HPC and Big Data
Reading Assignment

Florian Wolf - 12393339 (UvA)
flocwolf@gmail.com

Lotte Felius - 12368032 (UvA)
lotte.felius@student.uva.nl

Veerle Dinghs - 11426020 (UvA)
veerledinghs@live.nl

February 2, 2021

1 Introduction

In this report we reflect on two publications from a decade ago, which focus on the, back then novel topic Big Data. Further, these publications present the requirements of Big Data to High Performance Computing (HPC) systems and predict their evolution. We reflect on the content of the publications and set it in context with the knowledge gained during the course HPC and Big Data at the University of Amsterdam (UvA). Finally we compare the predicted technological developments to the current state-of-the-art (SOTA) in cluster and cloud computing.

2 Cluster & Cloud Computing

The publication *Cloud Computing and Grid Computing 360-Degree Compared* by Foster *et al.* [1] presents and compares, as implied by the title, cloud and grid computing. Thus, lets start by defining what these two terms are and further reflect on them from the point of view of architecture, resource management, programming, application and security.

What are grids and clouds?

The two terms cannot be fully separated. Both have the underlying function of providing access to large computational resources, of a scale which no desktop computer could ever reach. Grid or Grid Computing can be seen as the base infrastructure of for supercomputers. While Foster *et al.* still differentiates between supercomputers and clouds, we can now a days say that cloud services are also based on supercomputers. Supercomputer in this context describes a supercomputing facility, such as SURF [2], which joins the computation power of several nodes, including storage resources, with each node being an independent computer. Foster *et al.* makes a second differentiation between application and service-oriented systems. while grids are the backbone infrastructure for both, they differ in use case. Clouds are service-oriented interfaces, with a business model of providing access to computation power over the web to a client, potentially anywhere in the world. Supercomputers also rely on the grid infrastructure, but give access to a limited group of users, for example researchers of a university. This allows to adjust the system to the specific needs of its users and hence improve performance. So what exactly is a grid now? In its simplest definition, it is a standardized protocol, which builds the infrastructure to connect different, geographically distributed clusters and unites and provides access to their resources. The name grid originated from the grid-layout, in which all clusters are connected with each other.

Cloud and grid have developed in an different direction. Clouds have adopted to provide versatile services, using Virtual Machines (VM), dockers and kubernetes [3] [4]. Instead of interconnected grids, their infrastructure have been been vastly replaced by on-demand service and resource managers, E.g. Representational State Transfer (REST)-ful web-services. Clouds offering such web-services have the great advantage of being platform-independent and language independent, thus attracting a much larger variety of customers.

In terms of resource management, cloud and grid differ in availability. Clouds provide an on-demand service, meaning the user expect the resources to be available right away to fulfil the customers need. This is only possible by sharing the resources between all users. On grids, batch jobs are submitted, specifying the needed processing power and time. A resource manager allocates resources for the job and executes it when these resources become available. Reflecting on SURF, the computation facility accessed during this course, this way of resource management still applies. An exception are the SOTA Graphic Processing Units (GPU) resources, which due to being rare have the option to be shared [5] [2].

Foster *et al.*s prediction, that with increasing data size, data management would become crucial and data transfer a bottleneck, has proven true. While he also predicts that desktop computers will evolve to supercomputers, this did not yet come true, as modern Big Data has reached sizes which do not fit a desktop computer. Figure 1 shows his visualization of the relation between data, cloud and user as triangle. This diagram still applies today, as Big Data is not stored on private computers but is accessed over the internet. The diagram would require an adjustment in the sense that now clouds are also used as data storage. The management of Big Data and its allocation between the computing nodes is still the limiting bottleneck.

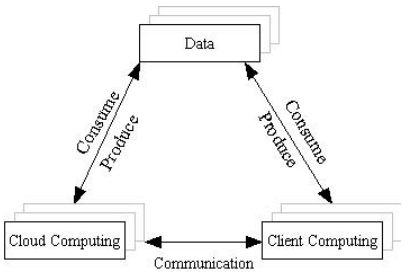

Figure 1: Prediction from 2008 for future data management. Source [1].

From the point of view of programming, Foster *et al.* points out the importance of parallel computing. Message Passing Interface (MPI) is named as the most common used programming model. We have seen in the scope of this course, that this still holds and the maximum performance of a cluster can only be achieved by implementing parallel computing.

Comparing applications, Foster *et al.* differentiates between HPC and High Throughput Computing (HTC). While HPC relies on MPI, HTC has loosely coupled data. In context of this course an example of HTC is the scientific visualization of data, using SURF as backend to render each datapoint. This is a grip specific application and to the best of our knowledge, cloud services do not yet offer the option of being the backend of a desktop program.

Foster *et al.* points out the complex security of grids and predicts that security in clouds will be a mayor challenge. The security me assures described are similar to those encountered on SURF, restricting rights and access based on the user. Clouds on the other hand have found an elegant solution to security by encapsuling the users OS in e.g. VMs or Docker images, which simulate an OS where the client is administrator, yet does not get access to the underlying managing system (Kubernetes, Haddop).

Finally Foster *et al.* compares the future development of grids to an electric power grid, which will have a variety of different energy sources all feeding the same network. For grids this assumption has not yet come true. Steps have been made e.g. between universities or even governments to unify their grids, yet we are not at the point where any grid can be connected to the same network. On a higher level, container based architectures have evolved which allow to reproduce and compute the exact same task on any kind of grid.

