timi vs APACHE Spark

THE ULTIMATE SHOWDOWN

<u>Executive Summary:</u> (1) On all the data-management-benchmarks that we could find, one TIMi server is faster than a Spark cluster, whatever the size of the cluster (on average, the “speed-up” of TIMi versus Spark is 39). (2) The promise of Spark to be horizontally scalable is erroneous: Depending on the workload, you can expect to obtain a “speed-up” between 2 and maximum 5, by adding more nodes to your Spark cluster (even with a 1000 nodes cluster, you won’t get a larger “speed-up”). In opposition, TIMi can commonly achieve speed-up over 10000 by adding more nodes. (3) Because of a more efficient engine, TIMi also reduces your architecture costs by a factor of by 22.2 on average. What do you think about reducing your half-a-million Amazon/Azure bill to only 22K€? (4) Using a more efficient data transformation engine, such as TIMi, means that the efficiency of your data science team is multiplied by more than four (5) These key-findings are in-line with the well-known Amdahl’s law about the efficiency of parallel & distributed architectures. All these results are also consistent with comparable results published inside the scientific literature.

Introduction - What’s the objective ?

We’ll compare the efficiency of two data management tools: TIMi and Spark. To do so, we’ll measure the execution time required to run the 22 data transformations that exactly replicates the 22 SQL queries included in the famous open-source TPC-H benchmark. This workload is interesting because it’s universally recognized as representative of the data transformations that are commonly used in the everyday life of all data scientists.

Why?

"Data scientists" are actually spending more than 80% of their time to do simple data management tasks (e.g. research, unification and data cleaning) (Source: IDG). This fact is also sometimes called the "*80/20 data science dilemma*" (google it!). Therefore, a fast and user-friendly solution for high-velocity data transformation is an important element for the comfort, the efficiency and the productivity of any data scientist. Since TIMi is 39 times faster than Spark, it’s thus possible to reduce these 80% of time devoted to data management to only 80/39. This means that the productivity of your data scientists is multiplied by $\frac{1}{20\%+80\%/39}=\frac{1}{22\%}=4.5$

About the two contestants

Spark is a well-known platform that is used in many commercial product as the underlying computing engine for “big data” transformations. Spark is thought to be the fastest data transformation tool in the Hadoop ecosystem (e.g. Spark is recognized to be several order of magnitude faster than Hive). Spark is a distributed “in-memory” tool that will typically uses all the combined RAM memory of all the servers inside the PC cluster to performs the requested data transformations. For this benchmark, we coded the data transformations in Scala. Scala is the native, fastest language available on Spark. Spark is created using the Java language.

The data management tool included inside TIMi is named Anatella. Anatella is created using the C and assembler language. Anatella is an out-of-memory tool. It means that the data that you can manipulate is not limited to your RAM size.

The results

The Table 1 (on the right) contains a summary of the timing results. The Amdahl’s law states that, in all distributed/parallel systems, there always exists an “incompressible runtime” (column B and C) that severely limits the “speed-up” achievable by a distributed process (when adding more CPU’s). Inside Table 1, we used the Amdahl’s law to extrapolate the Spark runtime on an infinite number of CPU’s (the result of these extrapolations is in column B). An illustration of this extrapolation process is given in Figure 1. The column C contains the same information as column B, but expressed as percent: $\textit{column } C=\frac{\textit{column } B}{\textit{column } A}$.Please note that, nearly all the time, the column C is above 50% (see the cells marked in **red**). This means that, usually, **the maximum speed-up for Spark is 2 (=$^1/_{50\%}$), whatever the size of your cluster** (since it’s not possible to reduce the runtime by more than 50% since the incompressible-runtime is 50%). In comparison, the average speed-up achieved with Anatella compared to Spark is 39 ($39=\textit{average}\left(\frac{\textit{column } A}{\textit{column } D}\right)$). These results are confirmed by many independent researchers in the field: For example: The scientific paper named “*Amdahl’s Law in Big Data Analytics: Alive and Kicking in TPCx-BB (BigBench)*” also displays the same incompressible runtimes (between 20% and 50%). **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 just a catastrophic failure for Spark. In other words: We observed that, for each TPC-H query, the total Anatella runtime is largely below the Spark incompressible time (i.e. we always have column D<B or column E>1). This basically means Anatella already finished all computations since a long time while Spark is still busy trying to complete its initialization/incompressible phase. This means that **one Anatella server is always faster than a Spark cluster whatever the size of the Cluster.**

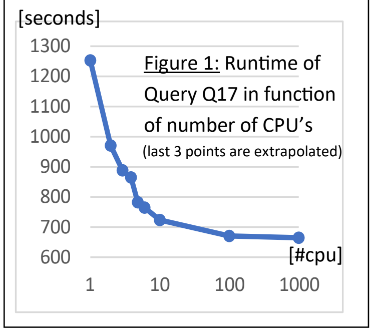
Scalability

The default strategy used by Anatella to distribute the workload amongst a cluster of PC (i.e. “run one query per node”) exhibit an “incompressible runtime” of (near) zero. This means that Anatella is the only solution that is truly infinitively scalable since it can achieve speed-up ratios over 10000 (while Spark is limited to a speed-up of 5 because it cannot run “one query per node” in a reliable way since it’s an “in-memory” tool).

Total cost of ownership

Adopt TIMi now and benefit from infrastructure costs that are divided by 22.2 on average! $\left(22.2=\frac{\textit{sum column } A}{\textit{sum column } D}\right)$ What do you think about reducing your half-a-million Amazon/Azure bill to only 22.5K€? That’s great! 👍👍😄

TPC-H Query (100GB data-base)	Spark (2018/12)			Anatella v1.94	
	running on 1CPU [seconds]	Incompressible runtime (when running on infinite number of CPU's) [seconds]	Incompressible runtime (when running on infinite number of CPU's) [percent]	Run time [seconds]	Speed-up vs Spark infinite CPU
	A	B	C $=^B/_A$	D	E $=^B/_D$
Q1	184	37	20 %	27.1	1.4
Q2	956	649	68 %	5.7	113.5
Q3	929	572	62 %	34.5	16.6
Q4	830	230	28 %	33.7	6.8
Q5	2275	674	30 %	43.7	15.4
Q6	102	21	20 %	6.2	3.3
Q7	1113	762	68 %	45.6	16.7
Q8	1621	1009	62 %	49.3	20.5
Q9	2059	1227	60 %	200	6.1
Q10	1035	699	68 %	38.9	17.9
Q11	441	304	69 %	4.2	71.8
Q12	454	263	58 %	47.7	5.5
Q13	377	143	38 %	105	1.4
Q14	373	275	74 %	3.2	85.5
Q15	error	error	error	9.7	
Q16	839	587	70 %	31.4	18.7
Q17	1255	665	53 %	26.7	24.9
Q18	1135	717	63 %	36.9	19.4
Q19	972	119	12 %	44.1	2.7
Q20	972	643	66 %	21.7	29.6
Q21	3815	2235	59 %	127	17.5
Q22	206	112	55 %	44.5	2.5



<u>Figure 1:</u> Runtime of Query Q17 in function of number of CPU’s (last 3 points are extrapolated)