Notes on data management in Digital Technologies

prof. Antonio Maratea

July 2022

Introduction

In any organization, the information requirements can no longer be satisfied through a collection of a few, well organized, structured data, compliant to a scheme and kept safe in a relational system. On the contrary, the proliferation of acquisition sources, the low cost of memories and the enormous computing power of current computers have made common to store, for analytic purposes, both structured and unstructured data, coming from heterogeneous sources and with a significant volume. The changed needs have led to the adoption of hybrid solutions capable of taming the great variety, complexity and dimensionality of current data, and generated the *Data Scientist* (DS) — a professional figure on whose shoulders the burden of ensuring a difficult coexistence between conflicting needs lies. The DS should be able to handle completely different storage systems, heterogeneous data and increasingly high performance demands.

First a quick reference to basic notions on data encoding and file format, then a detailed list of the parameters that characterize the system requirements to be set up is presented, finally the mapping problem is formalized.

Data encoding

Any representation of information through data passes through an encoding, that, once chosen the set of symbols — the alphabet — and the appropriate translation rules, transforms the information into a sequence of such symbols.

For example in Morse code for telegraphy, where the only symbols allowed are the dot "·" and the line "–" the latin letter F is coded in "$\cdot\cdot-\cdot$".

In the field of Computer Science, for purely technological reasons, the use of two-state devices has historically prevailed over multi-state devices and therefore the low-level coding of any information is ultimately binary, i.e. it uses a two-symbol alphabet (conventionally 0 and 1). Note that it is possible to superimpose one encoding on another, using higher levels of abstraction, as well as constructing words starting from letters and phrases starting from combinations of words. Therefore the binary data "1111" can become the letter F in hexadecimal encoding and can represent a letter of a Western alphabet, a number or a color depending on the context and the way it is read. The most abstract encodings hide the underlying binary code and appear as sequences of symbols other than 0 and 1, although ultimately attributable to these.

File extension

The extension of a file or the header of the file reveals in what format the data are stored in that file, that is, what is the correct way to read its bytes. It is as if each format would represent a specific encoding for all the files of that same type, which establishes how the bytes contained in them are to be interpreted and how those files can be opened correctly, similarly to a song that can be recorded on LP, on CD or left in "liquid "form for a multimedia player and which, depending on the chosen format, needs the relative device to be listened to.

Although all files are ultimately binary, we conventionally distinguish *plain text files* from *binary files*, and *data files* from *executable files*, with the following meanings:

- *plain text files* are files whose bytes correspond 1: 1 to a natural language text character encoding (including special characters). This feature does not optimize storage space, but makes the file very easy to read and edit, being directly interpretable by a human being.
- *binary files* on the other hand are files whose bytes need a specific application to be opened and are not directly intelligible for a human being.
- *Data files* are files in which there are no bytes that correspond to instructions to be executed, but only bytes that correspond to encoded information in the form of data to be read, written or modified.
- *Executable files* are compiled programs that contain, in addition to data, bytes corresponding to instructions for the processor.

From now on, data will be referred to as a data file, in text or binary format as appropriate.

Data nature

It is clear from the premises that in computer systems data are always binary and always require decoding to be interpreted, in this sense all data in any format (except randomly generated bit files) have a very precise structure. However, the following classification is commonly accepted:

- **Structured data**, that is in the form *attribute* = *value* which conforms to a *schema* and which can be stored in the form of rows and columns of a relationship (this is the standard data type in classical relational systems). They can be enclosed in plain text files. The scheme allows numerous consistency checks on the data entered.
- **Semi-structured data**, that is, data that is still enclosed in text files, but for which conformity to a schema does not exist or is partial (eg XML, JSON, CSV formats etc.).
- **Unstructured data**, that is data that are enclosed in binary files, not intelligible to a human being, where the schema is completely absent (they are called *schemaless*) and to decode which a specific application is needed (e.g. images, sounds, presentations, PDF, PPT, DOC, XLS etc files). Plain text files are also conventionally considered unstructured if their content is free text without any labels, markings or separators indicating a structure.

