Successful Systems in Production

A MAP-I curricular unit proposal on Technology Option 2018-19

Ali Shoker and Rolando Martins
University of Minho and University of Porto

June 2018

Abstract

We propose "Successful Systems in Production", a MAP-I curricular unit (CU) on the Technology Option. This CU aims at enriching the knowledge base of the students through addressing well-known systems that are considered successful, touching upon the theoretical and the practical aspects. The addressed systems cover wide technology areas (e.g., Databases, Coordination, Machine Learning, BigData), that fit different students backgrounds. This CU was first proposed and successfully taught last year, and 60% of the students anonymously "agreed" that it is "The most useful CU excluding my own area of research", whereas the remaining 40% "strongly agreed" on that.

Contents

1 Introduction and Motivations

Research and practice should nowadays go hand in hand to create value and make an impact in the society. Google, Microsoft, Amazon, Yahoo!, Facebook, Twitter, etc., are currently leading the computer technology through their successful pervasive services. This couldn't happen without embracing cutting edge research innovations and making extraordinary engineering efforts (in addition to the convenient investment and the business models). Concrete evidences are the notable number of highly cited publications of such institutions and the increasing demand on hiring PhD holders as software engineers. We believe that PhD students must have an arsenal of practical skills in addition to fundamental research; and it is thus wise and imperative to start by learning how the leading industry designed and engineer successful services.

A deeper look at these aforementioned services lead to two main observations, among others, behind their technical success: (1) the multi-layered software stack and (2) adopting distributed designs. The former indicates that even despite the "narrow" spectrum and defined target of a service (e.g., publishing tweets in Twitter) there are indeed several software layers (or say systems) underneath to provide the desired behavior and performance, starting with the network layer up to the application layer. The second observation indicates that the prime approach to build such scalable and available services, in order to reach out users all around the globe, is to design them (and often their several layers) in a distributed or replicated manner. Given that, future scientists should have a core theoretical and practical knowledge of multi-layered distributed designs, which this course tries to provide.

2 Objectives and Learning Outcomes

The objective of this curricular unit is to give the PhD student a comprehensive overview of successful systems in production considering not only the theoretical aspects, but also the practical design and implementation. The successful systems will be chosen according to their reputation or adoption, with some diversity of providers and software layers. In particular, the PhD student will understand the big picture of every tackled system, the main technical problem or service it implements, the theoretical background behind it, and, importantly, how it is/can be implemented in practice. We expect this course to yield several tangible outcomes:

- Educate the PhDs on how leading successful systems are built and designed.
- Help the PhD acquire theoretical and practical skills through learning existing successful designs to use or improve them.
- Help PhDs of different research areas to find synergies and understand potential dependencies thought addressing different systems and software layers.
- Break the ice through getting the PhD's hands dirty via confronting them with large and high quality code.

3 Covered Topics

The syllabus will address various systems across several layers like databases, communication protocols, coordination systems, BigData, machine learning, Cloud systems, Cache systems, etc. In general, the course will cover the following curated list of systems; however, we keep the flexibility to select the exact list of six topics later considering the relevance and interests of the students.

- NoSQL Databases: This topic will focus on a recently well adopted class of databases called NoSQL (Not only SQL) databases that target high availability on the expense of strong consistency. The topics will motivate Eventual Consistency and overview existing NoSQL databases, and finally dig in into a selected database among Cassandra (Apache), Dynamo [3] (Amazon), Riak [16] (Basho), or ETCD (CoreOS), etc.

- SQL Databases: This topic will address the other important class of SQL databases. Since many students have seen these DBs in undergraduate courses, the course will focus on the extremely scalable ones like Google's Spanner [8] and BigTable [7] which backbone most Google services (search, YouTube, mail, etc.). This topic is interesting due to the availability and scalability challenges imposed by the CAP theorem [4], on which many argue that ACID properties can no longer be guaranteed at a geographical scale. This topic will describe the design of these databases showing how they (claim to) achieve five nines availability.

- In Memory DB and Cache systems: A crucial layer for high available systems is to recently run an in-memory cache or database system to avoid the high overhead of storage access. In this class, Redis [28] and Memcached [19] are very well known and used by most of the technology leaders. This topic will address these systems with focusing on one of them and describing its design and implementation.

- Coordination systems: Distributed applications must have some way of coordination or synchronization to feel like a single system. Coordination services are layers that usually coordinate access to resources, maintain membership and grouping, etc. This topic will address such services in general and will focus more deeply on a selected well known coordination service as Zookeeper [13] (Apache), Chubby (Google), etc.

