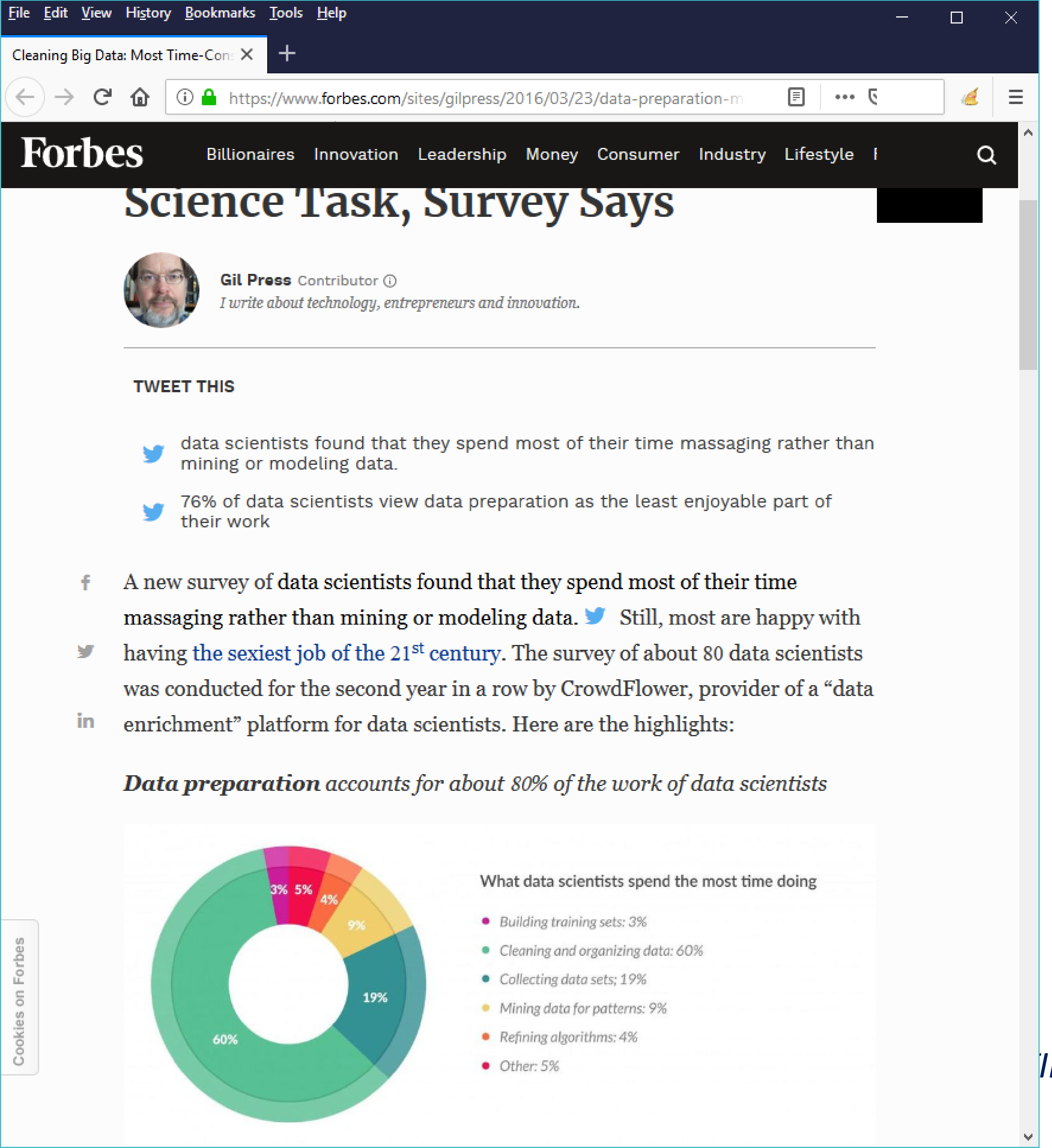
“Faster predictions, better decisions”

Part 1:
Why a fast ETL matters?


File Edit View History Bookmarks Tools Help

Cleaning Big Data: Most Time-Con:

https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-m

Forbes Billionaires Innovation Leadership Money Consumer Industry Lifestyle

Science Task, Survey Says

Gil Press Contributor

I write about technology, entrepreneurs and innovation.

TWEET THIS

data scientists found that they spend most of their time massaging rather than mining or modeling data.

76% of data scientists view data preparation as the least enjoyable part of their work

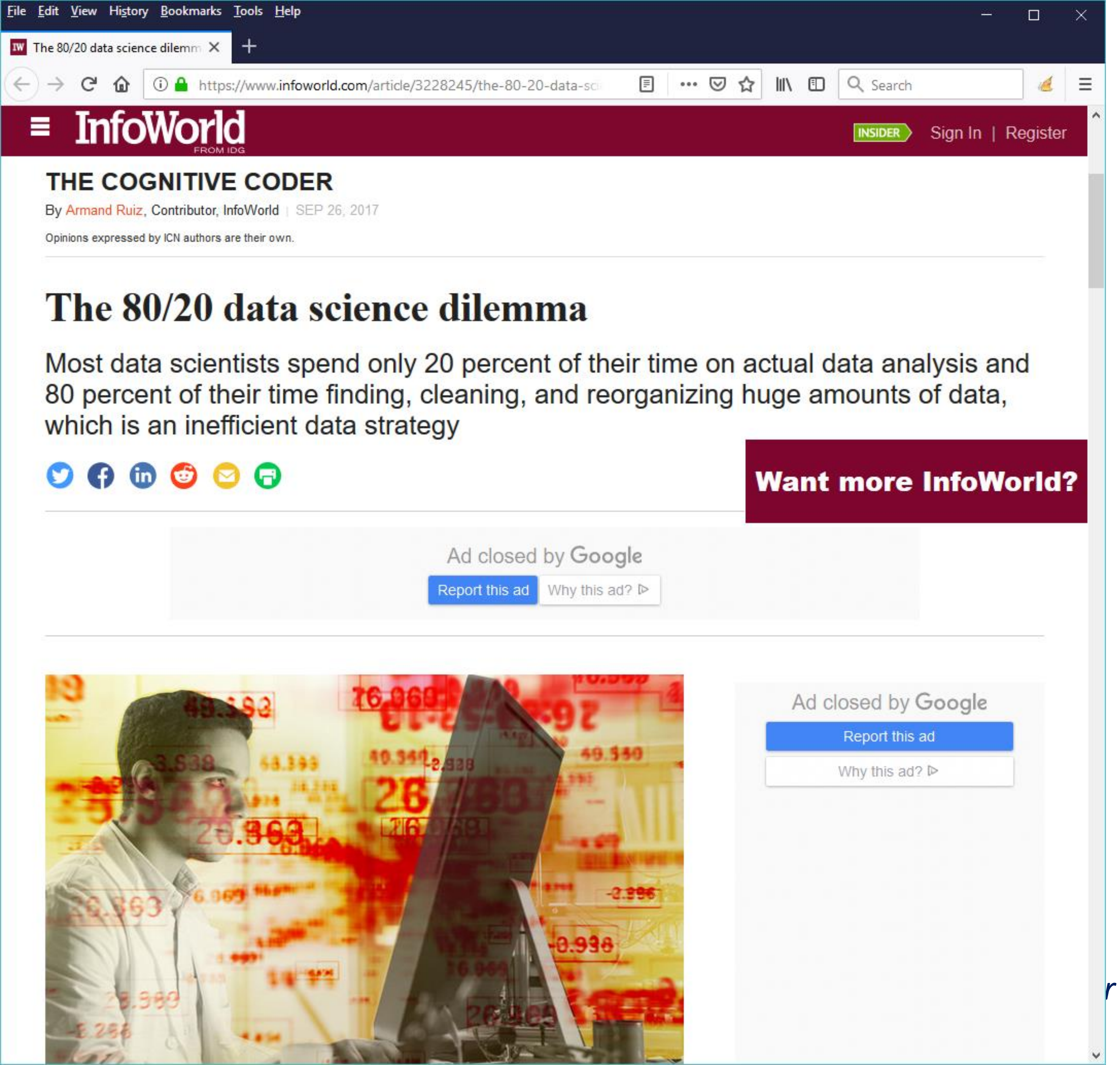
A new survey of data scientists found that they spend most of their time massaging rather than mining or modeling data. Still, most are happy with having the sexiest job of the 21st century. The survey of about 80 data scientists was conducted for the second year in a row by CrowdFlower, provider of a "data enrichment" platform for data scientists. Here are the highlights:

***Data preparation** accounts for about 80% of the work of data scientists*

The 80/20 data science dilemma 1/3

<u>**Forbes:**</u> *"Data Preparation (ETL tasks) account for about 80% of the work of data scientists"*

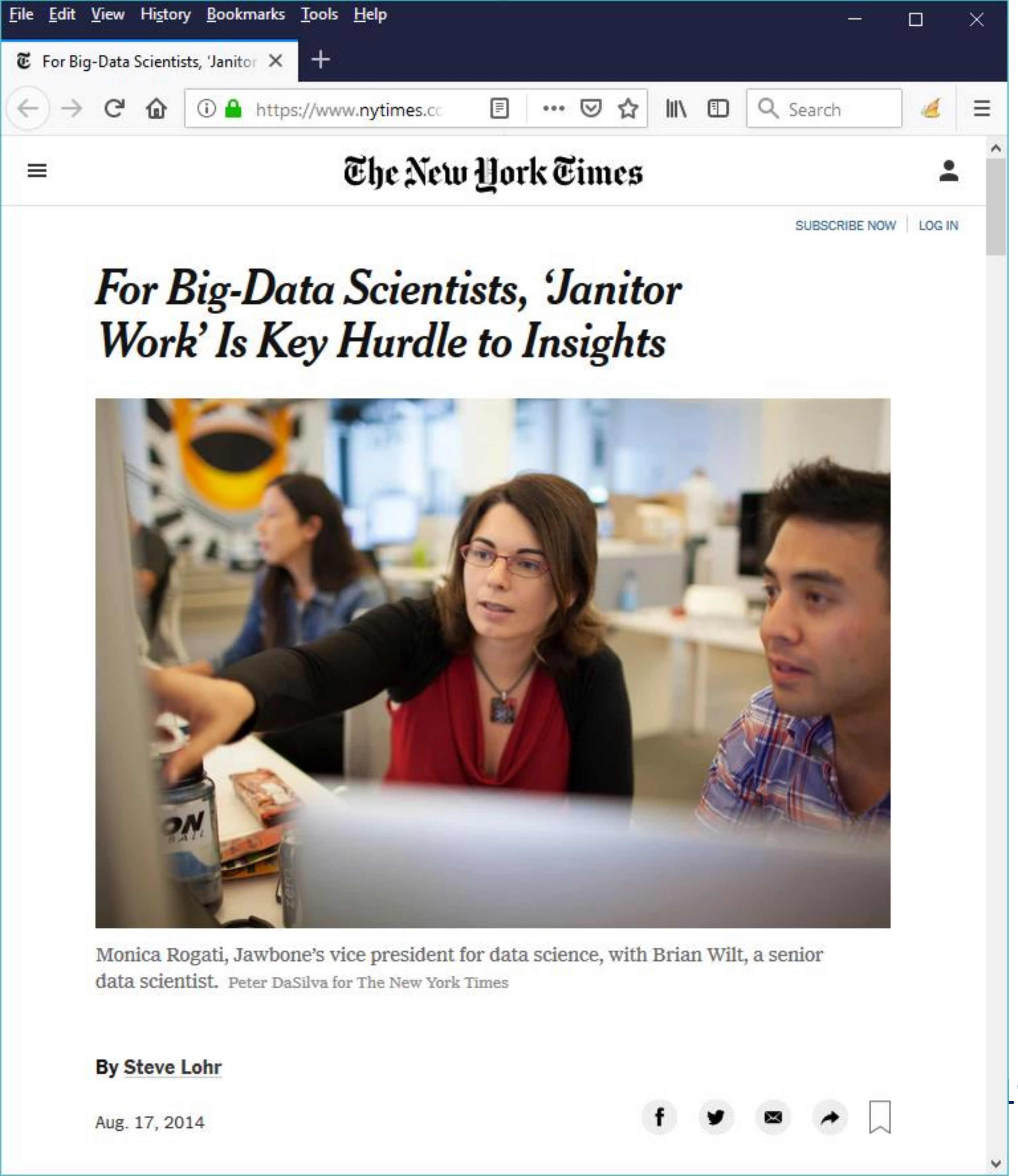
<u>Source:</u> https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#3640cbb36f63



The 80/20 data science dilemma 3/3

<u>InfoWorld:</u> *"…Data scientists spend … 80% of their time …reorganizing huge amounts of data (i.e. doing ETL tasks)".*

<u>Source:</u> https://www.infoworld.com/article/3228245/the-80-20-data-science-dilemma.html



The 80/20 data science dilemma 2/3

<u>New York Yimes:</u> *"Data scientists … spend from 50% to 80% of their time in … data wrangling (ETL tasks)"*

…Data scientists, according to interviews and expert estimates, spend from 50 percent to 80 percent of their time mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets.
"Data wrangling is a huge — and surprisingly so — part of the job," said Monica Rogati, vice president for data science at Jawbone…

<u>Source:</u> https://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html

timi
integrated data mining

First Conclusion

Data Scientists need a fast(er) ETL

Objective:

anatella
data in motion
vs
APACHE Spark™

THE ULTIMATE SHOWDOWN

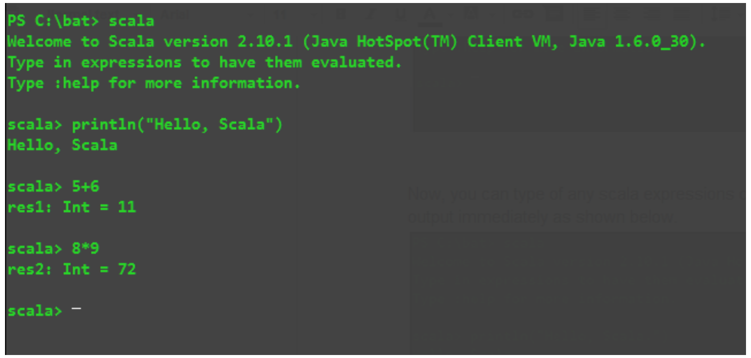
Apache Spark

anatella
data in motion

Coded in Java

Coded in C ASM

The users see:



Scala
PySpark

The users see:

R JS python

Spark is used inside:

databricks cloudera
data iku
IBM Analytics

Anatella is used inside:

timi
integrated data mining

Part 2

The TPC-H benchmark

TPC-H benchmark

Creation date: February 1998

http://www.tpc.org/tpch/

A world-famous benchmark to measure database efficiency on common “BI” Queries

8/6/2001

10/4/1999

10/9/2001

3/31/2003

12/5/2002

7/15/1999

2009/6/8

(The Dates are the dates of first participation)

TPC-H benchmark

Objective: run 22 SQL queries as fast as possible on a "reference" database:

