Informatics in Education, 2022, Vol. 21, No. 4, 635–653

DOI: 10.15388/infedu.2022.26

Designing a Master Course on Architectures for Big Data: A Collaboration between University and Industry

Andrea CONDORELLI[1], Dario MALCHIODI[2]
[1]*Viale Aldo Borletti 61/63 Milano, Marelli, Italy*
[2]*Via Celoria 18 Milano, Dipartimento di Informatica, Università degli Studi di Milano, Italy*
e-mail: andrea.condorelli@marelli.com, dario.malchiodi@unimi.it

Received: April 2022

Abstract. We describe a collaboration between Marelli and Università degli Studi di Milano that allowed the latter to add a course on «Architectures for Big Data» in its Master programme of Computer Science, with the aim of providing a teaching approach characterized by an intertwined exposition of discipline, methodology and practical tools. We were motivated by the need of filling, at least in part, the gap between the expectation of employers and the competences acquired by students. Indeed, several big-data-related tools and patterns of widespread use in working environments are seldom taught in the academic context. The course also allowed to expose students to company-related processes and topics. So far, the course has been taught for two editions, and a third one is currently ongoing. Using both a quantitative and a qualitative approach, we show that students appreciated this new form of learning activities, in terms of enrollments, exam marks, and activated external theses. We also exploited the received feedback in order to slightly modify the content and the structure of the course.

Keywords: industry-academia collaboration, computing education, big data architectures.

1. Introduction

Specializations in fields such as Artificial Intelligence (AI) or Data Science (DS) in the context of higher education, with particular emphasis on master courses, are characterized by a critical dichotomy: on the one hand, it is required that students acquire theoretical and methodological competences strong enough to be spent across a lifelong career; on the other one, it is also expected that students become acquainted with knowledge in tools that enable them to quickly and efficiently integrate in a working environment, although such tools might suffer of a relatively short technical lifetime. When dealing with settled topics, these tools are typically well-established, and reasonable standards are available (at least) in the medium term: focusing for instance on

the inference of feed-forward neural networks, scikit-learn (Pedregosa *et al.*, 2011), PyTorch (Paszke *et al.*, 2019), or Tensorflow (Abadi *et al.*, 2015) represent viable alternatives which are:

(i) Available for free.
(ii) Open-source licensed.
(iii) Very well documented.
(iv) Actually used in the working landscape.

Moreover, in such cases there is substantial agreement on the *architectural patterns* (Perry and Wolf, 1992; Stal, 2006) and/or best practices to be used in specific situations, and the above-mentioned tools directly implement (or, at least, support) these patterns and practices. For instance, still referring to the previous example about neural networks, the available libraries allow effortless use of several not-so-simple techniques related to model selection and model assessment (e.g., random search or stratified cross-validation). Things change radically if we consider relatively new topics: for instance, the use of message-based system in stream processing (Fu *et al.*, 2020) is characterized by several solutions based, e.g., on RabbitMQ (Johansson and Dossot, 2020), Apache Kafka (Garg, 2013), NATS.io (Quevedo, 2018), or NSQ (NSQ Authors, 2022). These technologies are not actually interchangeable (as essentially happens, for instance, with the binomial pythorch/tensorflow). As a side-effect, there is lesser consensus about architectural patterns, and – as a consequence – about best practices.

Given this situation, it is almost customary that Universities offer courses on different *nuances* of neural networks or machine learning, not necessarily within Computer Science / Computing degrees, or even whole programmes on DS and AI, while the architectural side of the discipline is generally less frequently taught. There is therefore room in tertiary educational offers also for these topics[1], although the related expertise is more likely to be settled outside the academic realm, in which methodology and best practices emerge naturally. This brings up the opportunity of considering collaborations between University and Industry grounded on the design of educational modules / courses in areas of common interest (Ying, 2008; Camacho and Alexandre, 2019). This paper describes a recent collaboration of this kind, in which the University of Milan (Università degli Studi di Milano, UMIL for short henceforth)[2] and Marelli[3] jointly designed a course on «Architectures for Big Data» within the Master in Computer Science (CS Master henceforth, for the sake of brevity) to address the previously described gap and to foster new capabilities for the academic vocation of tertiary education (Dooley and Kirk, 2007).

The paper is organized as follows: Section 2 describes the academic context with specific focus on the CS Master in the considered University, while Section 3 is devoted to introducing Marelli and the reasons behind its collaboration with UMIL. Section 4 details how the «Architectures for Big Data» course was jointly designed, subsequently reporting about its first two editions. Some concluding remarks end the paper.