- Publish/Subscribe (Pub/Sub) technology: Contrary to the above topics that are often server/client models, the pub/sub model is a kind of peer to peer service where nodes subscribe to topic and wait for publisher to send events on those topics; for instance, Twitter is considered a highly scalable pub/sub system. This topic will cover state-of-the-art of pub/sub systems and elaborate more on a selected service such as Twitter, Google pub/sub (http://bit.ly/2n78xWt) and OpenDDS. We will take a closer on their design choices as a way to gain insights and allow the students to be better prepared to choose the best solution for different scenarios.

- Message brokers: This topic will focus on the communication layer that is just upper to the UDP and TCP layers. This can be viewed as a (communication) "bus" that connects all the components (brokers) within a distributed system. Successful systems that are used or developed by most technology leaders in this class are Zeromq [10], Kafka [14], etc. The course will show the importance of these systems in practice showing how they differ in their design and implementation from each other and from TCP/UDP. The topic will discuss the dissemination patters, message duplication, message dropping, multicast, message forwarding, etc. The course will more deeply focus on a selected specific message broker showing it's low level design and an overview of it's code and use.

- Cloud-computing: From its inception, cloud-computing aims to seamlessly offer unlimited resources with minimal or no setup time. The first generation was brought by Amazon AWS [1], that offers the biggest commercial offering. With the ever-increasing adoption of cloud-computing, alternatives approaches have appeared, namely allowing users to deploy their own clouds. In that regard, OpenStack [22] is the most widely deployed private cloud software. It mimics the API and semantics from AWS and allows for hybrid deployments, where public infrastructure is seamless used to offload computation when local resources are deployed. In this topic, we will present the concepts and sub-systems necessary to understand modern hybrid clouds deployments.

- Containers: The cost associated with full virtualization is prohibitive for some deployments. Namely, scenarios that do not need to support multiple operating systems may end up using more resources than necessary. Containers first appeared in Solaris in the form of *Jails* infrastructure.

It was followed by OpenVz for Linux and more recently was integrated in mainline kernel throught the LCX project. Nowadays, there is a substantial interest in this type of technology that lead to the appearance of Docker [17] and Kubernetes [5] for example. This topic will provide insights on how these systems work, their limitation and advantages. In this topic we will address the different architectural components that are need to support and orchestrate containers, from low-level kernel facilities to clustering solutions at the application level.

- Computational frameworks for Big Data and Machine Learning: This topic will focus on the current state-of-art computational frameworks, including both batching (Map-Reduce [9]) and streaming paradigms (Spark [27] and Storm [23]). Built on top of Google File System (GFS), Map-Reduce [9] offers a batch computational framework suitable for long running jobs. It is currently one of the most used tools for data analytics. As an open-source offering, Hadoop and HDFS are Apache projects that mimic the functionality and architecture of the original system. In cases where information has to be processed in real-time, batch is no longer an option. For that purpose, Storm [27] and Spark [27] were built to handle large streams of data suitable for online (real-time) responses. They do not handle storage and just focus on the actual data processing. In this topic, we will analyze the inner-works of Storm and how to apply the system to actual end-cases.

4 Practical Work

The practical part is key and innovative in this course. As described above, each topic will address some implementation details of a system in addition to its design. Given that the lecture time will be short in principle (around three hours), the student will have the duty to understand an address the design and/or implementation further, after the lecture. An innovative strategy we followed last year is to define the practical work through discussing it with the student. The purpose is to engage the student through choosing assignments that are inline with his/her research work. We give the flexibility to students according to their preferences, as well as depending on the topic, through suggesting several possibilities as follows:

- Implement a demo or a feature in an exiting code (or an open-source variant implementation).
- Reverse engineer a service though compiling the functional specifications of its code.
- Analyze and discuss other existing successful systems.
- Compare an existing successful system with another one that's addressed in the course.

5 Relevance to the Program

This curricular unit fits very well the "Technology" option of the MAP-I program, and is complementary to other units. Since the systems we address are often practical (direct or indirect) implementations of some of the theoretical research outcomes in distributed computing, some topics will naturally overlap, in the theoretical design, with the Distributed Computing unit that used to be given throughout the previous years. Contrary to that unit, which focuses on "Theory and Foundations", our proposed unit "Successful Systems in Production" focuses on a system engineering perspective and how to build Distributed Systems, and is hence more practical.

On the other hand, the chosen topics of our proposed unit are diverse enough to touch on various research areas and interests. This is crucial to benefit a notable number of students not only through the knowledge acquired, but likely through using these topics in their own research focus. For instance, people that work on distributed systems will benefit from all topics; people working on data will benefit from SQL and NoSQL databases; people working on Clouds will benefit from virtualization and containers; people working on BigData and Data analytics will benefit from BigData platforms like Hadoop and Spark; people working on networking, distributed systems, and wireless sensor networks will benefit from message brokers, etc.