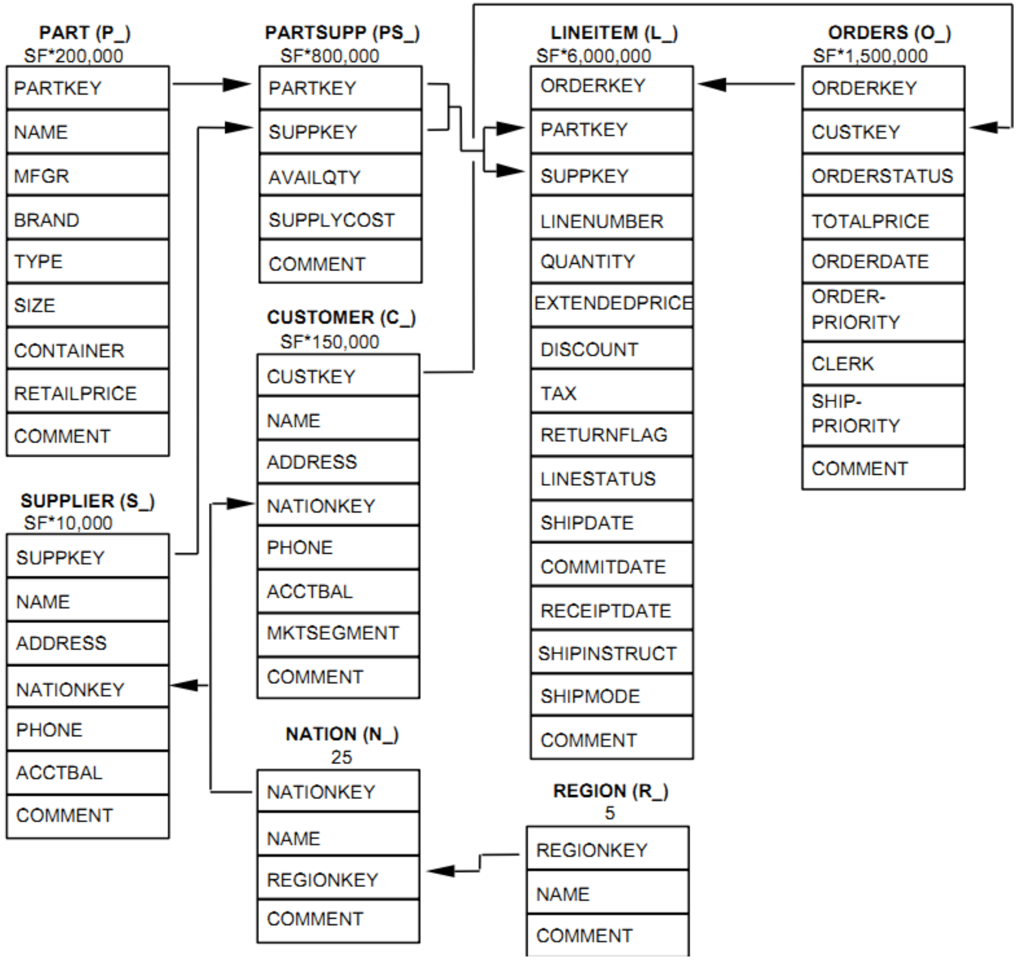
2 categories of results:
- Clustered category (Database is distributed on many PC)
- Non-Clustered category (Database is running on 1 PC)

Rankings:
- by Speed
- by "Efficiency" (i.e. speed divided by price; price includes hardware)

We run the 22 queries on 4 different database sizes (SF):

Unit: millions of rows	1GB	10GB	100GB	1TB
#Customers	0.15	1.5	15	150
#Purchases	6	60	600	6000

Figure 2: The TPC-H Schema



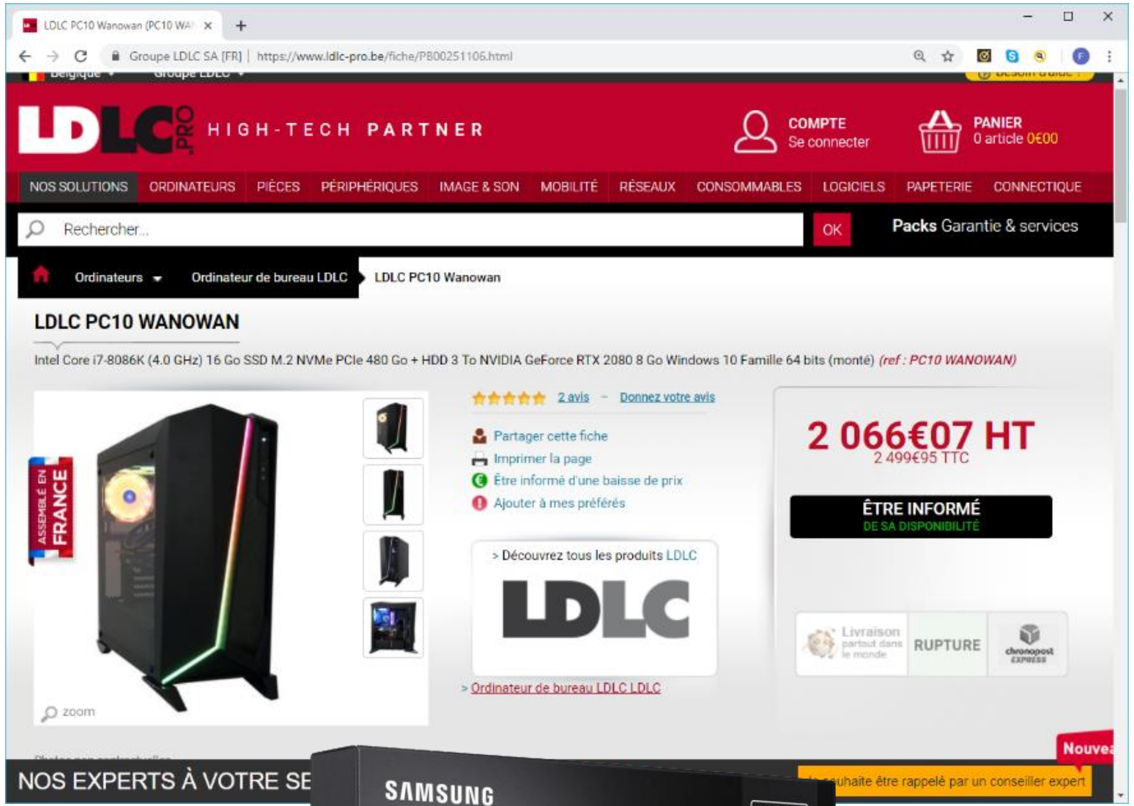
This is thus 6 billions rows in one table

Technical considerations

All tests are running on:
https://www.ldlc-pro.be/fiche/PB00251106.html
All data is stored on a SSD (Samsung 970 NVMe 2TB)

ETL tool	Data Storage
timi integrated data mining	Columnar Gel Files
Apache Spark from January 2019.	Parquet

All queries run inside a non-interactive session



454€ on LDLC

timi integrated data mining

TPC-H benchmark

"Official" TPC-H Query 4 expressed in "SQL":

```
select
        o_orderpriority,
        count(*) as order_count
from
        orders
where
        o_orderdate >= date '[DATE]'
        and o_orderdate < date '[DATE]' + interval '3' month
        and exists (
                select
                        *
                from
                        lineitem
                where
                        l_orderkey = o_orderkey
                        and l_commitdate < l_receiptdate
        )
group by
        o_orderpriority
order by
        o_orderpriority;
```

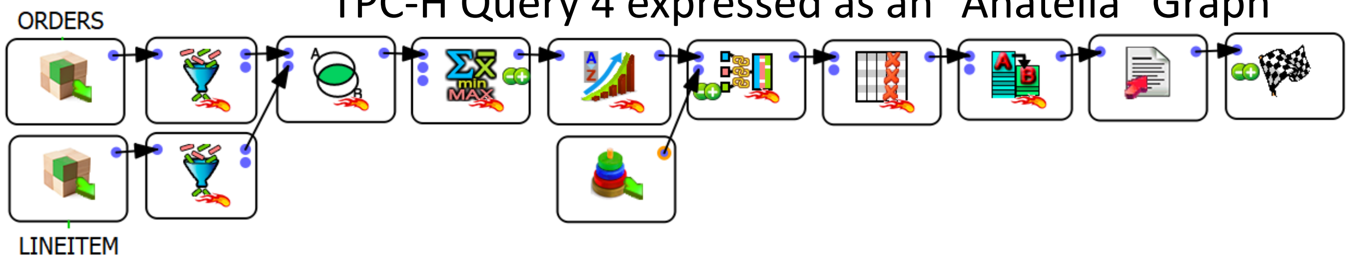
TPC-H Query 4 expressed in "Scala"

```
package main.scala
import org.apache.spark.sql.DataFrame
import org.apache.spark.SparkContext
import org.apache.spark.sql.functions.count
/**
 * TPC-H Query 4
 * Savvas Savvides <savvas@purdue.edu>
 */
class Q04 extends TpchQuery {
  override def execute(sc: SparkContext, schemaProvider: TpchSchemaProvider): DataFrame = {
    val sqlContext = new org.apache.spark.sql.SQLContext(sc)
    import sqlContext.implicits._
    import schemaProvider._

    val forders = order.filter($"o_orderdate" >= "1993-07-01" && $"o_orderdate" < "1993-10-01")
    val flineitems = lineitem.filter($"l_commitdate" < $"l_receiptdate")
      .select($"l_orderkey")
      .distinct

    flineitems.join(forders, $"l_orderkey" === forders("o_orderkey"))
      .groupBy($"o_orderpriority")
      .agg(count($"o_orderpriority"))
      .sort($"o_orderpriority")
  }
}
```

TPC-H Query 4 expressed as an "Anatella" Graph



Thanks to Savvas Savvides (savvas@purdue.edu) for providing the optimized Spark/Scala code!

All results are validated against the "reference" answers provided by the TPC-H. For example, for Q4:

DataTable (5 rows - 2 columns) (complete)

	O_ORDERPRIORITY	ORDER_COUNT
1	1-URGENT	10594
2	2-HIGH	10476
3	3-MEDIUM	10410
4	4-NOT SPECIFIED	10556
5	5-LOW	10487

timi integrated data mining

TPC-H benchmark result table

Table 1

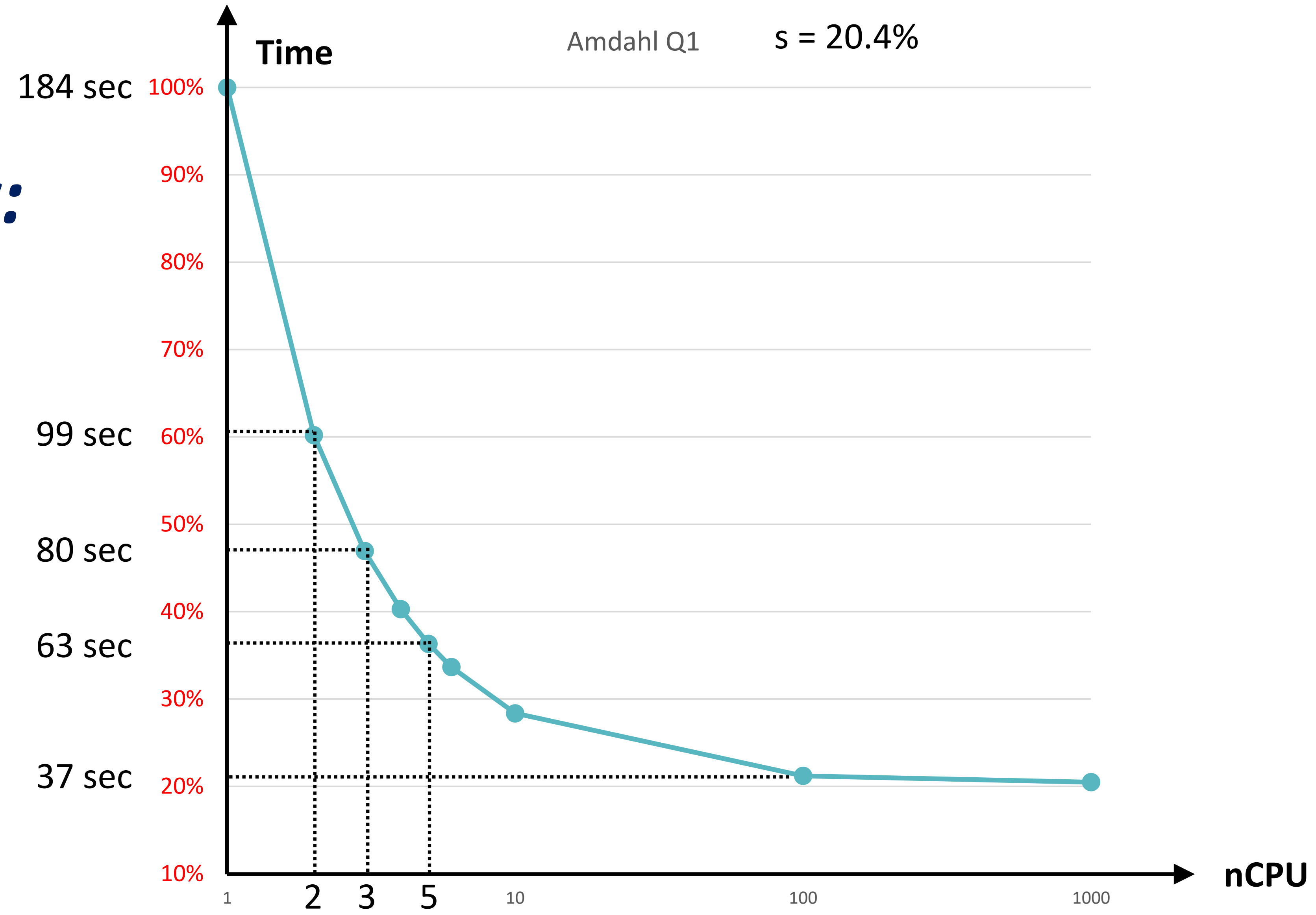
	1 GB database			10 GB database			100 GB database													1 TB database	
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
TPC-H Query	Anatella runtime [sec]	spark 6CPU runtime [sec]	Anatella speed-up	Anatella runtime [sec]	spark 6CPU runtime [sec]	Anatella speed-up	Anatella runtime [sec]	spark 1CPU runtime [sec]	spark 2CPU runtime [sec]	spark 3CPU runtime [sec]	spark 4CPU runtime [sec]	spark 5CPU runtime [sec]	spark - 6CPU runtime [sec]	Spark incom-pressible time "s t_1" [sec]	Anatella Speedup vs Spark 1 CPU (=J/I)	Anatella Speedup vs Spark 6 CPU (=O/I)	Anatella Speedup vs Spark infinite CPU (=P/I)	Spark incom-pressible time "s" (=P/J) [%]	"Anatella Time" / "Spark Time 1CPU" (=I/P) [%]	Anatella runtime [sec]	Anatella RAM usage [MByte]
Q1	0.72	17	23	3.70	22	6	27.1	184	99	80	74	63	59	37.4	6.8	2.2	1.4	20.4 %	14.8 %	260	204
Q2	0.16	27	176	0.70	218	310	5.7	956	792	751	700	688	686	649.0	167.2	120.0	113.5	67.9 %	0.6 %	61.5	800
Q3	0.70	19	27	3.72	126	34	34.5	929	732	727	651	643	932	571.9	26.9	27.0	16.6	61.6 %	3.7 %	360	10053
Q4	0.72	20	28	3.70	37	10	33.7	830	436	410	349	350	738	229.5	24.6	21.9	6.8	27.7 %	4.1 %	337	160
Q5	0.70	160	228	4.70	234	50	43.7	2275	1208	1123	994	994	1516	673.6	52.0	34.7	15.4	29.6 %	1.9 %	509	2045
Q6	0.22	14	64	1.20	17	14	6.2	102	55	46	39	37	35	20.7	16.4	5.6	3.3	20.3 %	6.1 %	65.3	154
Q7	0.70	231	329	3.70	214	58	45.6	1113	955	879	852	831	810	761.7	24.4	17.8	16.7	68.4 %	4.1 %	760	8828
Q8	0.70	190	270	4.22	208	49	49.3	1621	1312	1266	1147	1131	1124	1009.3	32.9	22.8	20.5	62.3 %	3.0 %	511	1576
Q9	2.70	283	105	17.73	111	6	200.0	2059	2064	1524	1389	1348	1366	1227.4	10.3	6.8	6.1	59.6 %	9.7 %	2668	7392
Q10	1.22	194	159	4.20	76	18	38.9	1035	849	805	756	766	758	698.6	26.6	19.5	17.9	67.5 %	3.8 %	394	2169
Q11	0.14	91	645	0.72	44	61	4.2	441	365	359	338	320	329	303.9	104.2	77.7	71.8	68.9 %	1.0 %	32.8	2192
Q12	0.70	20	28	3.22	20	6	47.7	454	349	334	306	301	299	262.8	9.5	6.3	5.5	57.9 %	10.5 %	284	1161
Q13	2.20	138	62	13.22	27	2	105.6	377	256	204	203	185	182	142.8	3.6	1.7	1.4	37.9 %	28.0 %	1109	1186
Q14	0.16	15	95	0.39	16	40	3.2	373	317	322	284	295	286	275.4	115.9	88.8	85.5	73.8 %	0.9 %	37.3	257
Q15	0.14	18	126	1.20	21	18	9.7	error	error	593	error	568	563	error				error		112	2528
Q16	0.39	173	442	3.20	84	26	31.4	839	698	671	647	643	637	587.3	26.7	20.3	18.7	70.0 %	3.7 %	280	11636
Q17	0.39	20	52	2.70	27	10	26.7	1255	972	889	862	779	763	664.8	46.9	28.5	24.9	53.0 %	2.1 %	646	525
Q18	0.72	21	29	4.20	33	8	36.9	1135	943	857	814	802	785	717.4	30.7	21.3	19.4	63.2 %	3.3 %	408	8672
Q19	0.70	15	21	4.70	17	4	44.1	972	331	312	295	290	287	119.2	22.0	6.5	2.7	12.3 %	4.5 %	492	188
Q20	0.39	41	105	1.70	44	26	21.7	972	803	744	732	737	698	643.2	44.7	32.1	29.6	66.2 %	2.2 %	314	885
Q21	1.70	578	339	12.72	204	16	127.7	3815	2976	2912	2629	2611	2469	2235.4	29.9	19.3	17.5	58.6 %	3.3 %	330	329
Q22	0.72	17	23	4.20	24	6	44.5	206	153	153	140	130	128	112.3	4.6	2.9	2.5	54.5 %	21.6 %	595	2027