Data sources

The data can be obtained from the compilation of forms, i.e. direct writing, by human beings and are called *human generated*, or they can be generated by a machine and are called *machine generated*, in which case they can be measurements of any physical phenomenon made through a sensor or generated directly by a software process independently from a physical phenomenon.

The generation frequency is relevant for all and regardless of this we speak of data in *Streaming* when the data is generated continuously, possibly from multiple sources, and of data in *Batch* when the generation is discontinuous or irregular. The frequency of data generation is essential to estimate the volume of data collected in a given time interval according to the following formula:

$$\overline{f} \cdot \Delta t \tag{1}$$

where Δt is the time interval and $\overline{f}$ is the average volume of data generated (in bytes or multiples) per unit of time.

In a typical case human generated text files and machine generated binary files must be managed and coexist, all with various categories and different acquisition frequencies.

Definition of expected requirements

Having established the nature of the data to be stored, identified the sources from which they come and estimated their generation frequency, the next step is to establish the requirements that the system must have both to ensure the correct ingestion of the data and to guarantee its efficient recovery and the necessary periodic backup.

0.0.1 Frequency and type of access

Not all data have the same priority with respect to access, especially as they age, and a Pareto distribution of requests is very common, where the 80/20 rule of thumb applies (80 % of requests concern 20 % of the data). In this sense we speak of **temperature** of the data, expressed in the following categories:

- **Hot data**: the most frequently requested data. Usually less than 1/5 of the total, usually the most recent, where most of the optimization efforts are concentrated;
- **Warm data**: data with infrequent requests. Usually uninformative data, not very recent or periodic backups for short-term restores;

- **Cold data**: data with very rare requests, such as full backups, historical data, auditing data, summary reports;
- **Frozen data**: data that is kept indefinitely for mainly legal reasons, where the recovery time can also be very long.

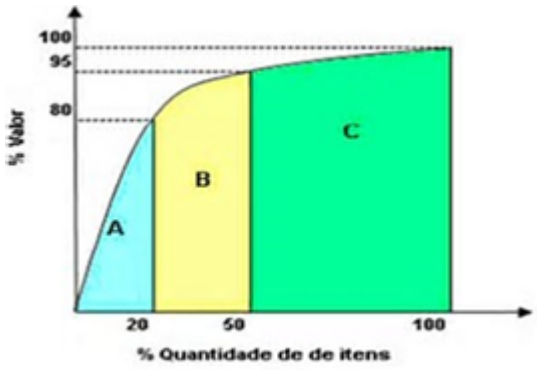

Figure 1: Percentage of requests as a function of the percentage of data involved.

It is then possible to operate on each of these categories of data:

- Reads only.
- Mostly reads.
- Readings and writings.
- Mostly writings.
- Writings only.

The prevailing type of operation determines the type of primary organization and preferable technology. For example, if readings are predominantly made on the most recent data and these are unstructured, a primary organization sorted by timestamp in a key-value system is likely the solution to be adopted.

Sometimes a temporary memory area called *Staging Area* is required to ensure data ingestion, which in more complex cases can become a real *Data Lake*. Also for this area it is necessary to estimate the ideal size and add its cost to the total management costs.

The data life cycle

Each data has its own time horizon, i.e. a time window within which it is useful, usually with a decreasing utility trend over time similar to a function χ^2. While maintaining a similar trend, in the same context the time window of usefulness of the data can vary from a few hours (think of table reservations at a restaurant) to tens of years (think of personal registry for documents), with various constraints that they are legal rather than technical.

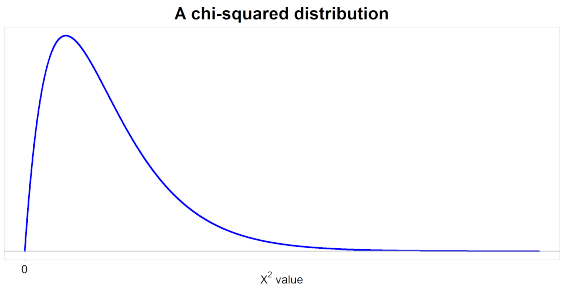

Figure 2: trend of the usefulness of data over time. $t = 0$ is the acquisition instant.