Clouds have matured and provide a vast variety of web-services. APIs have become indispensable with smart-phones becoming the center of our modern lives. At this point we dare to make a prediction of our own, namely that with rise of gadgets and ever improving internet capability, clouds will in near future completely replace desktop computations. Clouds will still use grids and grids will come to use clouds.

3 Pathologies of Big Data

The second publication included in this report is *The pathologies of big data* by Jacobs [6]. More than giving specific answers, Jacob asks the right questions and provides the reader with food for thought. We pick up these questions and reflect on them based on the knowledge gained in the scope of this course. While the publication is written as a narrative, its arguments can be sectioned into three main clusters. Jacobs analyzes the limitation imposed by Big Data from the point of view of hardware, data processing and data representation. Following the structure of the previous section, we upfront ask the question:

What is Big Data?

Jacobs defines it as *data whose size forces us to look beyond the tried-and-true methods that are prevalent at that time.* This is both a unspecific and universal definition, It does not define a range in terms of Gigabyte (GB) or datapoints but gives a definition which is still perfectly applicable 11 years later. E.g. the rise of Artificial Intelligence (AI) and specifically Deep Learning, which relies on huge amounts of data, continuously forces us to come up with new solutions. Further we can define Big Data as data we know little about and thus, holds new challenges. Once data is well explored and its challenges solved, it is empirically not seen as Big Data anymore.

Jacobs starts with relating hardware requirements with hardware development and availability. He makes two points which still apply, the hardware capacity will keep rising, and so will the requirements, thus the limitations will stay. Further he points out the scaling out, by connecting multiple hardware with less capacity, is cheaper and can reach has a higher upper limit than using a single SOTA machine. We have seen in this course that distributed systems and parallel computing are indispensable. Jacobs compares the computation performance between RAM and disk-storage and concludes that the real bottleneck is the data transfer speed. While modern solutions have adopted parallelism and speed up data transfer using parallel streams, it is still is the case that the transfer costs unproportionally more time than the computations done on the data.

'It's easier to get the data in than out', this points out that storing Big Data is less of a challenge than computing results. He demonstrates how popular office data solutions such as Microsoft Excel fail at managing Big Data. These programs are optimized to the common users needs, and do not scale due to data inflation and sub-optimal query algorithms. A milestone tackling this problem are the series of Apache solutions. Apache spark is an analytics engine for large-scale datasets supporting data-parallelism on clusters and with interfaces for various programming languages. Apache Storm offers a solution for real time data analysis and Apache Kafka a central message passing system, allowing high through-put and low latency [7] [8] [9]. These solutions evolved form the challenges imposed by Big Data, and allowed data to become even bigger.

Technical Limitations go further than only the software limits. Jacobs asks the question: *What makes data really "big"?*. He points out both, the size differences between storing the same data in binary or in e.g. SQL format, and the huge cost of random access to data storage. While the preference for random access still persist, alternative solutions have been proposed for the creation and representation of data. E.g. MapReduce is a solution for processing and generating big datasets [10]. In terms of format, Not Only SQL (NoSQL) databases have evolved, presenting scalable and elastic solutions for data formats [11]. An example for NoSQL formats is graph data, a format where data is stored as connected entities. This is widely used in the semantic web.

Finally, we reflect on the advice Jacobs gives. The idea of scaling out not up has been encountered numerous times during this course, GPUs are preferred over CPUs for AI applications and parallel computing improves the performance of calculation, visualization and data analysis. Solutions which enable parallelism are centralized message passing interfaces and high through-put, and parallel streamed data processing and transfer. Jacobs best-practice rule: *"allow none of foo, one of foo, or any number of foo."* reflects e.g. in multi-threading and data batching, practices which do not impose arbitrary limits and instead are only limited by the underlying hardware components.

Wrapping up this this chapter, we would like to point out open challenges of Big Data. Data diversity still imposes a unsolved challenge in terms of compatibility and provenance. Data integrity and security is not only a challenge but a major concern of our generation. Protecting privacy and ensuring transparent and unbiased data models will hopefully be on top of the agenda for the

evolution of Big Data,

References

[1] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *2008 grid computing environments workshop*, pages 1–10. Ieee, 2008.

[2] Song Feng, ETH Torsten Hoefler, Switzerland Jessy Li, Damian Podareanu, Qifan Pu, Judy Qiu, Vikram Saletore, Mikhail E Smorkalov, and Jordi Torres. Valeriu codreanu, surfsara, netherlands ian foster, uchicago & anl, usa zhao zhang, tacc, usa.

[3] Kelsey Hightower, Brendan Burns, and Joe Beda. *Kubernetes: Up and Running Dive into the Future of Infrastructure*. O'Reilly Media, Inc., 1st edition, 2017.

[4] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. 2014(239), 2014.

[5] Anthony Ralston, Edwin D. Reilly, and David Hemmendinger. *Encyclopedia of Computer Science*. John Wiley and Sons Ltd., GBR, 2003.

[6] Adam Jacobs. The pathologies of big data. *Communications of the ACM*, 52(8):36–44, 2009.

[7] Nishant Garg. *Apache Kafka*. Packt Publishing, 2013.

[8] Jan Sipke van der Veen, Bram van der Waaij, Elena Lazovik, Wilco Wijbrandi, and Robert J. Meijer. Dynamically scaling apache storm for the analysis of streaming data. USA, 2015. IEEE Computer Society.

[9] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[10] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. 51(1), 2008.

[11] Pramod J. Sadalage and Martin Fowler. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley Professional, 1st edition, 2012.