Part 3:
Amdahl's Law and incompressible times

Amdahl's Law: Example: TPC-H Q1

(100 GB database)

X axis: number of CPU's
Y axis: Runtime [%]

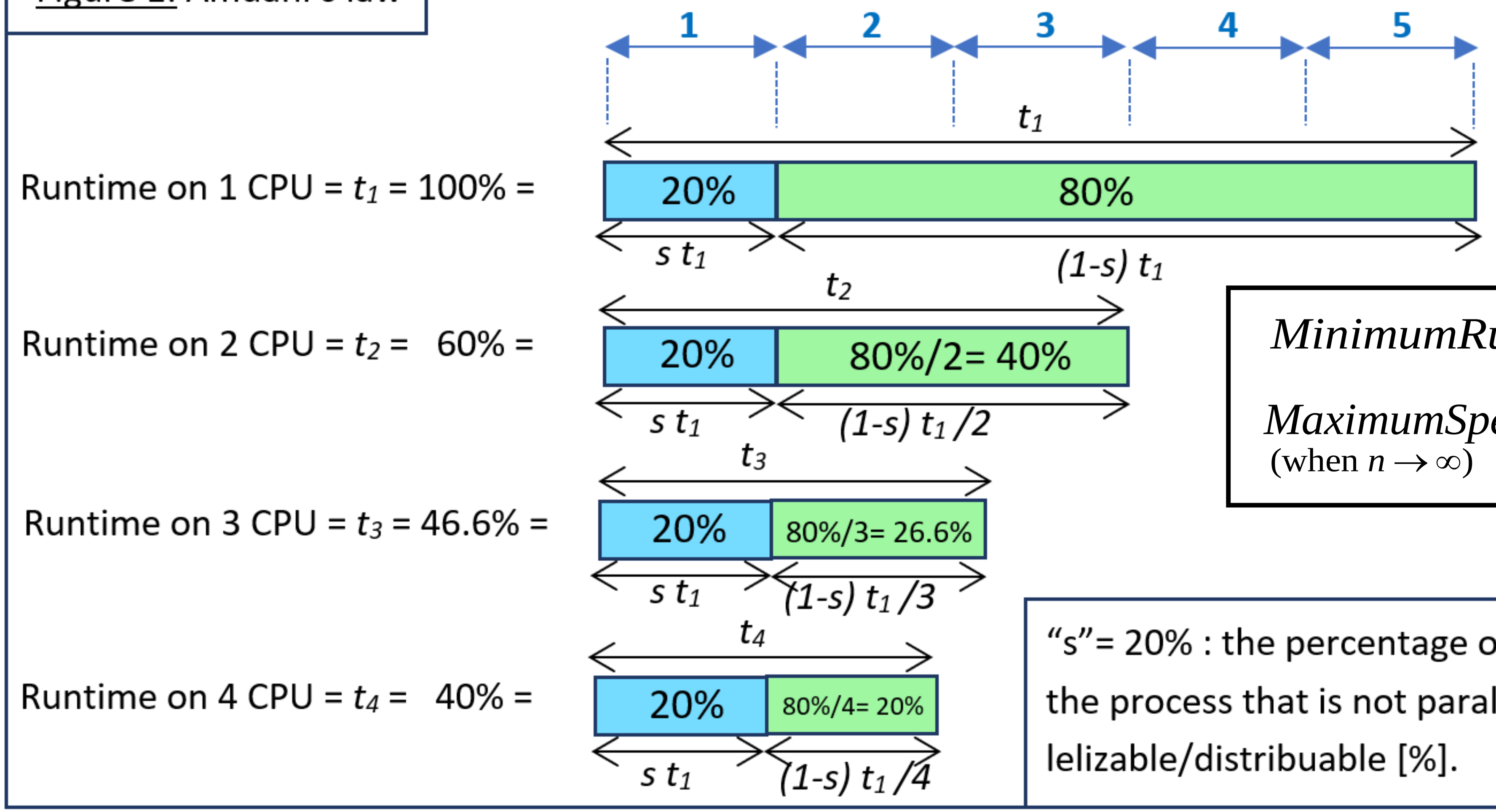


integrated data mining

Amdahl's Law for distributed computations 1/2

Amdahl's Law: Total running time= incompressible time (s) + $\dfrac{\text{compressible time } (1-s)}{\text{number of CPU } (n)}$ = $s + \dfrac{1-s}{n}$

Figure 1: Amdahl's law



$$MinimumRuntime\,[\%] = s \quad (= 20\%)$$

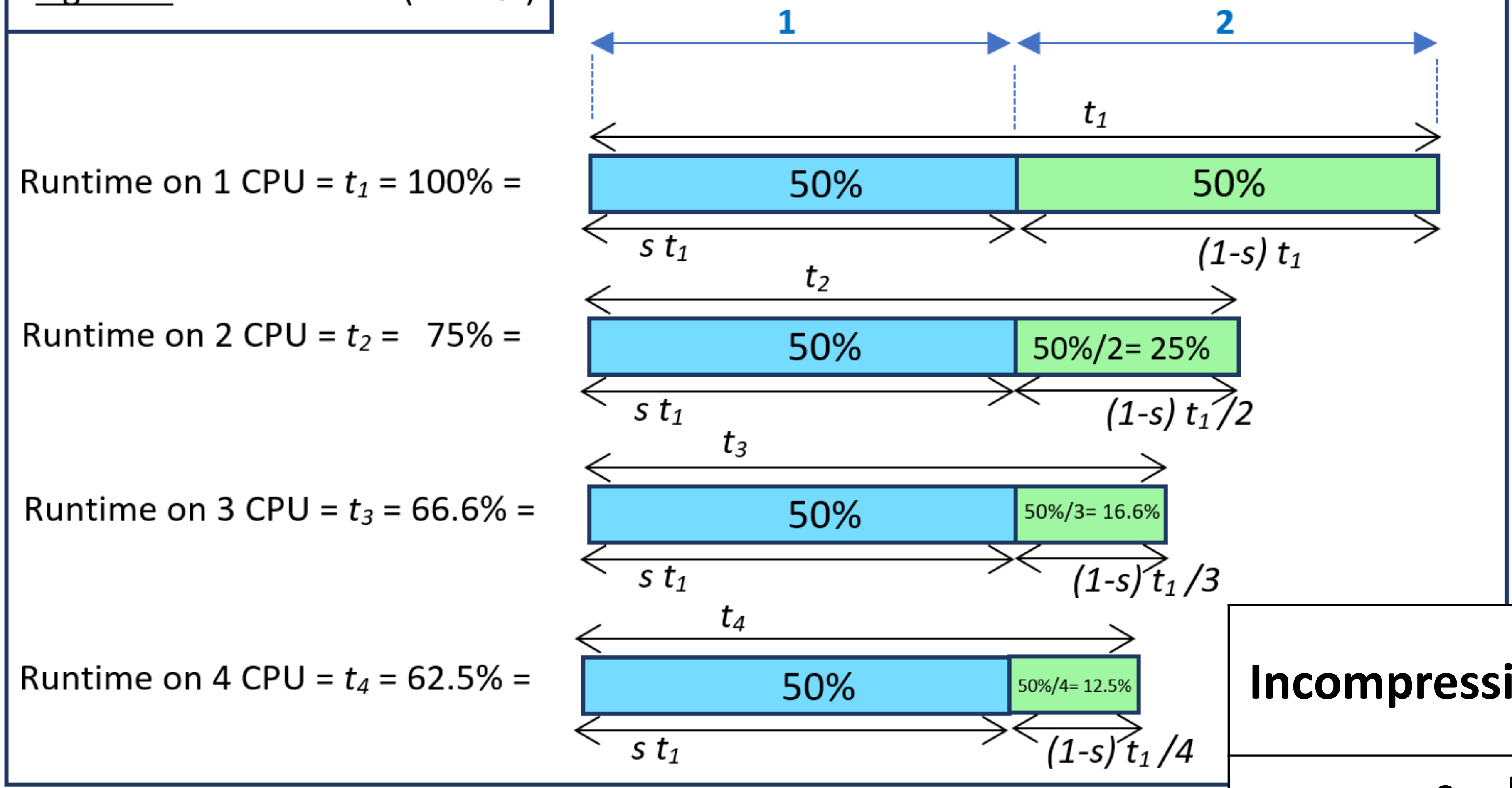
$$MaximumSpeedUp = 1/s \quad \left(= 1/20\% = 5\right)$$
(when $n \to \infty$)

"s"= 20% : the percentage of the process that is not parallelizable/distribuable [%].

timi integrated data mining

Amdahl's Law for distributed computations 1/2

Figure 2: Amdahl's law (s= 50%)



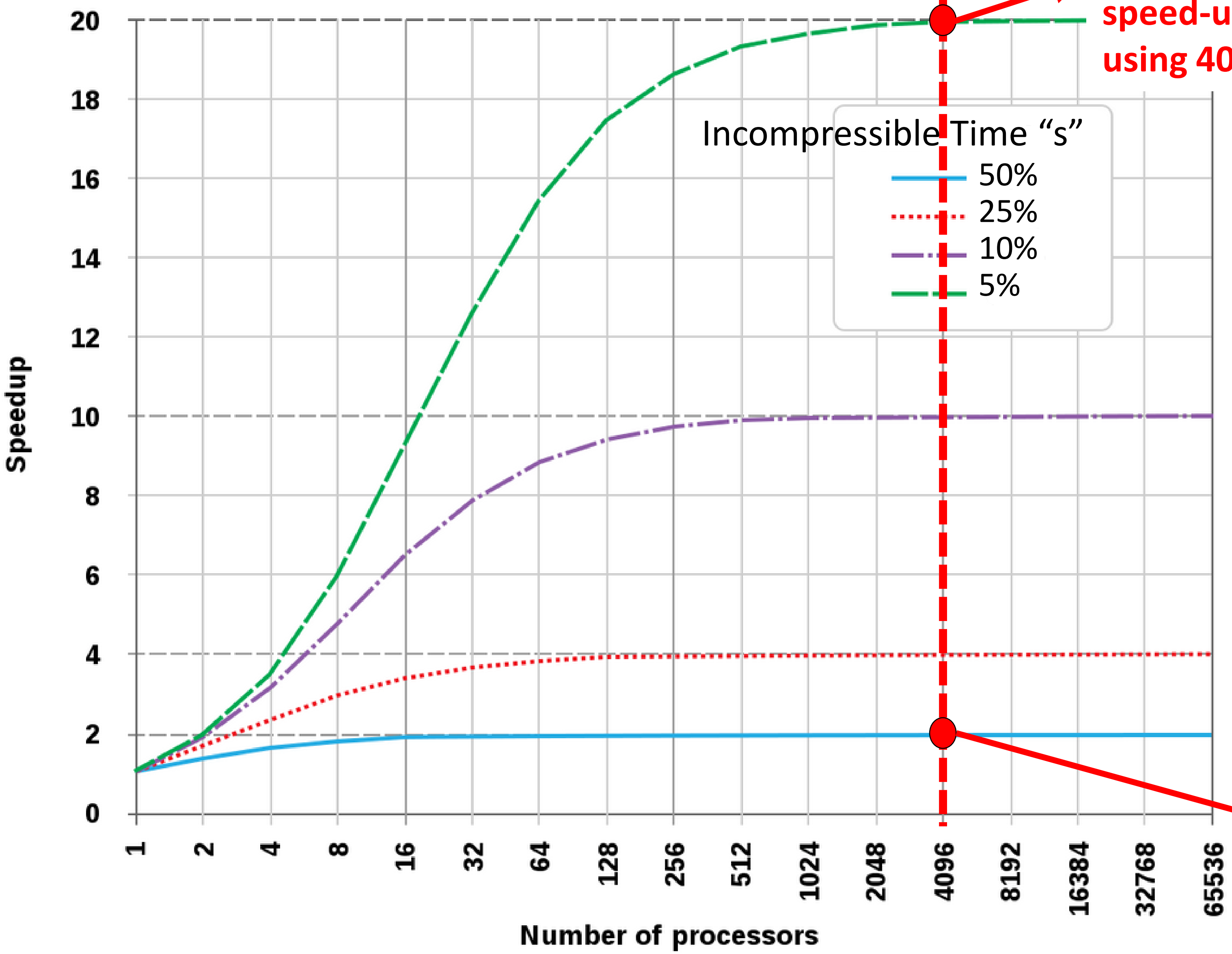
This is important: When "s">=50, it means a "failure" of the (distributed) computation engine

