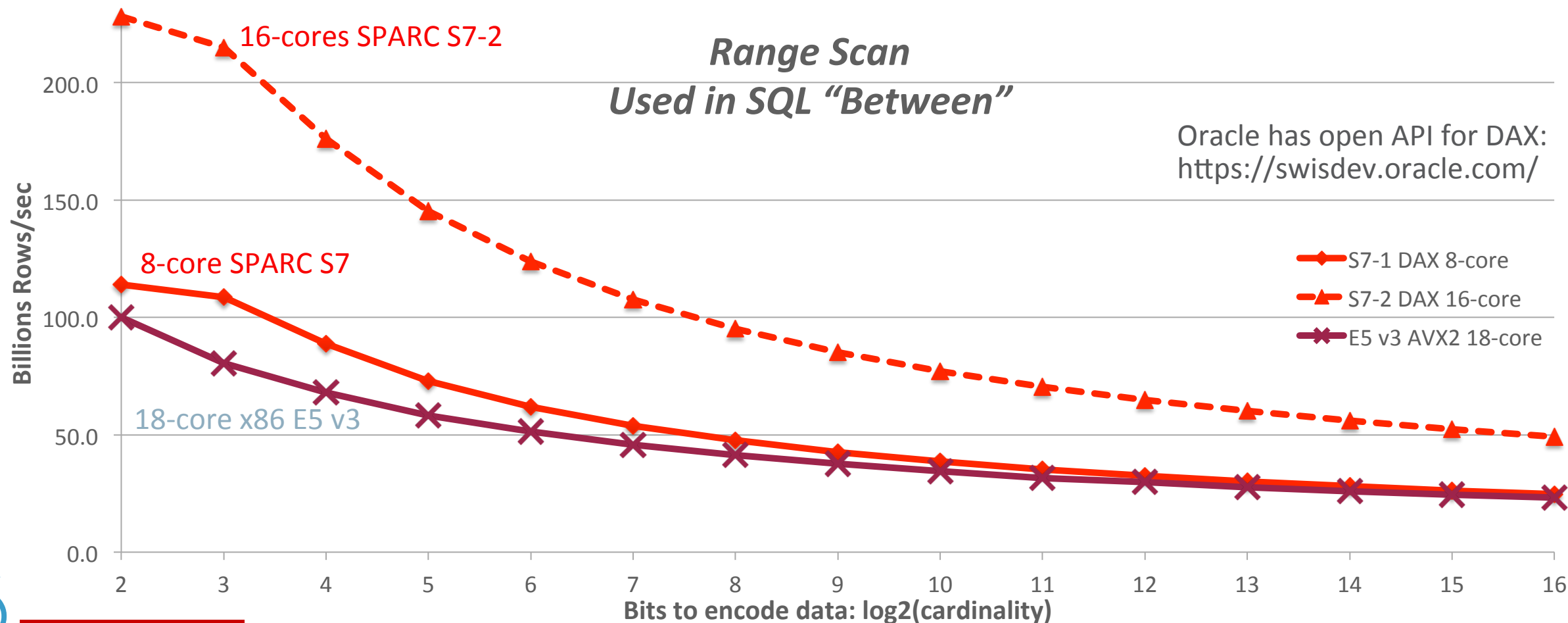
SPARC Advantages for Every Form of Analytics

SPARC is 1.3x to over 10x faster per core on Analytics than x86

<i>Analytics</i>	<i>In-memory Columnar SW & Software in Silicon (SPARC DAX)</i>	<i>SPARC's efficient CPU & memory design</i>
Database Analytics <i>Analytic SQL Queries for database</i>	✓	✓
Java Analytics <i>Java Code that probes data SQL-like code</i>	✓	✓
Machine Learning (ML) <i>Automatically finding hidden patterns & correlations</i>	✓	✓
Graph <i>Automatically find patterns in Social networks, etc.</i>		✓
NoSQL <i>Fast access to semi-structured & unstructured data</i>		✓
Spatial <i>Fast throughput on geometric data</i>		✓

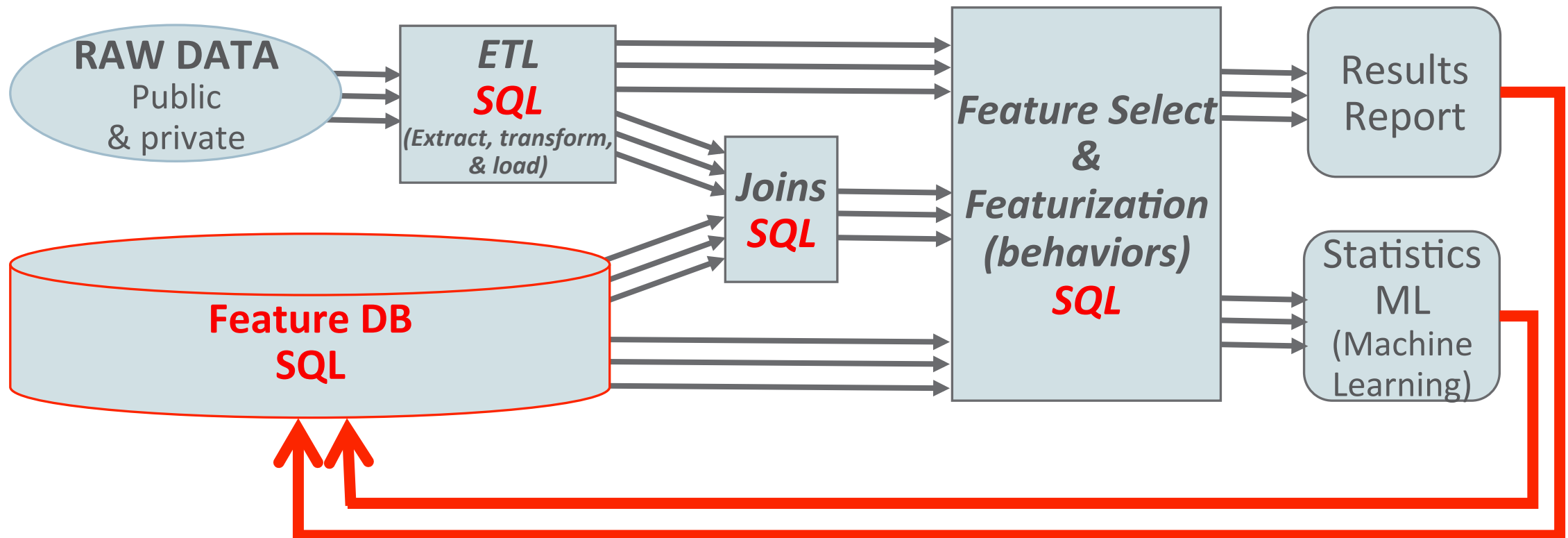
SPARC DAX Scan Performance versus x86 E5 v3 (AVX2)

