Foundations in Data Engineering

Ilaria Battiston *

Winter Semester 2019-2020

*All notes are collected with the aid of material provided by T. Neumann. All images have been retrieved by slides present on the TUM Course Webpage.

Contents

1 Tools and techniques for large-scale data processing

Data processing, especially in the age of big data, can be done with many different approaches and techniques: it is handled with software layers, which have different principles and use cases. All digital data created reached 4 zettabytes in 2013, needing 3 billion drives for storage.

Scientific paradigms concerning collection and analysis have developed as well, including a preprocessing part consisting in observations, modeling and simulations.

Big data is a relative term: each field has its own definition, but it's usually hard to understand. A model might be as complicated as its information, and tools need to be reinvented in a different context as the world evolves.

1.1 Volume and privacy

Volume is one of the most important challenges when dealing with big data: it is not often possible to store it on a single machine, therefore it needs to be clustered and scaled.

Supercomputers (and AWS) are extremely expensive, and have a short life span, making them an unfeasible option. In fact, their usage is oriented towards scientific computing and assumes high quality hardware and maintenance, taking months to develop a program.

Cluster computing, on the other hand, uses a network of many computers to create a tool oriented towards cheap servers and business applications, although more unreliable. Since it is used to solve even large tasks, it relies on parallel database systems, NoSQL and MapReduce to speed up operations.

Cloud computing is another different instance which uses machines operated by a third party in a data center, renting them by the hour. This often raises controversies about the costs, since machines rarely operate at more than 30% capacity.

Resources in cloud computing can be scaled as demand dictates, providing elastic provisioning. The problem with this are the proprietary APIs wit lock-in that makes customers vulnerable to price increases, and local laws might prohibit externalizing data processing.

Providers might control data in unexpected ways, and have to grant access to the government - privilege which might be overused.

Privacy needs to be guaranteed, stated and kept-to: numerous web accounts get regularly hacked, and there are uncountable examples of private data being leaked.

Performance is strictly linked to low latency, which works in function of memory, CPU, disk and network. Google, for instance, rewards pages that load quickly.

1.2 Velocity

Velocity is the problematic speed of data: it is generated by an endless stream of events, with no time for heavy indexing leading to developments of new data stream technologies.

Storage is relatively cheap, but accessing and visualizing is next to impossible. The cause of this are repeated observations, such as locations of mobile phones, motivating stream algorithms.

1.3 Variety

Variety represent big data being inconclusive and incomplete: to fix this problem, simple queries are ineffective and require a machine learning approach (data mining, data cleaning, text analysis). This generates technical complications, while complicating cluster data processing due to the difficulty to partition equally.

Big data typically obeys a power law: modeling the head is easy, but may not be representative of the full population. Most items take a small amount of time to process, but a few require a lot of time; understanding the nature of data is the key.

Distributed computation is a natural way to tackle Big Data. MapReduce operates over a cluster of machines encouraging sequential, disk-based localized processing of data. The power laws applies, causing uneven allocation of data to nodes (the head would go on one or two workers, making them extremely slowly) and turning parallel algorithms to sequential.

2 File formats

Files can have a huge variety of formats, when in doubt the `file` command can be useful.

CSV are plain text files containing rows of data separated by a comma, in a simple format with customizable separator. It requires quoting of separators within strings.

XML is a text format encoding of semi structured data, better standardized than CSV but not human writable. It is suitable for nested objects and allows advanced features, yet it is very verbose and its use is declining.

JSON is similar to XML although much simpler and less verbose, easy to write. Its popularity is growing.

2.1 Command line tools

Text formats are overall well-known and can be manipulated with command line tools, a very powerful instrument to perform preliminary analysis:

- `cat` shows the content of one or multiple files (works with piped input as well);
- `zcat` is `cat` for compressed files;
- `less` allows paging (chopping long lines);
- `grep` is useful to search text (regex goes between quotes) with options such as file formats, lines not matching and case insensitiveness;
- `sort` to (merge) sort even large output;
- `uniq` handles duplicates;
- `tail` and `less` display suffixes and prefixes;
- `sed` to edit text, match and replace characters (in-place update using the same file);
- `join` to combine sorted files according to a common field;

- `awk` (followed by a `BEGIN-END` block) executes a program for every line of input, such as average or sum;
- ...

2.2 Performance spectrum

Analyzing the performance is relevant to understand whether an operation is taking too long, and comparing it according to time and input size.

The performance spectrum involves many variables: theoretical limits, programming tools and methods, or hardware (clustering, scaling). Real input is complex, yet just analyzing simple text queries can be highly explicative.

Performance when manipulating text completely ignores CPU costs, which are not important for disks but very relevant for reading data from DRAM: this option is one of the fastest, followed by SSD.

A good way to optimize code is memory mapping: files are saved in the address space of the program, and are treated as arrays being accessed in the same way as dynamic memory.