Incompressible Time "s"	Maximum SpeedUp
s = 50%	2
s = 20%	5

From the previous slide

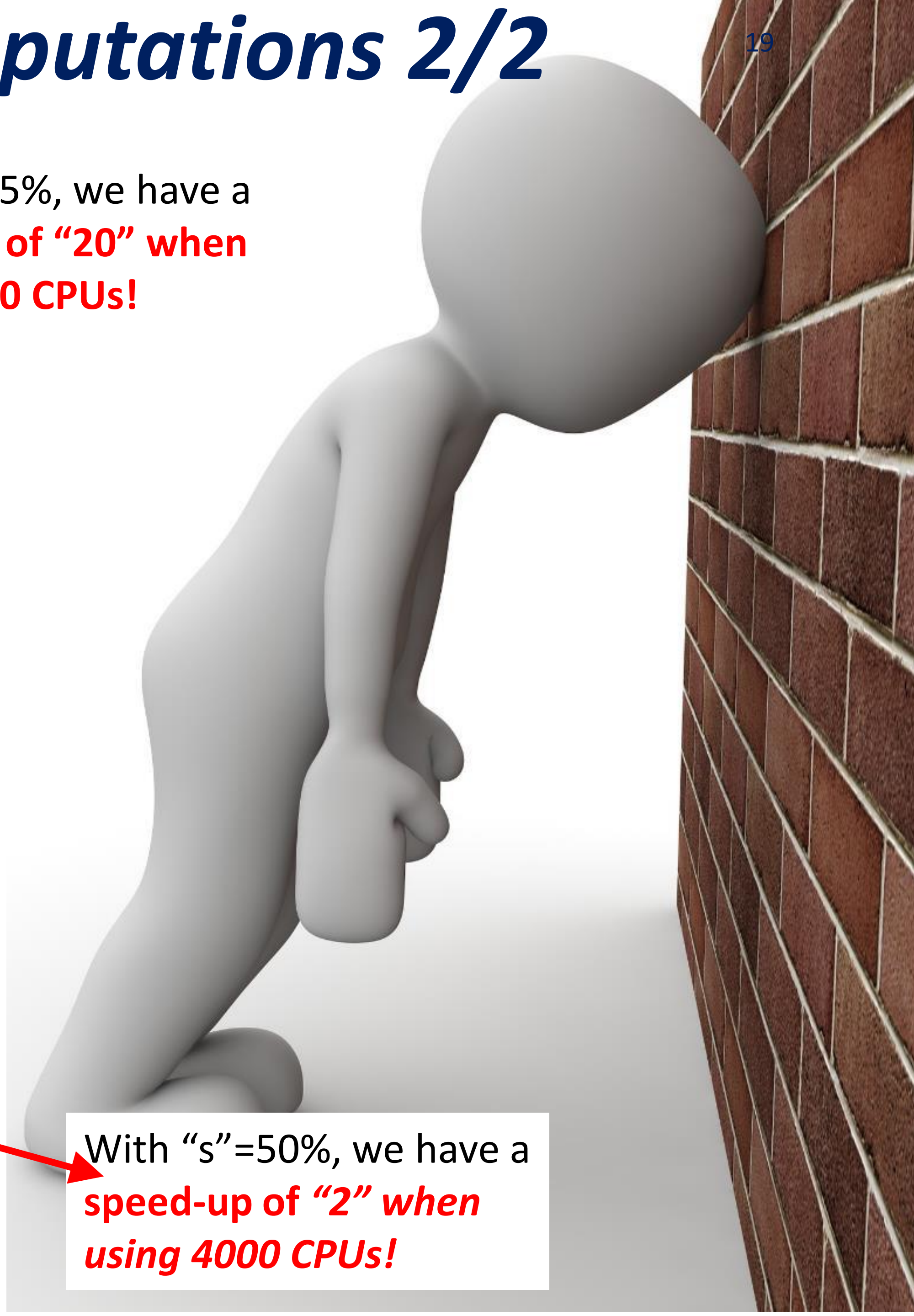
Amdahl's Law for distributed computations 2/2

Amdahl's Law: Total running time = $S + \frac{1-s}{n}$



With "s"=5%, we have a **speed-up of "20" when using 4000 CPUs!**

With "s"=50%, we have a ***speed-up of "2" when using 4000 CPUs!***



Part 5: Deep dive into the benchmark results

Deep dive: Q13: How to estimate “s”?

X axis: number of CPU’s
Y axis: Runtime [%]

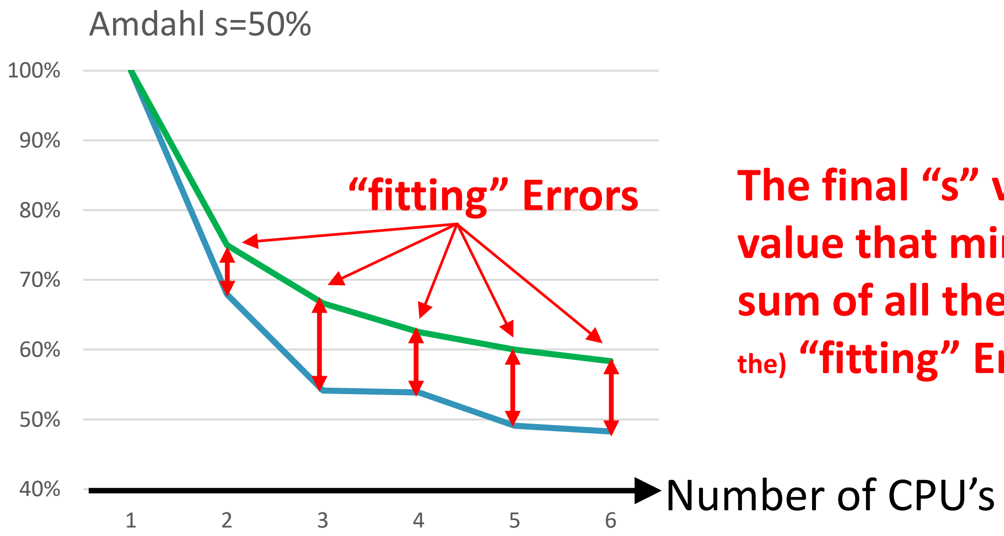
https://en.wikipedia.org/wiki/Inverse_problem

“s” is the “incompressible” time in [%]

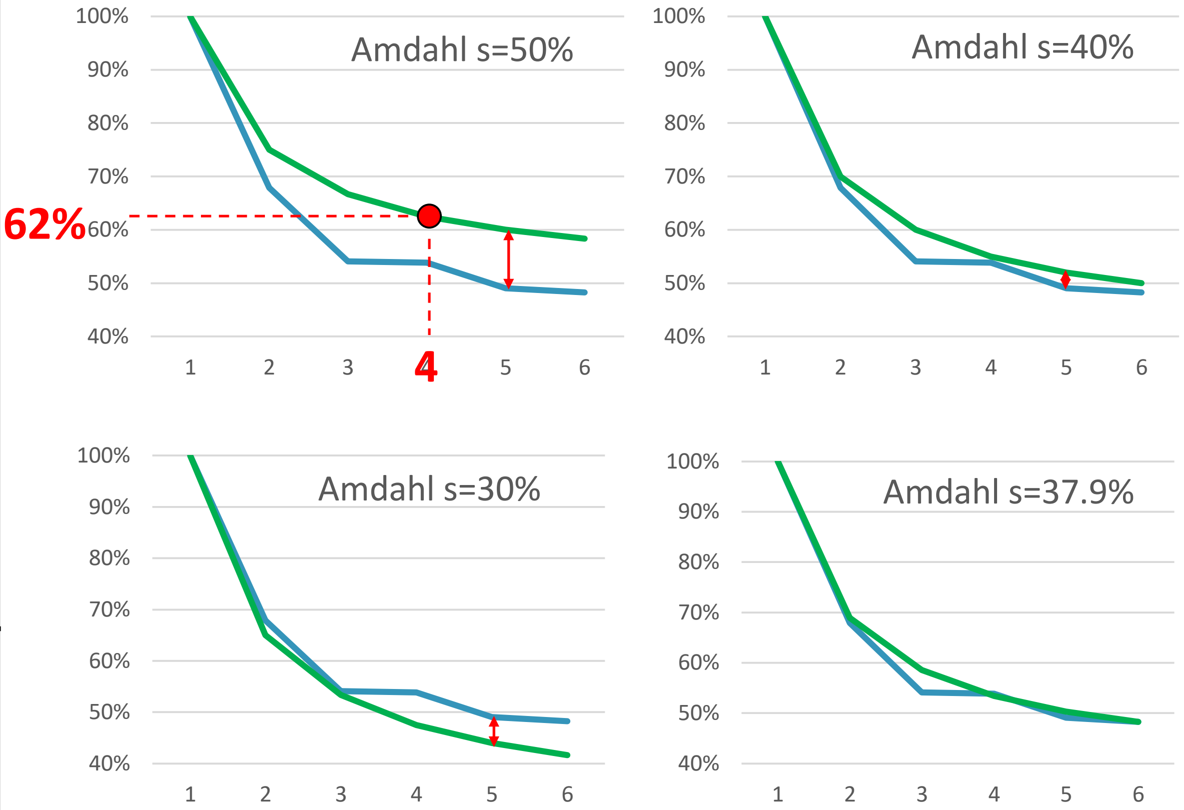
Number of CPU’s	1	2	3	4	5	6
Q13 Measured Spark time [sec]	377	256	204	203	185	182
Q13 Measured Spark time [%]	**100%**	**68%**	**54%**	**54%**	**49%**	**48%**
Amdahl s=50%	100%	75%	67%	63%	60%	58%
Amdahl s=40%	100%	70%	60%	55%	52%	50%
Amdahl s=30%	100%	65%	53%	48%	44%	42%
Amdahl s=37.9%	100%	69%	59%	53%	50%	48%

… where we used “Amdahl’s Law”:

$$\boxed{RunTime\ on\ n\ CPU[\%] = s + (1-s)/n}$$



The final “s” value is the value that minimizes the sum of all the (absolute value of the) **“fitting” Errors**



Blue: real measures of the runtime on Q13

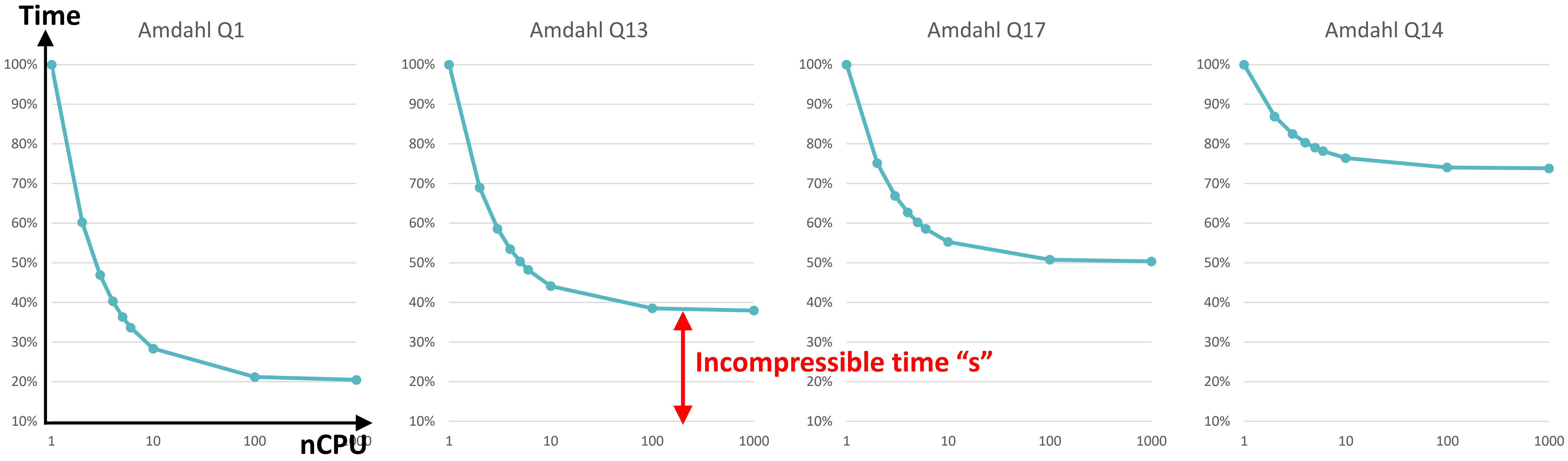
Green: runtime computed using the Amdahls’s law for different values of “s”

Red: one fraction of the global “fitting” error

Details : https://github.com/Kranf99/TPC-H-Benchmarck-Anatella-Spark
Precisely: inside the file “compute_incompressible_time_s_v2.anatella”
STEP: http://download.timi.eu/docs/Global_optimization_algorithm_STEP.pdf