[1] We believe, however, that this might also promote advances on research activities.
[2] `https://unimi.it/en`
[3] `https://www.marelli.com/company-profile/`

2. The Educational Context

The Computer Science Department of UMIL[4] is currently one of the biggest Italian Academic institution devoted to research and higher education in the Computing field, with its faculty staff amounting to around ninety active professors and researchers. It organizes and offers four BSc and two MSc degrees rooted in Computer Science, and it contributes to specific educational offers focused on AI, Bioinformatics, Cultural Heritage, and DS, involving around five thousand students. In particular, its CS Master programme has been reorganized in 2014, giving students a very high flexibility in the choice of their courses. Specifically, students should earn 120 ECTS (European Credits Transfer and Accumulation System (European Commission, 2022)), in line with analogous European degrees. But they are free to compose a customized curriculum by choosing 18 ECTS within a set of foundational courses, 48 ECTS from two additional tables (respectively containing disciplinary and affine courses), and 12 ECTS virtually attending any course offered by the university (although some consistency in the resulting curriculum is requested). Testing of English language skills and thesis work complete the remaining credits (see Table 1 for a detailed description of the constraints to be satisfied in organizing a curriculum). Furthermore, students can directly refer to the following set of *learning paths*, that is, curricula already fulfilling the constraints in Table 1:

- Algorithms and Fundamentals.
- Analytics and Optimization.
- Artificial Intelligence.
- Industry and Business Informatics.
- Music Information Science.
- Machine Learning and Data Science.
- Methods and Models for Software Design and Development.
- Mobility and Pervasive Computing.
- Perceptual Computing.
- Video Game.

Table 1
Constraints for the curriculum of the CS Master at UMIL

Activity	ECTS
Foundational courses (computing)	18
Disciplinary courses (computing)	30–36
Affine courses (computing and other areas)	12–18
Elective courses (computing and other areas)	12
English language skills	3
Thesis	39

[4] `https://www.di.unimi.it`

In particular, one of the authors is the reference person for the *Machine Learning and Data Science* path, proposed from 2015/16. The targeted learning objective is that of training experts with an in-depth knowledge of the computing discipline, and of its applications in fields characterized by the need to extract information in contexts where (possibly massive) data comes from heterogeneous sources, or are affected by errors or uncertainty. Thus, the related curriculum is focused on data management on distributed platforms, large-scale data processing, and application of intelligent data analysis techniques. As a consequence, graduates are expected to profitably integrate in work contexts focused on risk analysis, advanced data management, intelligent data analysis, anti-fraud prevention, and optimization in Web environments. As the analysis of data is nowadays permeating all working environments, there is evidence that the *Machine Learning and Data Science* path represents a strong opportunity to enhance the placement of graduates in promising sectors such as industry, electronic commerce, banking and finance, telecommunications, insurance, health, transport and logistics, pharmaceuticals, and public administrations. The competences acquired in this path also allow a profitable continuation of studies towards a PhD in Computer Science, AI, or related fields, also in the perspective of a professional path within public and private research institutions.

The organization of this learning path underwent fine-tuning on a regular basis, considering feedback from students, teachers and industry representatives, including people from Marelli (cfr. Section 3). In is initial formulation, the learning path included core courses in the fields of Information Systems, Intelligent Systems and Machine Learning. In particular: i) the courses «Statistical Methods for Machine Learning» and «Algorithms for Massive Datasets» provided the mathematical and algorithmic competencies necessary to understand and apply the methodologies introduced in the rest of the curriculum; ii) the courses «Distributed and Pervasive Systems» and «Information Management» strengthened the students' abilities to process data in modern decision support systems; and iii) the course «Artificial Intelligence» provided an overview of the different computing approaches to data analysis. This core was developed towards two directions: a first one focusing on data management, and a second one concerning data processing and analysis. In its current organization (see Table 2), the *Machine Learning and Data Science* path is completely taught in English[5].

To the best of our knowledge, the course on «Algorithms for Massive Datasets» (taught by one of the authors) has been one of the first teaching activities within a University Master expressly focusing on big data processing organized in Italy. Since its first editions, this course had to cope with the architectures/algorithms duality. Indeed, an exclusive focus on algorithmic solutions to problems characterized by a massive amount of data would not have trained students exploiting, where appropriate, a "hands-on" learning approach. Therefore, lectures started by introducing basic tools (initially Apache Hadoop (White, 2012), with the addition of Apache Spark (Salloum *et al.*, 2016) starting from the third edition of the course) enabling students to actually write code and perform experimentations. However, the resulting teaching activity was

[5] This choice was done to foster the participation of international students, although a variant of the path mixing courses taught in English and in Italian is available for local students.