From the point of view of storage, data cools as they age, that is, its priority decreases and both readings and changes on them become less frequent. Since low-priority storage areas have higher order-of-magnitude access times and lower order-of-magnitude costs, it is up to the Data Scientist to decide wisely when to downgrade the data that are less useful at that time.

Hardware or resources available or required

From a traditional view of hosting data *in house*, we can imagine three levels of mass storage available:

- fast and expensive semi-random access, online, medium capacity memories (eg HD NVMe, SSD, intel optane);
- memories of medium speed and cost with semi-random access, online, of good capacity (eg rotational HDs);

- slow and inexpensive, sequential access, large capacity, offline memories (eg tapes).

For each of these memories it is necessary to estimate the total capacity, the annual cost per byte, the latency, the throughput, the availability and the durability and it is up to the Data Scientist to decide how best to allocate the data and which technologies to use at each level of the hierarchy. He must also establish a periodic **backup policy**, which can be *incremental*, *full* or mixed, trying to avoid or reduce system downtime and ensuring that all important data is not lost in the event of a failure.

In-house hosting involves very high management and updating costs and, unless there are particular needs for privacy and confidentiality of the data, it is no longer economically convenient. Cloud hosting, on the other hand, ensures a wide range of options in terms of performance and prices, with the important advantage of a *pay-as-you-use* pricing option and simple scalability if needed, with zero hardware upgrade costs.

Characteristics of data storage systems

availability (or uptime) quantifies the percentage of time that data is accessible in a year, that is the probability that a request will be satisfied in a year. A value of 99% of availability means that in a year there are 3.65 days in total — the 1% — of data unavailability (downtime) — this value may or may not be acceptable, since it is not all applications must be operational 24/7. On the other hand, going up to 99.99% the unavailability is reduced to about 53 minutes per year, but the costs of the service also rise considerably. Higher availability figures are difficult to achieve due to hardware maintenance operations in data centers. Also in this case it is up to the Data Scientist to find an acceptable compromise between the cost of adding significant figures to the availability and the losses resulting from the temporary unavailability of the data.

Note that since the availability value does not take into account network outages, problems with the ISP, faults or power outages downstream of the purchased service and since the actual availability depends on a logical AND between the operation of the local and global network, of the client, of the ISP, by the absence of hardware failures or extraordinary maintenance operations, the actual availability will be the product of the various availabilities and will be strongly conditioned by the minimum value p_{min}.

$$a = p_1 p_2 p_3 \dots p_n \tag{2}$$

For example, if the ISP guaranteed a 98% network availability, and this was the minimum value, it means a downtime of one week in a year and at that point buying 99.99 availability for the data hosting service would be completely ineffective.

The **durability** or persistence is in turn expressed with a value that quantifies the probability that a memorized object will not be lost in one year. It seems strange to see persistence expressed as a percentage because we are used to thinking that once stored in the *bare metal* the data are safe, but in reality the hard disks have a probability of failure that is never zero and there is the possibility that the data will be corrupted or lost.

In the case of data hosting services, persistence is guaranteed with 11 significant figures - 99.999999999%. Such a value is possible thanks to multiple copies of data and redundant error-correcting encodings and what it means in practice is that even on Petabytes of data stored for prolonged periods of time the probability of losing a file is extremely low, much lower than storing it on a hard drive or tape and even lower than an asteroid hits the earth. Note that in this case a value of 99.99% would mean having one file lost in every ten thousand, which in a large number of real-world applications is unacceptable.

The **latency** is the average time it takes for a request to be answered. It is a parameter whose maximum value is largely determined by a human factor, representing the maximum time that a user of the system is reasonably willing to wait before seeing his request satisfied. It is very important to generate a good User eXperience (UX) that the latency is low and as constant as possible, regardless of the volume of data processed, and this is achieved through careful physical design and fine tuning of the system.