Amdahl's Law: Examples

X axis: number of CPU's
Y axis: Runtime [%]



s = 20.4%

s = 37.9%

s = 53%

s = 73.8%

On the query Q17, Spark fails because s>50%

timi
integrated data mining

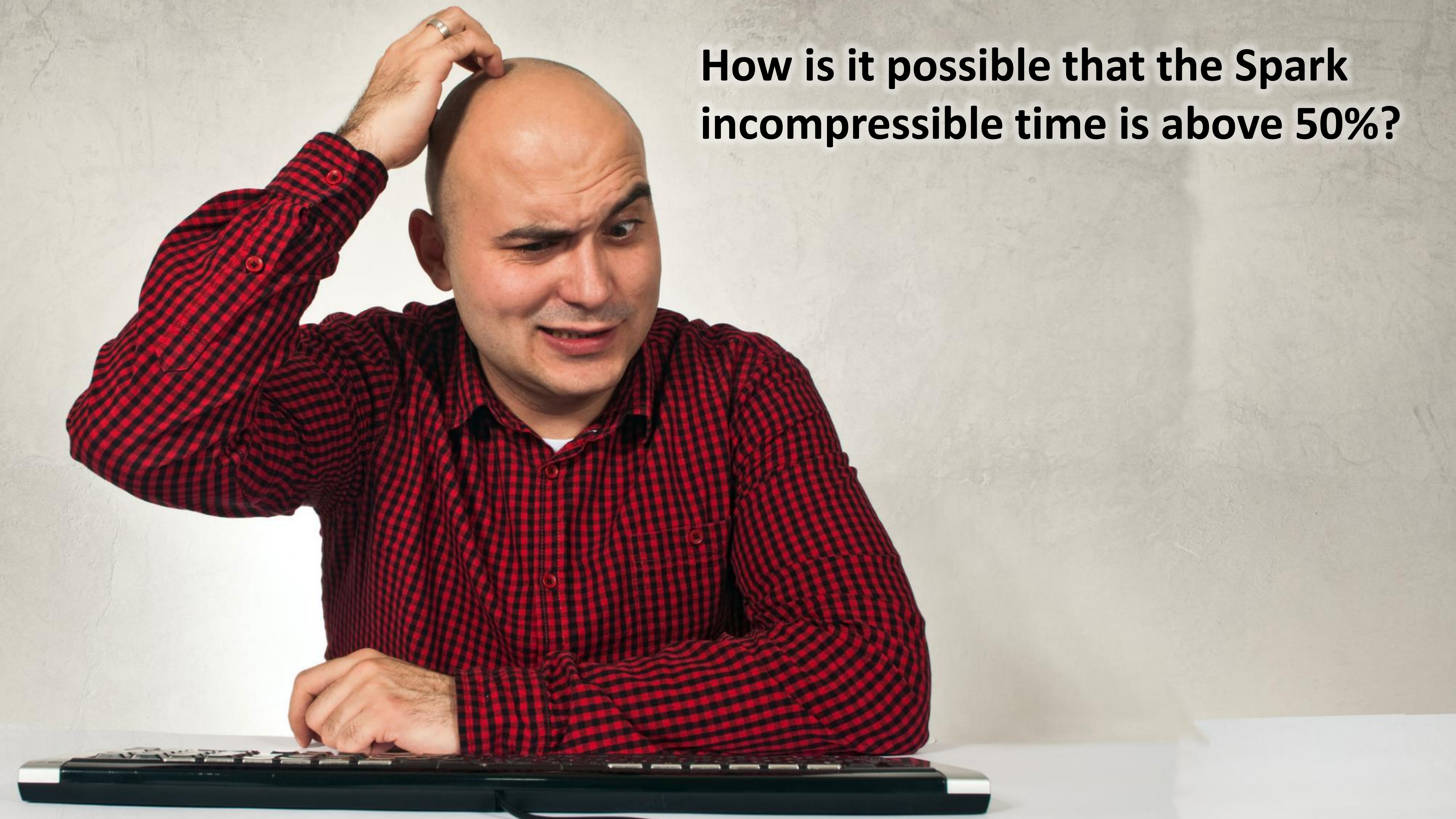
Part 4: Timing results and incompressible times

The Spark incompressible runtime “s”: The Harsh Truth

TPC-H Query	Spark incompressible time s [%]	Spark incompressible time “s” [sec]	Anatella runtime [sec]	Anatella Speedup vs Spark infinite CPU
Q1	20.4%	37.4	27.1	1.4
Q2	67.9%	649	5.7	113.5
Q3	61.6%	571.9	34.5	16.6
Q4	27.7%	229.5	33.7	6.8
Q5	29.6%	673.6	43.7	15.4
Q6	20.3%	20.7	6.2	3.3
Q7	68.4%	761.7	45.6	16.7
Q8	62.3%	1009.3	49.3	20.5
Q9	59.6%	1227.4	200	6.1
Q10	67.5%	698.6	38.9	17.9
Q11	68.9%	303.9	4.2	71.8
Q12	57.9%	262.8	47.7	5.5
Q13	37.9%	142.8	105.6	1.4
Q14	73.8%	275.4	3.2	85.5
Q15	error	error	9.7	
Q16	70.0%	587.3	31.4	18.7
Q17	53.0%	664.8	26.7	24.9
Q18	63.2%	717.4	36.9	19.4
Q19	12.3%	119.2	44.1	2.7
Q20	66.2%	643.2	21.7	29.6
Q21	58.6%	2235.4	127.7	17.5
Q22	54.5%	112.3	44.5	2.5

s > 50%

ALWAYS >1


How is it possible that the Spark
incompressible time is above 50%?

Part 6: Other benchmarks results

Could it be luck?

Table 2

TPCx-BB (Big-Bench) Query	SF1000 incom-pressible time s" [%]	SF3000 incom-pressible time "s" [%]
Q1	57 %	43 %
Q2	21 %	20 %
Q3	34 %	30 %
Q4	22 %	22 %
Q6	30 %	20 %
Q8	42 %	29 %
Q10	89 %	75 %
Q11	44 %	35 %
Q12	38 %	26 %
Q13	32 %	24 %
Q14	61 %	37 %
Q15	77 %	57 %
Q16	23 %	16 %
Q17	87 %	74 %
Q18	85 %	56 %
Q19	95 %	84 %
Q21	59 %	33 %
Q22	67 %	41 %
Q24	42 %	31 %
Q29	24 %	15 %
Q30	17 %	16 %

s_{min}

"Amdahl's Law in Big Data Analytics: Alive and Kicking in TPCx-BB (BigBench)".
IEEE International Symposium on High Performance, 2018

s > 50%

s < 20%

Results are **consistent** with published litterature

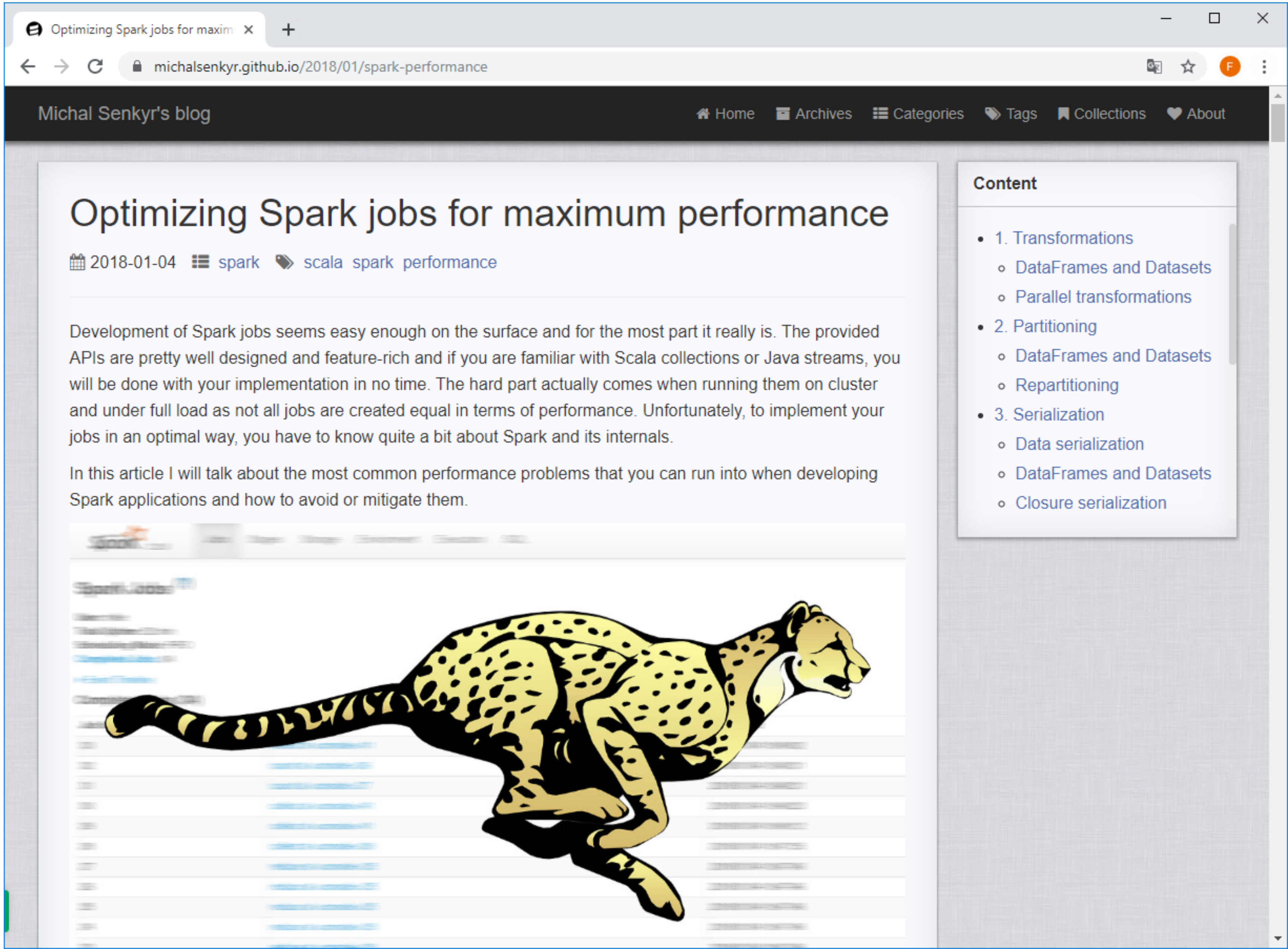
Maximum Speed-up: $\frac{1}{s_{\min}} = \frac{1}{0.15} = 6.6$

timi
integrated data mining

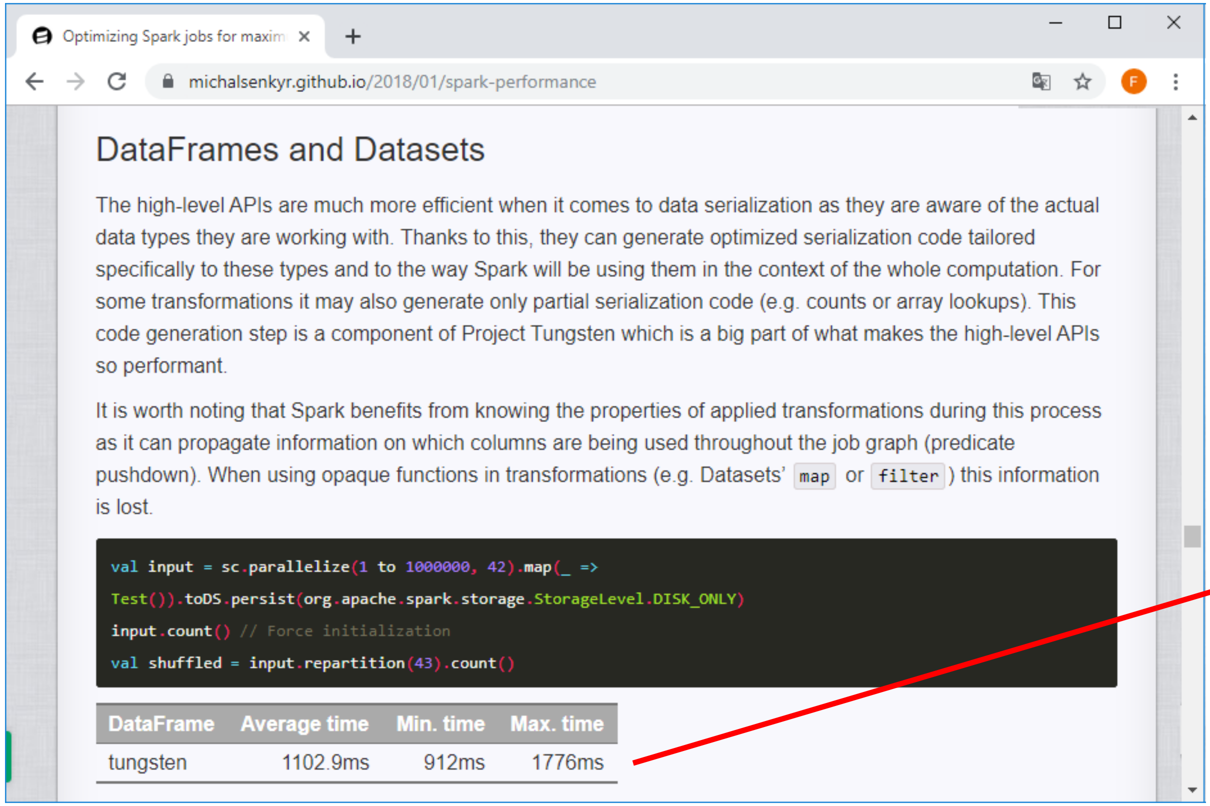
Spark “tuning” for maximum performance 1/2

Many thanks to
Savvas Savvides (savvas@purdue.edu) from the Purdue University for providing the optimized Spark/Scala code!

Blog about “tuning” spark:
https://michalsenkyr.github.io/2018/01/spark-performance



Spark “tuning” for maximum performance 2/2


Optimizing Spark jobs for maxim

michalsenkyr.github.io/2018/01/spark-performance

DataFrames and Datasets

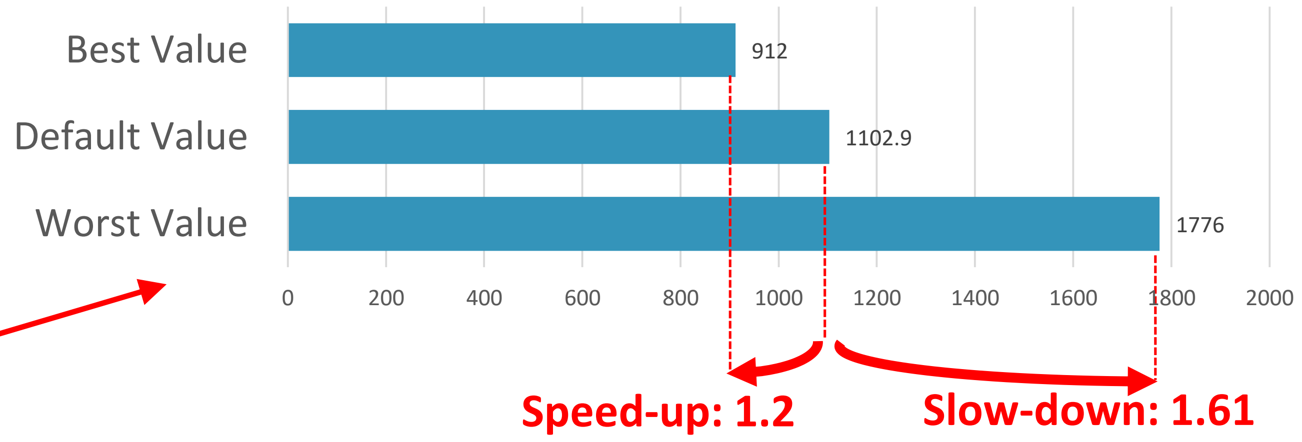
The high-level APIs are much more efficient when it comes to data serialization as they are aware of the actual data types they are working with. Thanks to this, they can generate optimized serialization code tailored specifically to these types and to the way Spark will be using them in the context of the whole computation. For some transformations it may also generate only partial serialization code (e.g. counts or array lookups). This code generation step is a component of Project Tungsten which is a big part of what makes the high-level APIs so performant.