Table 2
Current organization of the *Machine Learning and Data Science* curriculum of the CS Master

Fall semester		Spring semester	
Activity	ECTS	Activity	ECTS
Algorithms for Massive Datasets	6	English Language	3
Architectures for Big Data	6	Information Retrieval	6
Artificial Intelligence	6	Two elective courses	12
Bioinformatics	6	Thesis	39
Decision Methods and Models	6		
Distributed and Pervasive Systems	6		
Heuristic Algorithms	6		
Information Management	6		
Privacy and Data Protection	6		
Statistical Methods for Machine Learning	6		

heavily biased on the algorithmic part: the course corresponds to 6 ECTS, thus only a few lectures could be devoted to the study of tools and architectural patterns tailored for data at scale. This is why, during academic year 2018/19, a formal collaboration with Marelli has been undertaken in order to organize a course on «Architectures for Big Data».

3. The Industrial Context

Marelli is one of the world's leading global independent suppliers to the automotive sector. With a strong and established track record in innovation and manufacturing excellence, its mission is to transform the future of mobility through working with customers and partners to create a safer, greener and better-connected world. With around 54,000 employees worldwide, the Marelli footprint includes 170 facilities and R&D centers across Asia, the Americas, Europe, and Africa, generating revenues of 1,380 Billion JPY (10.6 Billion EUR) in 2021.

Marelli – likely as well as most car-related manufacturing companies – is facing two huge transformations: electrification and data. The former is more straightforward: if one needs to create new products, she also needs to invest on people and machines that will support the company in such change. The latter is more tricky: the company needs to find out a way to realize a full digital transformation. So the innovation that Electric Engines is pushing to manufacturing, AI and Advanced Analytics are pushing to any functional process that supports the company core business. Even if in these years one of the critical points has been hard shortages – gas and oil shortages due to war, and chip shortages due to the COVID, as well as to an increased need – there always has been another kind of shortage: students with knowledge, skills, and competency close to the needs of the company. With the aim of coping with this shortage, Marelli established connections with several Universities, and specifically with UMIL:

in particular, since 2017 the authors organized external seminars in the courses «Statistical Methods for Machine Learning» and «Algorithms for Massive Datasets» and co-taught a PhD course on «Architectural Patterns for Distributed Machine Learning Applications» in 2020.

4. The «Architectures for Big Data» Course

Most of Data Experts on the market focus only on the analytical and algorithmic part of the dichotomy mentioned in the Introduction: the interview question "How can I bring this idea to production?" caused more casualties than a whole Mixed Martial Art night. And this is fair: the most innovative part of a project is obtained through new algorithms and approaches, and people who recently graduated (or who is in the final stages of the graduation process) likely feels comfortable with that. But for the industry, this is not enough: there are other – for sure less noble, although critical – issues such as the return on investment, the benefit over cost analysis, the maintenance costs, or more in general the sustainability of the project. It is therefore advisable that Academy and Industry collaborate in the design of parts of the learning activities offered to students, possibly focusing on an intertwined presentation of concepts, methodologies, and tools. Starting from this shared belief, UMIL and Marelli decided to jointly design a course on «Architectures for Big Data» (ABD henceforth) to be added to the educational offer of the CS Master. In particular, this course has been conceived as part of the *Machine Learning and Data Science* learning path, although students are also allowed to insert it in a customized curriculum (see Section 2), as well as to choose it as elective activity within other learning paths.

4.1. *Course Conception*

Besides the already mentioned aim of contributing to the revision of a (small) part of the educational offer within the CS Master, so as to make it closer to a given industry need, Marelli joined the challenge of designing the ABD course for the following reasons: first of all, the course aims at bringing some kind of unique perspective to students: the more in advance they can understand nine out of ten taken decisions, also basing on some kind of business/money rationale, the better; moreover, participating in this learning activity is a great opportunity to establish a two-way interaction with students, getting fresh ideas from young minds and proposing theses.