6 Methodology and Grading

The course is credited with five ECTS in the European Credit Transfer and Accumulation System. Six topics will be selected considering the students' background. The first class will give an overview of the entire course to allow students choose their projects. Each topic will be addressed in a class of three hours including the follow-up of the practical work. In the last class, the students shall present their practical work through presentations and demos.

In each topic, we plan to follow this methodology:

- Motivate the topic through presenting the leading worldwide services using/developing it.
- Describe the big picture of a system to understand the overall design and work-flow.
- Overview state-of-the-art of the theoretical foundations of the solution and compare existing systems.
- Dig in to the design and implementation specifies of a chosen system, adopting an open-source code or a variant implementation when the code is proprietary.

Given the various background of different students, we plan to suggest several options as a practical work so that the student feel involved and comfortable. These options, discussed in details in Section 4, will be suggested at the end of each lecture. Homework and projects are thus assigned, discussed, and followed up at the end of lectures depending on the assignment.

Grading will be divided into two parts: one third will be given to attendance and contribution in the class, and the remaining two thirds will be direct evaluation of the assignments and the merits of the practical work.

7 Supporting Material and Books

Meant to be a complement to references given at each topic/class, we also propose a set of additional resources for students that are interested on getting more detail on any given topic.

- Fundamentals of distributed systems and databases:
  - Distributed Systems: Principles and Paradigms (2Nd Edition) [24].
  - Principles of distributed database systems, 2011 [20].
  - Distributed algorithms, Nancy Lynch, 1996 [15].
  - Principles of eventual consistency, 2014 [6].
  - Distributed Systems for System Architects (Advances in Distributed Computing and Middleware, Volume 1) [26].
- Messaging Brokers and Cache Systems
  - E-Book: Redis in Action [28].
  - Message Queues, 2017 [21].
- General Purpose Middleware Systems:
  - The raise and fall of Corba [12]

  - The essential CORBA: systems integration using distributed objects [18]
  - A new approach to object-oriented middleware [11]
- Computational frameworks and Containers:
  - Apache Yarn [25]
  - Containers and Cloud: From LXC to Docker to Kubernetes [2]

8 Lecturers

This curricular unit is proposed by Ali Shoker (University of Minho) and Rolando Martins (University of Porto) who will be the principle lecturers. Ali will be the coordinator of the unit, and is thus responsible for scheduling sessions, updating the curriculum, establishing and conducting grading procedures, and for all administrative contact with students.

8.1 Ali Shoker

Ali Shoker is currently an invited assistant professor at the Department of Informatics of the University of Minho, Braga, and Senior Researcher at INESC TEC, HASLab research unit. He holds a PhD in Informatics and Telecommunications from the University of Toulouse, France. Ali spent six months at EPFL working with Rachid Guerraoui who co-supervised his PhD thesis on Adaptive Byzantine Fault Tolerance with Jean-Paul Bahsoun. He also spent one year as a postdoc at INSA de Lyon university, France, working on anonymous P2P communication. Prior to his position as Senior Researcher, Ali joined HASLab in 2013 as a postdoc working on Data Management, Infrastructure Security and Fault Tolerance, Blockchain, and Edge Computing. Ali has been involved in several research projects among them SocEDA (France), Syncfree (EU FP7), Tec4Growth-SMILES (FCT), and he is currently leading INESC TEC's contribution within LightKone H2020 project. Ali's research focus is at the heart of this CU topics, and he published his work in top international conferences in the field. (List of publications can be seen here: `http://www.alishoker.com/`).

Ali proposed, coordinated, and taught the MAP-I CU "Successful Systems in Production" in 2017-2018. He contributed in teaching Masters courses on Distributed Computing and Security (Blockchain) with Carlos Baquero at UMinho Spring 2016 and Vitor Fontes (Fall 2017), respectively. He also taught a practical undergraduate course on Functional Programming (Fall 2017) with Alberto Simões. Ali gave a thematic seminar at MAP-I (Spring 2016) and a 4-hours tutorial at the NESUS winter school in Calabria, Italy (2017). Ali got practical skills during his work as Software Engineer at Clifton Myers Enterprises (http://www.gotocme.com/), and he co-implemented and fully implemented several distributed fault tolerance protocols and P2P messaging protocols in C/C++ and Java.

Ali is currently supervising two MAP-I PhD students working on Fault Tolerance for Distributed Data Types and Exactly-once delivery in almost infinite-scale applications. He is also co-supervising a third MAP-I PhD student with Carlos Baquero on building comprehensive platforms for Conflict-free Replicated DataTypes (CRDTs).