It is worth noting that Spark benefits from knowing the properties of applied transformations during this process as it can propagate information on which columns are being used throughout the job graph (predicate pushdown). When using opaque functions in transformations (e.g. Datasets' `map` or `filter`) this information is lost.

```
val input = sc.parallelize(1 to 1000000, 42).map(_ =>
Test()).toDS.persist(org.apache.spark.storage.StorageLevel.DISK_ONLY)
input.count() // Force initialization
val shuffled = input.repartition(43).count()
```

DataFrame	Average time	Min. time	Max. time
tungsten	1102.9ms	912ms	1776ms

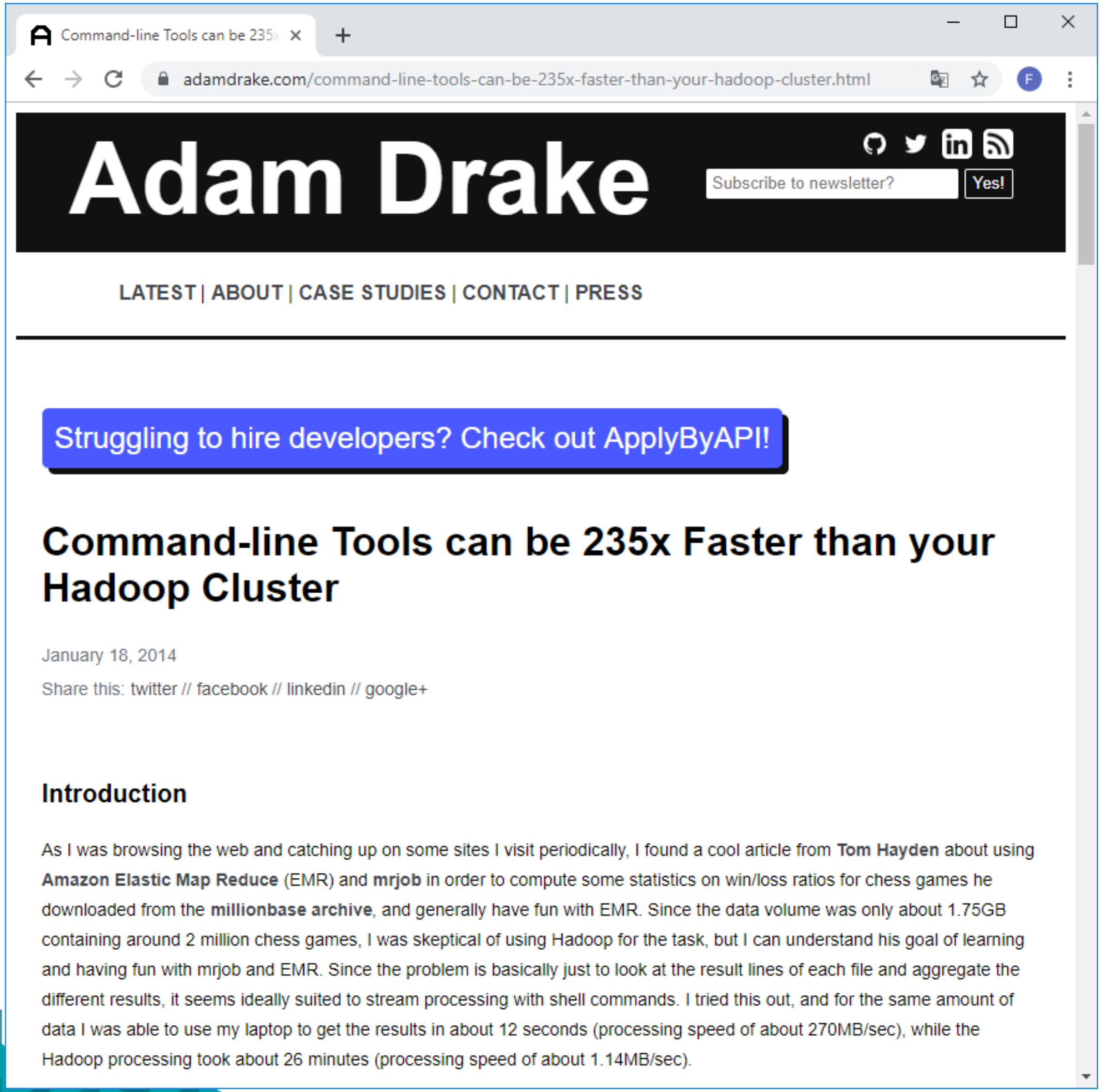
Running Times in function of meta-parameter (Smaller is Better)



In the best scenario, optimizing everything, you can expect to have **a speed-up of maximum 1.5** compared to the default values.

Chess benchmark 1/3

https://adamdrake.com/command-line-tools-can-be-235x-faster-than-your-hadoop-cluster.html



Command-line Tools can be 235x

adamdrake.com/command-line-tools-can-be-235x-faster-than-your-hadoop-cluster.html

Adam Drake

Subscribe to newsletter? Yes!

LATEST | ABOUT | CASE STUDIES | CONTACT | PRESS

Struggling to hire developers? Check out ApplyByAPI!

Command-line Tools can be 235x Faster than your Hadoop Cluster

January 18, 2014

Share this: twitter // facebook // linkedin // google+

Introduction

As I was browsing the web and catching up on some sites I visit periodically, I found a cool article from **Tom Hayden** about using **Amazon Elastic Map Reduce** (EMR) and **mrjob** in order to compute some statistics on win/loss ratios for chess games he downloaded from the **millionbase archive**, and generally have fun with EMR. Since the data volume was only about 1.75GB containing around 2 million chess games, I was skeptical of using Hadoop for the task, but I can understand his goal of learning and having fun with mrjob and EMR. Since the problem is basically just to look at the result lines of each file and aggregate the different results, it seems ideally suited to stream processing with shell commands. I tried this out, and for the same amount of data I was able to use my laptop to get the results in about 12 seconds (processing speed of about 270MB/sec), while the Hadoop processing took about 26 minutes (processing speed of about 1.14MB/sec).

Chess benchmark 2/3

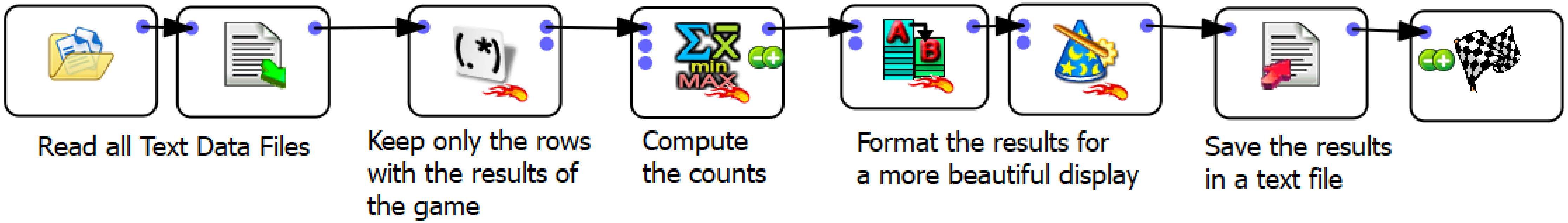
https://adamdrake.com/command-line-tools-can-be-235x-faster-than-your-hadoop-cluster.html

Objective: Count the different game results in a chess text-file database of 3.46GB

[Event "URS-ch31"]
[Site "Leningrad RUS"]
[Date "1963.11.??"]
[Round "16"]
[White "Kholmov, Ratmir D."]
[Black "Zakharov, Alexander V."]
[Result "1-0"]
[ECO "B36j"]
[PlyCount "65"]

(moves from the game follow...)

Game Outcome	count
[Result "1/2-1/2"]	1782291
[Result "1-0"]	1888992
[Result "0-1"]	1383030

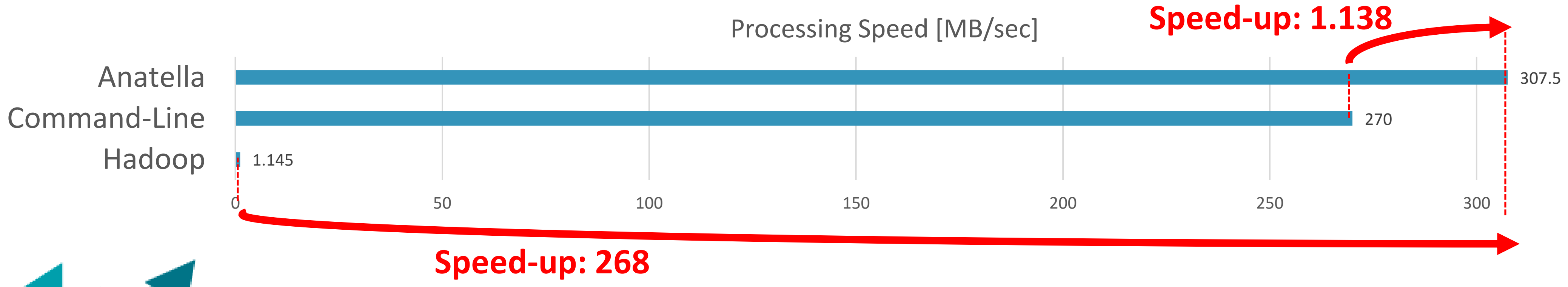


Chess benchmark 3/3

https://adamdrake.com/command-line-tools-can-be-235x-faster-than-your-hadoop-cluster.html

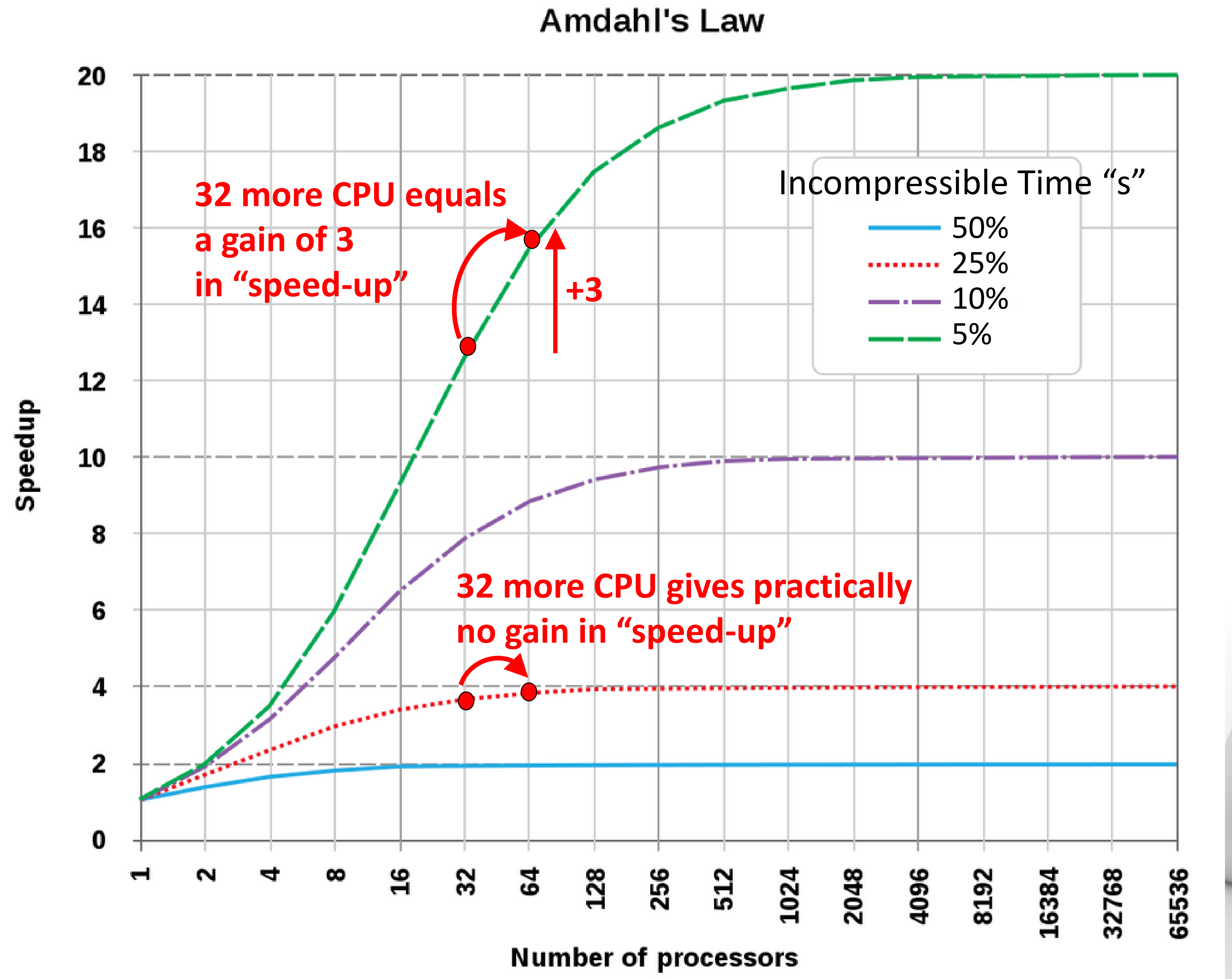
Adam Drake writes: “*…for the same amount of data* (3.46GB in 140 files) *I was able to use my laptop to get the results in about 12 seconds (processing speed of about 270MB/sec), while the Hadoop processing took about 26 minutes (processing speed of about 1.14MB/sec).*”

	Running on	Run-Time	Processing Speed	Relative Speed
Hadoop	7 nodes (c1.medium) on AWS	26 minutes	1.145 MB/sec	1
Shell	1 portable PC (unknown brand)	12.8 seconds	270 MB/sec	235
Anatella	1 Portable PC (MSI-WS65)	11.25 seconds	307.5 MB/sec	268



Benchmark Sources on https://github.com/Kranf99/Chess-Benchmark

Amdahl's Law for distributed computations 2/2

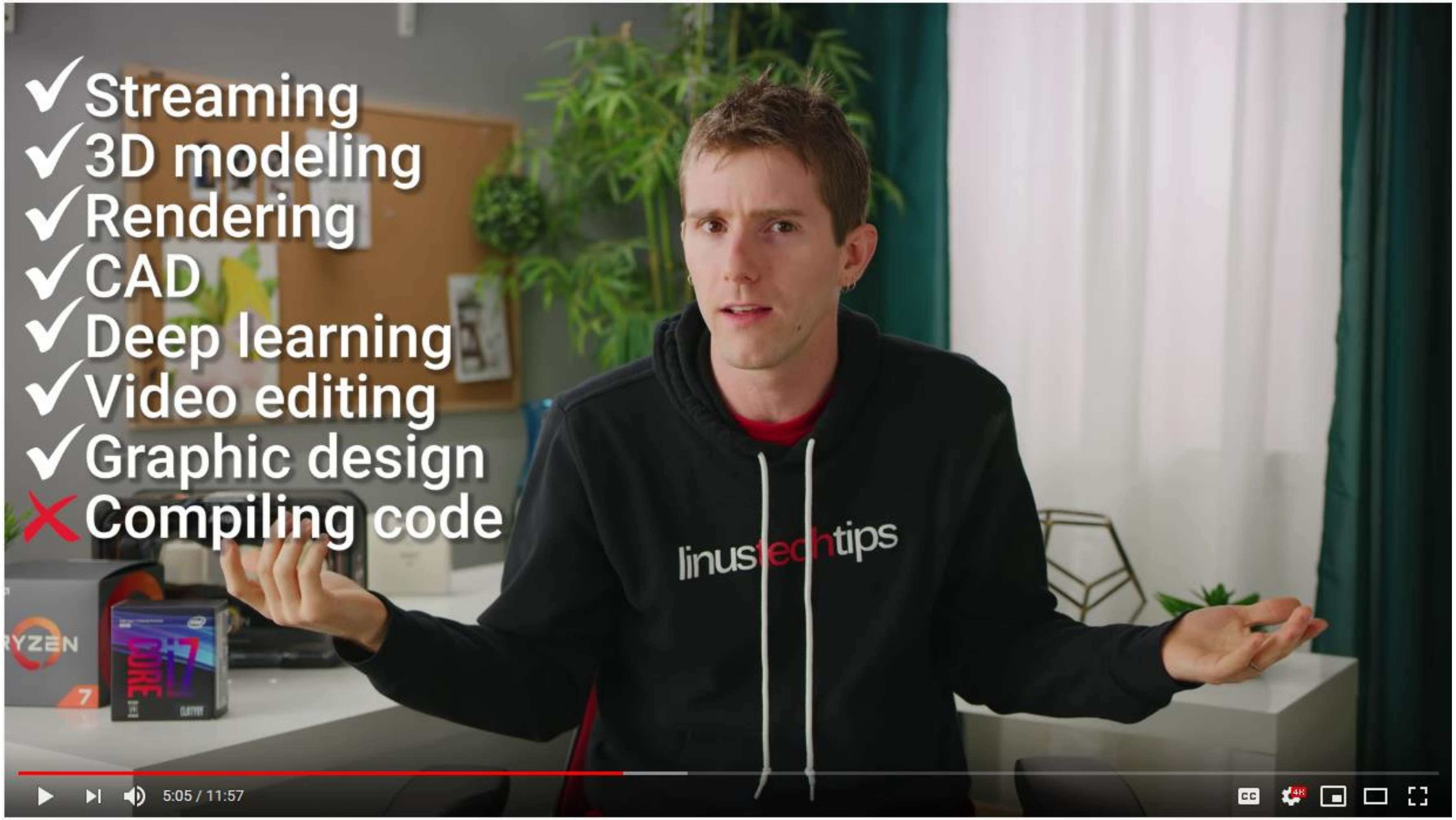


Choose a new CPU for your next laptop?!

https://www.youtube.com/watch?v=DEw-3vpqhbQ

Intel Killed their OWN Product Lineup – Core i9 vs Xeon

Choose a new CPU for your next laptop?!