We run a preliminary qualitative analysis on a set of around one hundred job interviews for post-graduate positions opened between 2018 and 2020. Such analysis underlined common lacks of knowledge in the background of students. Mostly notable, we experienced a poor knowledge of: (a) some internals of DB management, e.g., ACID transactions, (b) big-data related frameworks, (c) MapReduce patterns, and (d) the economical impact of technical solutions w.r.t. an overall project. Having this in mind, we designed the ABD course aiming at the following targets:

- To provide students with the knowledge of distributed computing frameworks, with special emphasis on Apache Spark (Salloum *et al.*, 2016).
- To let students become competent on Software Architectures (Perry and Wolf, 1992).
- To explain workflows and attention points w.r.t. decisional processes within a company (Chung *et al.*, 2003).
- To focus on the business/money side of any challenge (Pittarese, 2009).
- To challenge some *de facto* standards (e.g., uServices (Thönes, 2015)) with the aim of helping students make their own opinions.
- To organize workshops in which external experts challenge key messages conveyed by the teacher himself (Reichlmay, 2006): these workshops involved the participation of Google, Marelli, and Artea.com in 2021 and of Microsoft, Amazon Web Services, Marelli, and Artea.com) in 2022.

The course is taught by one of the authors, and it is active since the 2020/21 academic year: the first edition was fully held remotely, due to COVID emergency, with 40 students and a high percentage (close to 90%) of them successfully taking the exam. The 2021/22 edition (a third edition is currently ongoing) was organized in a hybrid modality, with some students in class and others attending from home via a streaming platform. The number of participants roughly doubled; moreover, among the 76 students attending lectures, 45 successfully took the exam during the first available session. A pretty good Return On Investment was obtained through theses: during the first edition, one student decided to work on a thesis focused on topics related to the course, graduating in the same academic year; three theses are currently ongoing and more than five students expressed an interest for a thesis on ABD-related topics. We remark that all the mentioned theses are (or have been) done externally from the University, precisely in industrial contexts. Overall, students expressed good feedbacks on ABD, with KPIs (Key Performance Indicators) pretty close to the other courses of the CS Master: the focus on an industrial perspective on each topic was one of the most common (positive most of the time) feedback given, as illustrated in Table 3, which contains some excerpts from the platform used by the University to collect the course evaluations provided by students (see also Section 4.4).

Table 3

Some of the anonymous comments on the ABD course from students of the 2021/22 cohort

Comments
"*The course has a lot of interesting topics, but maybe it's a bit too industry-imprinted. This leads to a difficulty of comprehension by a lot of students*"
"*The course is very interesting and it provides useful knowledge to find a job, the professor really makes an effort to make you understand not so easy topics*"
"*Just a great course. Too often university forgets that computer scientists exist to support business activities. This architectures for big data course with real business experience flavour fills the gap*"

4.2. *Course Structure*

The ABD course has been designed using a top-down strategy: starting from hundreds of post-graduation hiring interviews for Junior positions, a list of weaknesses shown from the interviewed people (mostly former students who recently graduated) has been created. The more important items in this list were related to:

- Formal knowledge and definitions on traditional topics (e.g., "How would you define the term «Design Pattern»?" (Gamma *et al.*, 1995)).
- History and reasons behind some topics/technologies (e.g., "Why we use the term «Architecture» in software development?" (Perry and Wolf, 1992)).
- Knowledge on Big Data frameworks such as Apache Spark (Salloum *et al.*, 2016) – most of the times, this topic was said to have never been heard before the interview.
- Capability to explain an end-to-end solution for a given problem, encompassing both the technical and economical points of view.

The course has been proposed with the mission to fill these gaps, or at least to try doing it, giving students additional tools allowing them to deal with job interviews. Software architecture is a complex topic to teach because of the "fuzziness" of the concept itself (Galster and Angelov, 2016), and often most of architectural decisions are just opinions – there are no clear solutions, either good or bad (Rupakheti and Chenoweth, 2015). Furthermore, quoting Galster and Angelov (2016), "the nature of software architecture is practical rather than theoretical (like fundamental mathematics). Therefore, to provide a real practical experience, teaching architecting activities require a suitable context and problems of sufficient complexity (i.e., big enough, several quality attributes)".

The ABD course is delivered as 24 two-hours (almost) frontal lectures, organized as described here below:

- Basic topics related to Software Architecture, such as design patterns, definition of Software Architecture, and its links with building Architecture (six hours). Each year a real-world case from a project handled by Marelli is presented and used to exemplify what are the skills of a *Software Architect*, from functional analysis up to post-go live support.
- Hadoop architecture – as also suggested by Demchenko (2019) – and its evolution towards Apache Spark (four hours), starting from the seminal paper describing the Google approach to distributed file systems (Ghemawat *et al.*, 2003).
- Spark Coding, both in the form of frontal lectures on some programming patterns (e.g., getting the closest element using the MapReduce abstraction (Lee *et al.*, 2012)) and collaborative laboratories where students write code (ten hours)[6].