8.2 Rolando Martins

Rolando Martins studied at Faculty of Science of the University of Porto (FCUP), where he also obtained his M.Sc in Informatics: Networks and Systems. As part of his Masters thesis (YapDss), he researched the field of distributed stack splitting in Prolog, exploring Or-Parallelism. He also worked at EFACEC as a software engineer/architect and later as a systems researcher.

He obtained his Ph.D in Computer Science from FCUP, as a part of a collaborative effort between FCUP, EFACEC and Carnegie Mellon University, under the supervision of Prof. Fernando Silva, Prof. Luís Lopes and Prof. Priya Narasimhan. His Ph.D. research topic arose from his employment at EFACEC, where he was exposed to the difficulties underlying today's railway systems and light-rail deployments, and came to understand the scientific challenges and the impact, of addressing the issues of simultaneously supporting real-time and fault-tolerance in such systems.

He is a former member of the the Intel Science and Technology Center (ISTC), where he was involved in both Cloud Computing and Embedded Computing centers, and Parallel Data Lab (PDL) at Carnegie Mellon University (CMU). At the same time, he was also a computer research scientist at YinZcam, a spinoff from CMU that provided mobile applications for the NBA, NHL, NFL and MLS, where he was involved on cloud computing, content management systems, OAuth and video streaming.

He is currently an invited assistant professor at the department of Computer Science at FCUP and researcher at CRACS (Center for Research in Advanced Computing Systems) part of INESC-Tec. Rolando currently teaches Distributed Systems, Operating Systems and System Administration. Some of his research interests include security, privacy, intrusion tolerance, (secure) distributed systems, edge clouds, P2P, IoT, cloud-computing, fault-tolerance (byzantine and non-byzantine), operating systems (with special interest in the Linux kernel). (List of publications can be seen here: `http://www.dcc.fc.up.pt/~rmartins/`).

References

[1] Amazon. Amazon AWS, 2017.

[2] David Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1:81–84, 2014.

[3] Josh Blum. Dynamo: Amazon's highly available key-value store.

[4] E. Brewer. Cap twelve years later: How the "rules" have changed. In *Computer*, 2012.

[5] Eric A Brewer. Kubernetes and the path to cloud native. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 167–167. ACM, 2015.

[6] Sebastian Burckhardt et al. Principles of eventual consistency. *Foundations and Trends® in Programming Languages*, 1(1-2):1–150, 2014.

[7] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Michael Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26:4:1–4:26, 2006.

[8] Brian F. Cooper. Spanner: Google's globally-distributed database. In *SYSTOR*, 2012.

[9] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[10] Nicolas Estrada and Hernán Astudillo. Comparing scalability of message queue system: Zeromq vs rabbitmq. In *2015 Latin American Computing Conference (CLEI)*, 2015.

[11] Michi Henning. A new approach to object-oriented middleware. *IEEE Internet Computing*, 8(1):66–75, 2004.

[12] Michi Henning. The rise and fall of corba. *ACM Queue*, 4:28–34, 2006.

[13] Patrick Hunt, Mahadev Konar, Flavio Paiva Junqueira, and Benjamin Reed. Zookeeper: Wait-free coordination for internet-scale systems. In *USENIX annual technical conference*, volume 8, page 9, 2010.

[14] Jay Kreps, Neha Narkhede, and Jun Rao. Kafka: a distributed messaging system for log processing. 2011.

[15] Nancy A Lynch. *Distributed algorithms.* Morgan Kaufmann, 1996.

[16] Christopher Meiklejohn. Riak pg: distributed process groups on dynamo-style distributed storage. In *Erlang Workshop*, 2013.

[17] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, 2014.

[18] Thomas J Mowbray and Ron Zahavi. *The essential CORBA: systems integration using distributed objects.* Wiley New York, 1995.

[19] Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C. Li, Ryan McElroy, Mike Paleczny, Daniel Peek, Paul Saab, David Stafford, Tony Tung, and Venkateshwaran Venkataramani. Scaling memcache at facebook. In *NSDI*, 2013.

[20] M Tamer Özsu and Patrick Valduriez. *Principles of distributed database systems.* Springer Science & Business Media, 2011.

[21] Queues.io. Queues, 2017.

[22] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3), 2012.

[23] Apache Storm. Distributed and fault-tolerant realtime computation, 2014.

[24] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms (2Nd Edition).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[25] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM, 2013.

[26] Paulo Veríssimo and Luís Rodrigues. Distributed systems for system architects (advances in distributed computing and middleware, volume 1)(advances in distributed computing and middleware). 2001.

[27] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[28] Yuqing Zhu, Jianxun Liu, Mengying Guo, Wenlong Ma, and Yungang Bao. Transaction support over redis: An overview. *CoRR*, abs/1702.00311, 2017.