DAX offload scans: **SPARC S7 only uses 25% cores and x86 uses 50% cores**



“Big” Features: Attributes/Qualities to Analyze

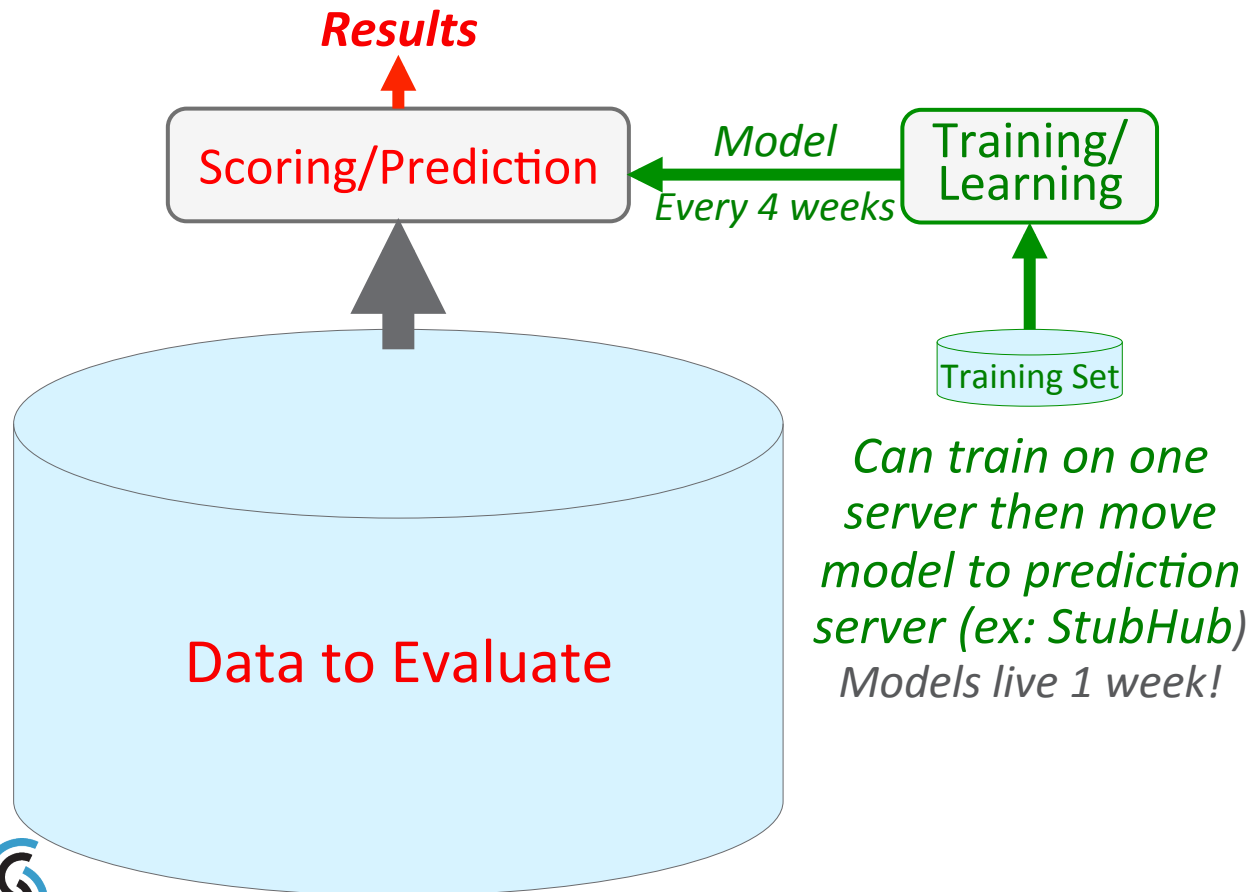
Big Data is also Big Features: 10's to 100's features to chose, create, & **manage**



- Organizing features is critical to the analytics cycle

Machine Learning (ML): Scoring/Prediction versus Training/Learning Characteristics

Prediction/Scoring operates on huge amounts of data with low compute intensity



	ML Score/ Prediction	ML Learn/ Train
% of activity	Most Data	*Initial
Computation	$O(n^2)$ Matrix-vector	$O(n^3)$ Matrix-matrix
Data	$O(n^2)$	$O(n^2)$
Compute Intensity (Compute/Data)	Low constant	$O(n)$
SPARC Advantage Due to Memory Bandwidth & design	3x to 6x per core	Up to 1.3x per core

**Initial and then occasionally update models data samples*

ORACLE®