The **throughput** is the volume of data that the system is able to ingest and process in the unit of time, it must be adequate in relation to the volume of data generated. If the volume of data generated is greater than the throughput, the system is undersized; in the event that this happens only in particular moments (think of traffic peaks due to holidays or exceptional circumstances) it is necessary to have a staging area with an adequate throughput to then postpone the ingestion at a later time.

reliability is the ability of the system to recover to a consistent state following a non-catastrophic malfunction or failure. It is a fundamental requirement of transactional systems, although there are cases where it is not strictly necessary. It is often coupled with a failover mechanism that guarantees system operation even during recovery (such as fuzzy checkpointing) and helps to define what is called **resilience**, i.e. the ability of the system to continue operating even at following a hardware or software failure.

The **access control** can be mandatory and discretionary. While on SQL-based systems this control is native, on alternative systems it is often not available and must be implemented in another way.

The **consistency** is the guarantee of data integrity and, in a distributed environment, the guarantee of consistency of the various copies of the data. It is necessary to evaluate on a case-by-case basis whether strong consistency is necessary (ACID properties, SQL systems) or whether the eventual consistency (BASE properties, NOSQL systems) is sufficient, or a certain delay in the synchronization of the copies is tolerable.

The **cryptography** is the protection of data against unwanted reading, and it is necessary to establish at each level of the hierarchy of memories and in each data transfer in output or in the system if it is necessary and in what form. It requires a significant overhead so should be used only if strictly necessary.

The mapping problem

The final task of the Data Scientist is to establish which data goes on which system and with which technology, at what level of the memory hierarchy and at what annual cost.

This process is a surjective mapping between data and storage resources, made in such a way as to respect the requirements imposed on the data itself in terms of availability, latency, temperature etc: on the one hand there are the data to be managed classified according to their nature and their temperature (priority) and on the other hand the available or necessary memory resources, henceforth called *media*. Each line indicates which data ends up on which storage / support resource it should be noted with the technology chosen to manage it (key-value, graph, relational etc.). Coexistence of multiple systems is the norm.

The Data Scientist also establishes the data partitioning strategy, redundancy level, the load level of each system (how much free space to leave), the expected performance (latency and throughput), the type of technology to use and any other detail of the physical design necessary to ensure compliance with the previously established requirements.

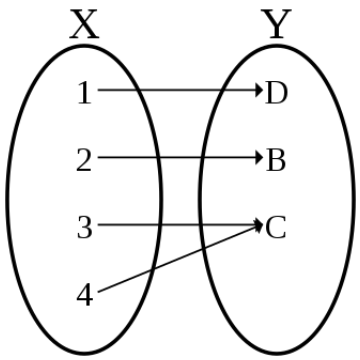

Figure 3: mapping between data (left) and storage resources (right).

To this end, it is essential to evaluate the following quantities.

0.0.2 Estimation of data volume

Considering the various types of data (structured, semi-structured and unstructured) and the various

sources, each of them generally produces data of one or more different types at different frequencies.

Assuming without losing generality four sources called S_1, S_2, S_3, S_4, each generally has three data generation frequencies: $f_{S_.}^{st}$ for structured data; $f_{S_.}^{ss}$ for semi-structured data; $f_{S_.}^{ns}$ for unstructured data. The total volume of data generated by type in a time interval Δt is given by the following formulas:

$$V^{st} = \Delta t \sum_{i}^{4} f_{S_i}^{st} \tag{3}$$

$$V^{ss} = \Delta t \sum_{i}^{4} f_{S_i}^{ss} \tag{4}$$

$$V^{ns} = \Delta t \sum_{i}^{4} f_{S_i}^{ns} \tag{5}$$

$$V_{tot} = V^{st} + V^{ss} + V^{ns} \tag{6}$$

that if each source produces only one type of data [1] and S_3 and S_4 are sources of the same type, simplifies as follows:

$$V^{st} = \Delta t f_{S_1}^{st} \tag{7}$$

$$V^{ss} = \Delta t f_{S_2}^{ss} \tag{8}$$

$$V^{ns} = \Delta t (f_{S_3}^{ns} + f_{S_4}^{ns}) \tag{9}$$

$$V_{tot} = V^{st} + V^{ss} + V^{ns} \tag{10}$$

It is important to keep the separate volume estimate according to the nature of the data, not only V_{tot}, given the need to manage data of different nature with different technologies.