Choose a new CPU for your next laptop?!

	Core i7 9700K	Core i9 9900K	Core i7 8700K
Cores / Threads	8c / 8t	8c / 16t	6c / 12t
Base Clock	3.6 GHz	3.6 GHz	3.7 GHz
Boost Clock	4.9 GHz	5.0 GHz	4.7 GHz
Manufacturing Process	14 nm	14 nm	
MSRP	$409	$549 Good Luck finding any at this price atm	$358

Lower Core Count + Higher Frequency:

CPU's names ending with a "K"

For "**difficult** to parallelize" tasks:

- Office (Word, Excel, etc.)
- Video games
- 95% of Machine Learning algorithms.

Higher Core Count + Lower Frequency:

CPU's names ending with a "X"

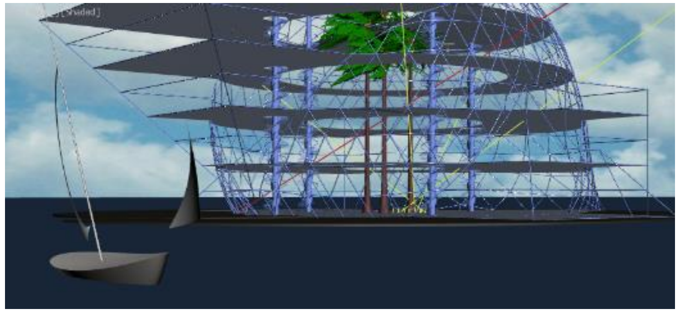
For "**easy** to parallelize" tasks:

- 3D rendering
- 2D video compression/production
- ~~Machine learning~~

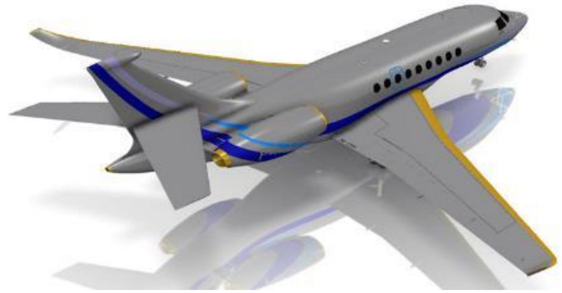
	Core i7-9800X	Core i9-9820X	Core i9-9900X	Core i9-9920X	Core i9-9940X	Core i9-9960X	Core i9-9980XE
Cores / Threads	8c / 16t	10c / 20t	10c / 20t	12c / 24t	14c / 28t	16c / 32t	18c / 36t
Base Clock	3.8 GHz	3.3 GHz	3.5 GHz	3.5 GHz	3.3 GHz	3.1 GHz	3.0 GHz
Boost Clock	4.4 GHz 4.5 GHz TBM	4.1 GHz 4.2 GHz TBM	4.4 GHz 4.5 GHz TBM	4.4 GHz 4.5 GHz TBM	4.4 GHz 4.5 GHz TBM	4.4 GHz 4.5 GHz TBM	4.4 GHz 4.5 GHz TBM
Cache	16.5 MB	16.5 MB	19.25 MB	19.25 MB	19.25 MB	22 MB	24.75 MB
Price per 1k	$599	$899	$999	$1,199	$1,399	$1,699	$1,999

Choose a new CPU for your next laptop?!

3D Rendering benchmark: SPECviewperf 13 (https://www.spec.org/gwpg/gpc.static/vp13info.html)

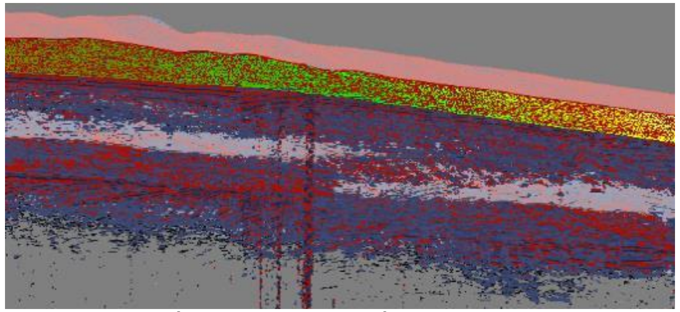


3ds Max (3dsmax-06)

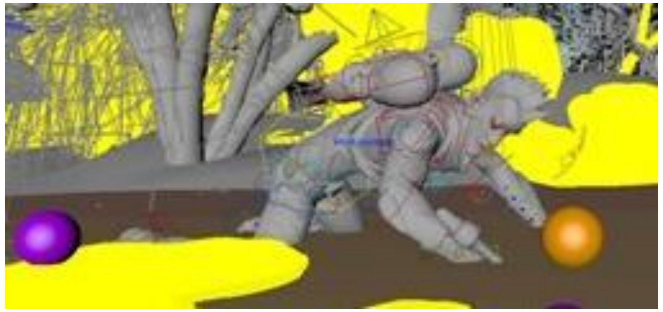


CATIA (catia-05)

Creo (creo-02)

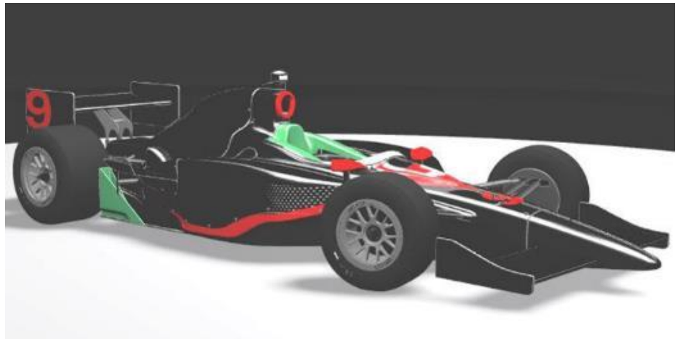


Energy (energy-02)

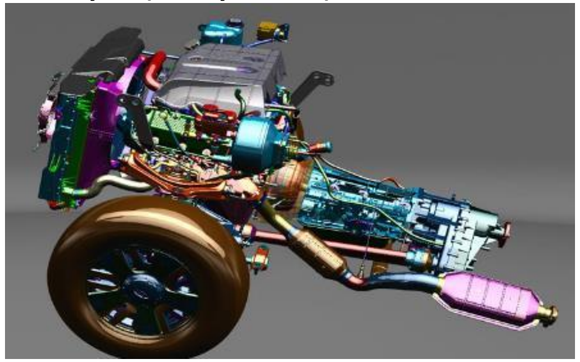


Maya (maya-05)

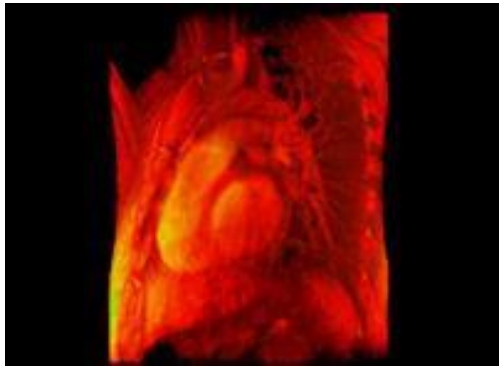
Solidworks (sw-04)



Showcase (showcase-02)



Siemens NX (snx-03)



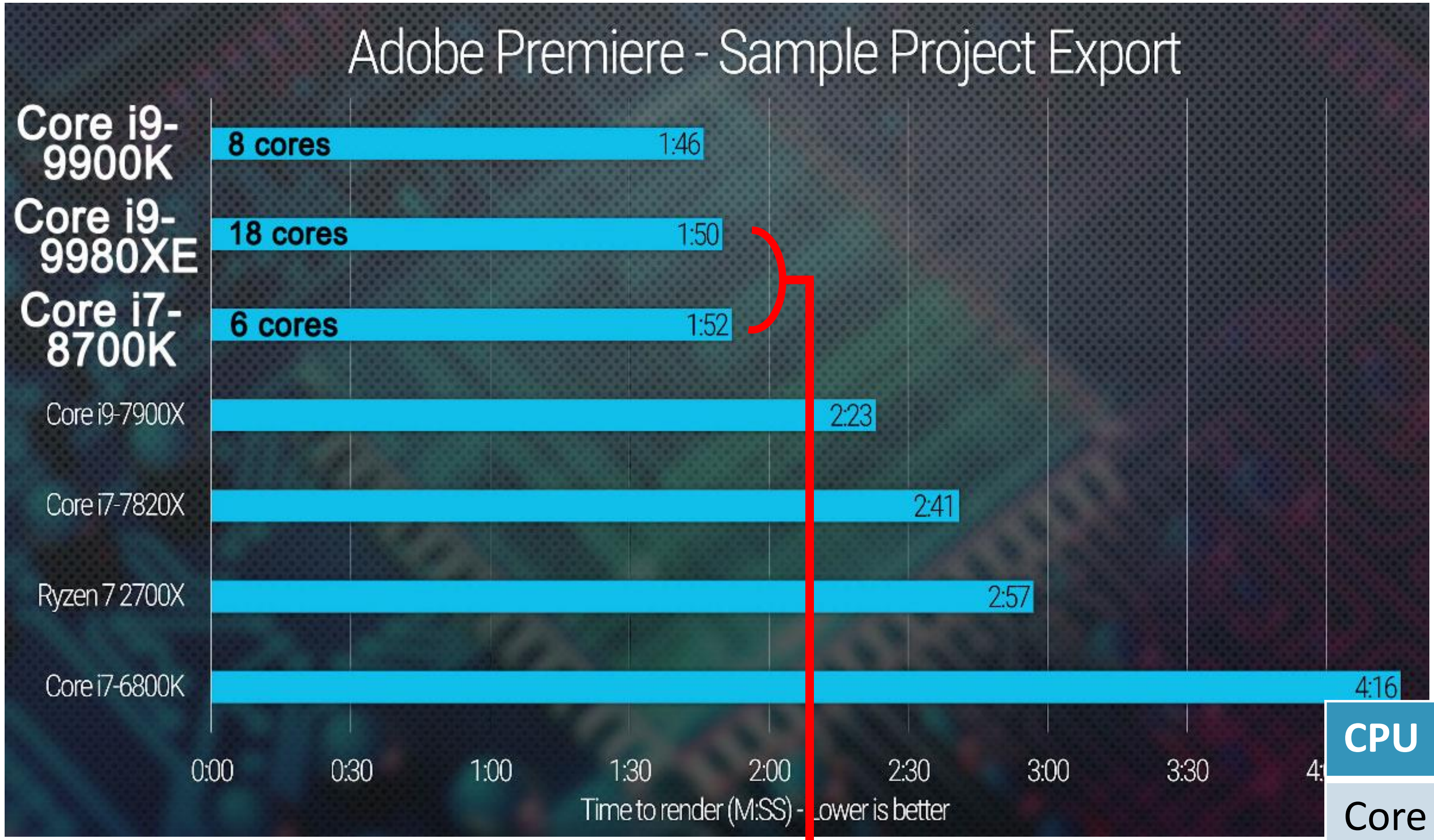
Medical (medical-02) (Heart)

Choose a new CPU for your next laptop?!

Computing Shapes & Rendering 3D images:
https://www.spec.org/gwpg/gpc.static/vp13info.html

CPU	Core counts	Frequency
Core i7-8700K	6 cores / 12 Threads	3.7 GHz
Core i9-9900K	8 cores / 16 threads	3.6 GHz
Core i9-9980XE	18 cores / 18 threads	3 GHz
Core i9-7900X	10 cores / 20 threads	3.3 GHz

Choose a new CPU for your next laptop?!



Almost the same execution time despite that one is running on 6 cores and the other is running on 18 cores!

CPU	Core counts	Frequency
Core i7-8700K	6 cores	3.7 GHz
Core i9-9900K	8 cores	3.6 GHz
Core i9-9980XE	18 cores	3 GHz
Core i9-7900X	10 cores	3.3 GHz
Nvidia 2080 Ti	4352 cuda cores	1.5 GHz

Part 7:

To distribute or not to distribute?
To parallelize or not to parallelize?

The Spark incompressible runtime “s”

TPC-H Query	Spark incompressible time s [%]	Spark incompressible time “s” [sec]	Anatella runtime [sec]	Anatella Speedup vs Spark infinite CPU
Q1	20.4%	37.4	27.1	1.4
Q2	67.9%	649	5.7	113.5
Q3	61.6%	571.9	34.5	16.6
Q4	27.7%	229.5	33.7	6.8
Q5	29.6%	673.6	43.7	15.4
Q6	20.3%	20.7	6.2	3.3
Q7	68.4%	761.7	45.6	16.7
Q8	62.3%	1009.3	49.3	20.5
Q9	59.6%	1227.4	200	6.1
Q10	67.5%	698.6	38.9	17.9
Q11	68.9%	303.9	4.2	71.8
Q12	57.9%	262.8	47.7	5.5
Q13	37.9%	142.8	105.6	1.4
Q14	73.8%	275.4	3.2	85.5
Q15	error	error	9.7	
Q16	70.0%	587.3	31.4	18.7
Q17	53.0%	664.8	26.7	24.9
Q18	63.2%	717.4	36.9	19.4
Q19	12.3%	119.2	44.1	2.7
Q20	66.2%	643.2	21.7	29.6
Q21	58.6%	2235.4	127.7	17.5
Q22	54.5%	112.3	44.5	2.5

For most of the queries (see the cells in **red** in the second column), the Spark incompressible time “s” is above 50%! Meaning that the maximum speed-up for Spark is 2, whatever the size of your cluster.