Workload estimation

Workload estimation serves to size not only the memory, but also the computing power needed to manage the requests of the various users on the data. When dealing with data-intensive applications, it is crucial to assess the I/O load to which each storage resource is subjected. A workload consists of a set of transactions launched in a defined time interval and can include a certain volume of writing data and a certain volume of reading data, on which the size of a possible cache memory and the effectiveness of the chosen caching policy, both condensed in the *hit ratio* H_R parameter of the cache, have a major effect.

Assuming a single cache and three storage resources, B, C and D, are available, sorted by decreasing temperature $T_B > T_C > T_D$ and called *media*, a set of k transactions $Tr_1, Tr_2, \ ldots, Tr_k$ operating on the system can write and read simultaneously on multiple media. The aim is to estimate how many writes and how many reads occur per unit of time on each media. Called W_B the number of write operations on B overall present in k transactions and R_B the number of read operations on B overall present in k transactions in the same time interval $Deltat$, the total number of I / O operations on each media is:

$$O_B = (1 - H_R)(W_B + R_B) \tag{11}$$

$$O_C = (1 - H_R)(W_C + R_C) \tag{12}$$

$$O_D = (1 - H_R)(W_D + R_D) \tag{13}$$

$$O_{tot} = (1 - H_R) \tag{14}$$

Knowing the average time for each I/O on each media, it is possible to calculate the Δs time required to run the k transaction workload and compare it to Δt. If the ratio $r = \Delta s / \Delta t$ is close to 1, the system is not dimensioned adequately, while a good balance is obtained with a value $r < 0.2$.

The workload can be very variable over time, in which case a worst-case analysis is required rather than a mean-case analysis and normally a Poisson distribution is used (figure 4). However, sizing everything on the worst case may not be a good idea in economic terms if this is quite rare and if it does not lead to a total blockage of the system but only a slowdown. The estimate can also be made at different time granularity (hourly, daily, weekly etc.) where there are credible data on the number of transactions submitted to the system instant by instant in operating conditions.

[1] It is always possible to put oneself in this condition by considering as coming from independent sources the data of a different nature arriving from the same source

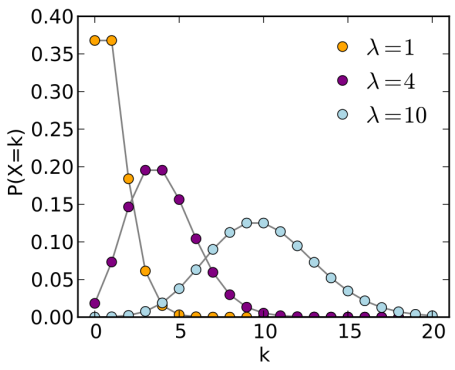

Figure 4: Distribution of the number of successive I/O in a given time interval.

Estimation of storage costs

Assuming three storage resources are available, B, C and D, sorted by decreasing temperature $T_B > T_C > T_D$ (therefore by decreasing price $c_B() > c_C() > c_D()$, where $c()$ is the cost function per byte / year) and assuming four data sources S_1, S_2, S_3, S_4, each producing a different volume of data per unit of time, respectively V_1, V_2, V_3, V_4, the mapping function indicates which data goes to which medium. Assuming without losing generality the absence of fragmentation between the data of the same source (but the fragmentation could very well exist) and a redundancy factor f_1, f_2, f_3, f_4, with $1 \leq f. \; leq 1.5$, different among the various sources, the cost of storage for one year of data with the mapping function in figure 3 is given by the following formula:

$$C_T = c_D(V_1 f_1) + c_B(V_2 f_2) + c_C(V_3 f_3 + V_4 f_4) \quad (15)$$

that in case of a solution with a constant value of temperature ($c_B() = c_C() = c_D() = c()$) and constant redundancy ($f_1 = f_2 = f_3 = f_4 = f$) is simplified as follows:

$$C_{TOT} = c(f \sum_{i}^{4} V_i). \quad (16)$$

The cost functions $c()$ can be time-dependent, in which case the annual cost is the integral over the year of the daily (or hourly) cost:

$$C_{TOT} = \int_{t_1}^{t_2} c(t)\, dt. \quad (17)$$

Conclusion

"I always thought something was fundamentally wrong with the Databases"