“s” [in seconds] is the time that you get when your run a query using an infinite number of CPU’s

Ratio Always >1: This means that whatever the amount of CPU used to run a query, one Anatella server will always be faster than any number of Spark servers.

This makes the whole Spark system nearly unusable since the major Spark promise (i.e. horizontal scalability: to deliver higher-speed on a larger infrastructure) is not achieved: it’s a catastrophic failure for Spark.

Distributed computations: 2 Alternatives

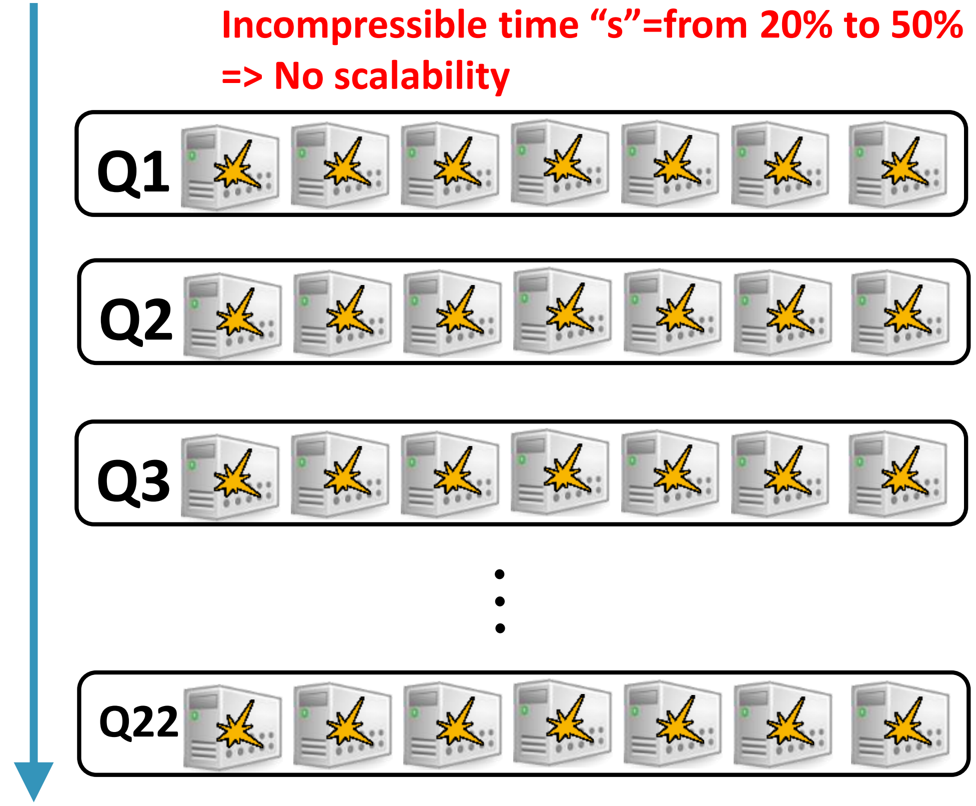
(1) One Query per Cluster

Time

For **In-Memory Tools** that needs the whole RAM of the cluster to operate

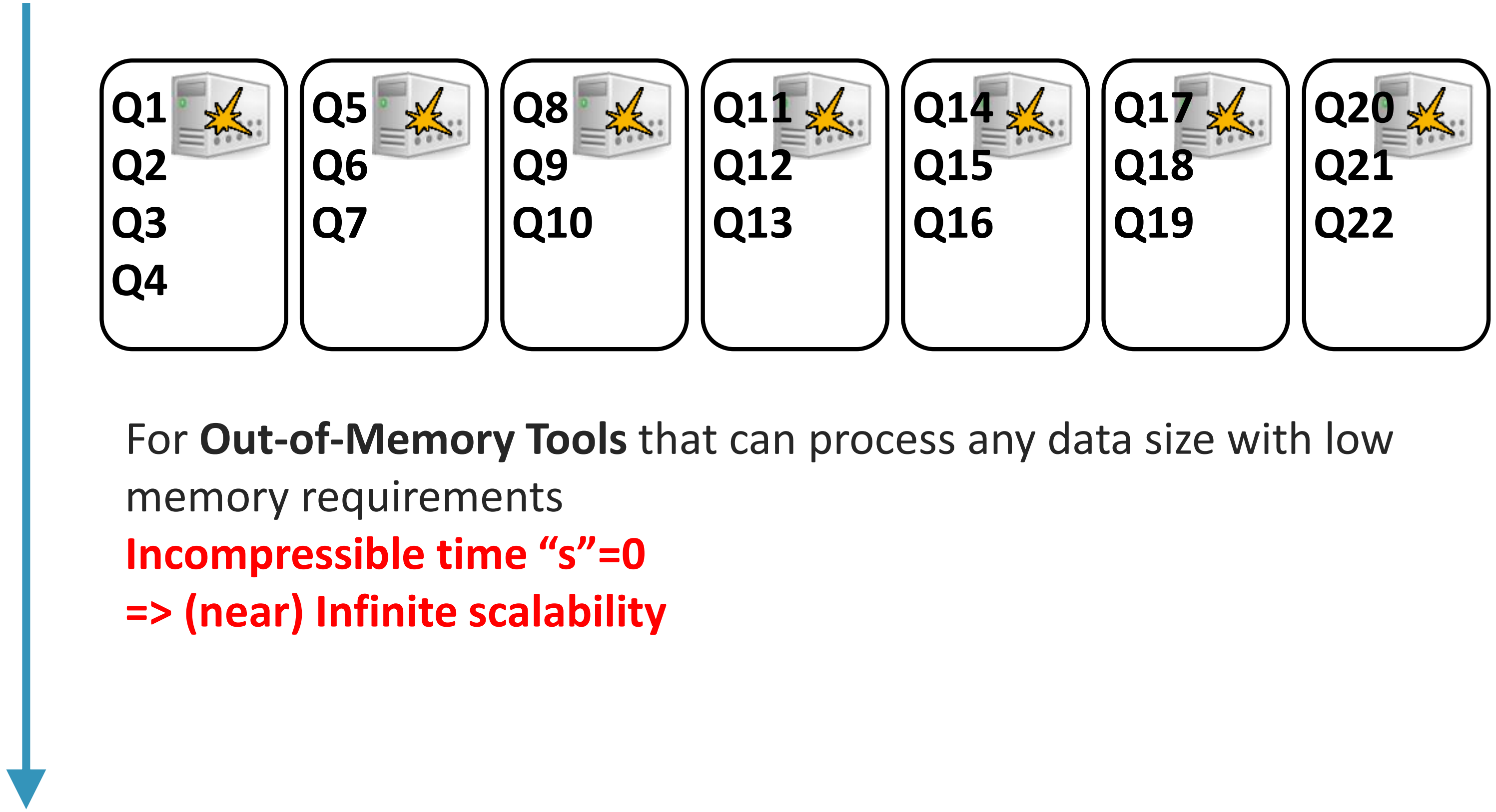
Incompressible time "s"=from 20% to 50%

=> No scalability



(2) One Query per Node

Time



For **Out-of-Memory Tools** that can process any data size with low memory requirements

Incompressible time "s"=0

=> (near) Infinite scalability

	1 TB database	
TPC-H Query	Anatella runtime [sec]	Anatella RAM usage [MByte]
Q1	260	204
Q2	61.5	800
Q3	360	10053
Q4	337	160
Q5	509	2045
Q6	65.3	154
Q7	760	8828
Q8	511	1576
Q9	2668	7392
Q10	394	2169
Q11	32.8	2192
Q12	284	1161
Q13	1109	1186
Q14	37.3	257
Q15	112	2528
Q16	280	11636
Q17	646	525
Q18	408	8672
Q19	492	188
Q20	314	885
Q21	330	329
Q22	595	2027

Distributed computations: “One query per node”: Low RAM requirements

Average of the “RAM” consumption is: 2953 MB

With Anatella, we manipulate a 1TB database using less than 3GB RAM on average!

As a comparison, on a 1GB database, Spark uses between 2 GB and 4GB RAM.

Inside Anatella, we can rewrite Q3,Q7,Q9,Q16,Q18 to use around 2GB (at the price of 30% more seconds at runtime)

timi
integrated data mining

TIMi vs Spark “in the cloud”

TPC-H Query	Anatella 1 CPU runtime [sec]	spark 1CPU runtime [sec]	Anatella Speedup vs Spark 1 CPU
Q1	27.1	184	6.8
Q2	5.7	956	167.2
Q3	34.5	929	26.9
Q4	33.7	830	24.6
Q5	43.7	2275	52
Q6	6.2	102	16.4
Q7	45.6	1113	24.4
Q8	49.3	1621	32.9
Q9	200	2059	10.3
Q10	38.9	1035	26.6
Q11	4.2	441	104.2
Q12	47.7	454	9.5
Q13	105.6	377	3.6
Q14	3.2	373	115.9
Q15	9.7	error	
Q16	31.4	839	26.7
Q17	26.7	1255	46.9
Q18	36.9	1135	30.7
Q19	44.1	972	22
Q20	21.7	972	44.7
Q21	127.7	3815	29.9
Q22	44.5	206	4.6
SUM	988.1	21943	

If we assume “one query per node” distributed computation model (i.e. we use the most efficient distributed computation model):

Average of the “Speed-up” compared to Spark is 39.4

1.000 € with Anatella

/22

22.200 € with “Unreliable” Spark on 1 machine

Anatella is 22.2 times more efficient than Spark

x5

111.000 € with Spark on 10 machines

Infrastructure Cost is multiplied by 10 because of 10 machines
Running-time is divided by two
=> Price is multiplied by 10/2=5

x100

(555.000 € with Spark on 50 machines)

$$\frac{21943}{988.1} = 22.2$$ **: Anatella is, at least, 22.2 times more efficient than Spark**

Summary

- Spark incompressible-time "s" is between 20% to 50%.
  Catastrophic failure: The maximum "speed-up" for Spark is between 2 and 5 (when adding more CPU's).
- One Anatella server is always (several orders of magnitude) faster than a Spark cluster of infinite size.
- With Anatella, there are no limits in computing power: i.e. "Speed-ups" above 1000 are possible.
- With Anatella, there are no limits in volumetry (manipulate a 1TB database using less than 3GB RAM!).
  Anatella is also much more reliable.
- When you switch from Spark to Anatella: Divide you Amazon bills by 100!
- With Anatella, you have the choice to totally avoid the cloud and all the disagreements that comes with it!
  (You get: higher computation speed, lower costs, a more secure infrastructure)
- Data scientist's efficiency multiplied by a factor between 4 to 11 (because of Anatella's speed & integration with TIMi).
- Better results: enough computing power to find the "golden egg"
- No headache: better and easier maintenance
- Anatella has a Free community edition!

timi
integrated data mining

WHAT ABOUT THE OTHER 20 %?

"No free lunch": There will always be a specific, ad-hoc algorithm that solves a problem better than any generic and automated tool.

Competition	Metric	Winner	TIMi (or similar automated tool)	Diferencia
Heritage Health Price	Some kind of R^2	46.12%	46.24%	0.12%
AUSDM2009 (following Netflix)	AUC	69.41%	69.24%	0.17%
Kaggle Axa Telematics 2015	AUC	96.35%	95.97%	0.38%
PAKDD2007	AUC	70.01%	69.28%	0.73%
PAKDD2010*	AUC	64.10%	63.30%	0.80%
KDD2009-upselling	AUC	90.92%	89.94%	0.98%
Datascience.net Axa cross-selling 2015	Lift at 10%	26.09%	24.74%	1.35%
KDD2009-churn	AUC	76.51%	74.74%	1.77%
KDD2009-appetency	AUC	88.19%	86.31%	1.88%

Let's consider

We solved it in 2007.

Thanks for your Attention

For more information, please consult our website:

https://timi.eu

Download your free copy of Anatella today!

Backup up Slides

The following slides are not part of the presentation. They are used occasionnaly to answer to some specific technical questions.

Stop dreaming... Start acting now with TIMi !

	"Old School", Legacy Solutions (SAS, IBM, Statistica, ...)	New Wave: Classical Hadoop (Spark,etc.)	New Wave: TIMi
Main Bottlenecks (complexity)	There are not enough: • specialized statisticians, • computing power	There are not enough: * specialized data scientists	There are not enough: * Marketers ☺ (self-service on laptops)
Self – Service (Citizen Data Scientist)	😐 (only for simple things such as dashboards)	☹	☺ Everything is in self-service, without code: ETL, modeling, dashboards
Architecture	1 or 3 BIG servers Exadata....)	Giant clusters (200-300 servers)	Everything can run on 1 or 2 Laptops
1 Model	3-4 weeks	3-4 weeks	1-3 hours (+ high accuracy)
100 Models (time)	☹	😐 (but tricky)	☺ (1 day + high accuracy)
Data Access (For Telco: e.g. ASN1)	Third party tool	Not in ecosystem (Third party tool)	Integrated & Fast
Warehouse Update (speed)	3-4 / year	1/ month	Daily (or more)
360° Customer View (For TelCo,...)	300	500	2000
Advanced AI functionalities (e.g. network mining, text mining)	On a sample	☺ (but tricky – no graph mining)	More accurate results, No Size Limits & Self-Service
Deployment / Scoring	Strategic Only	2-3 weeks (High Maintenance Cost)	One click
Small Datasets (less than 200 rows)	☺	😐 (using the integrated R engine)	😐 (using the integrated R engine)
Man Hour	$$$$$$$$$$ (a lot, PhD in Math)	$$$$$$$$$$$$$$$$$$$ (too much, many MS in Data Science & IT)	$ (with people like us) (license per PC per year)
ROI	?	?	☺
Community	😐 (required because of bad Hotline)	☺ (required because full of bugs)	😐 (unimportant because of fast Hotline)

People of timi

Frank Vanden Berghen

- Chairman & CEO timi Global
- Specialized in data mining since 1999
- PhD in applied Mathematics
- Extensive consulting experience in many industries including TelCo, FSI, Retail, etc.

Frank founded Timi (Business Insight) in 2007, after completing a PhD in applied mathematics focused on optimization methods and predictive modeling.

As he faced constant challenges in processing big data on client project, he started adding more functionalities and developing the integrated data mining suite that is known today.

Frank steers the company and transmits his values of uncompromised ethics in all we do: high quality code, excellent client focus, and over-achieving in service.

Frank leads the R&D department, is chairman of the board of timi global, CEO, leads academic relations and certification programs.

timi
integrated data mining

Daniel Soto Zeevaert

- Executive Director American Markets
- Specialized in Advanced Analytics since 1999
- Expert in data mining, quantitative market research
- Previous work include Deloitte Consulting, Essec Business School, the Pennsylvania State University,
InSites Consulting, and Direktio.

Daniel leads our operations in the American markets.

He has an extensive experience in analytics and has been a promoter of Timi for the past 5 years.

Daniel combines a strong academic background, a extensive consulting experience and an entrepreneurial profile that make him uniquely suited to lead a team of experts in predictive analytics.

He has worked in many industries and has been a speaker in many professional conferences such as SAS forum, Baqmar, Professional Pricing Society, Deloitte Analytics, and ACEMI.

He also gave conferences and courses in universities in the US, Belgium, France, Peru and Colombia.

timi
integrated data mining