[6] It is worth noting that, on the average, students find the MapReduce abstraction pretty hard to understand, in particular in all cases in which most of the foundational concepts of traditional software development are simply not applicable (e.g., when dealing with iteration, or with any algorithm based on sorting). Such difficulties have been addressed through focused examples and exercises. The use of visual languages specifically designed for learning programming strategies might be of help (Feng *et al.*, 2017), although it should be carefully considered in view of the limited available time within the course, taking also in mind the considerations done at the beginning of the Introduction.

- Data patterns, with a focus on Change Data Capture strategies, Extract Transform and Load, and Data Lake (eight hours).
- Service Oriented Architecture (Komoda, 2006) and how this style could be used nowadays (two hours).
- Use of Big Data technologies alongside traditional ones, e.g., how to write from Apache Spark to a SQL Server (four hours).
- "As A Service" pattern, with focus on its exploitation when the architecture is being designed to move costs between CAPEX (capital expenditure) and OPEX (operative expenditure) (two hours).
- Business and cost-related topics, such Return on Investment, Benefit over Cost analysis, and types of costs (four hours).
- Four two-hours workshops with Industrial Guests (eight hours).

The syllabus has been slightly modified between the first and the second academic year, reducing the number of topics to give more time to focus on each of them, and this choice has been appreciated by students. The original idea was that of following the pattern shown by (Asamoah *et al.*, 2017), presenting both standard frameworks and commercial ones (e.g., Elastic Search, Microsoft Synapse, and so on) but it ended up being too complex for students. Table 4 illustrates the differences between lectures in the two editions of the course, linking the former to the related topic within the categorization proposed by the 2020 ACM/IEEE report on Computing Curricula (CC2020 Task Force, 2020) and detailed in Table 5.3 of this report: in particular, the following topics are involved: 1.4. Enterprise Architecture, 1.5. Project Management, 2.2. Systems Analysis & Design, 3.2. Intelligent Systems (AI), 3.3. Internet of Things, 3.4. Parallel and Distributed Computing, 3.8. Platform Technologies.

Table 4

Comparison between the 2020/21 and the 2021/22 editions of the ABD course. The **Lecture** and **Topic** columns show the title of each lecture and the corresponding topic within Table 5.3 of the ACM Computing Curricula 2020 topic categorization (CC2020 Task Force, 2020): precisely: 1.4. Enterprise Architecture, 1.5. Project Management, 2.2. Systems Analysis & Design, 3.2. Intelligent Systems (AI), 3.3. Internet of Things, 3.4. Parallel and Distributed Computing, 3.8. Platform Technologies. Checkmarks identify the academic year in which each lecture was delivered

Lecture	Topic	2020/21	2021/22
Why do we need an Enterprise Architecture	1.4	✓	✓
From Building Architecture to Software Architecture	1.4	✓	✓
Software Architecture Pillars	1.4	✓	✓
Introduction to Design Patterns	2.2	✓	✓
Big Data Design Patterns	2.2	✓	✓
Change Data Capture (CDC) Pattern	2.2		✓
Service Oriented Architecture	2.2	✓	✓
Introduction to Big Data: Hadoop	3.4	✓	✓
"Hadoop Components & HDFS & and Map Reduce"	3.4	✓	✓
Delta Lake	3.4	✓	
How to use docker to build your own Apache Spark Cluster	3.4.	✓	✓

Continued on next page

Table 4 – continued from previous page

Lecture	Topic	2020/21	2021/22
Apache Spark Theory	3.4	✓	✓
Apache Spark: Coding	3.4	✓	
Apache Spark Practical Workshop: visual representation of Map Reduce	3.4		✓
Apache Spark Practical Workshop: commutative reduction	3.4		✓
Apache Spark Practical Workshop: Distributed Nearest Neighbor search	3.4		✓
Apache Spark Practical Workshop: mean and median distribution	3.4		✓
Apache Spark Practical Workshop: Cosine Similarity	3.4	✓	✓
Apache Spark Practical Workshop: tackling a Natural Language Processing challenge	3.2		✓
From Apache Spark to SQL: how to write into a Database	3.4	✓	✓
SQL: focus on what happens behind the scene with indexed tables when a query is submitted	3.4	✓	✓
Introduction to ELK Stack (Elasticsearch & Logstash & Kibana)	3.4	✓	
Big Data and Big Money: the impact of economics on any IT project	1.5	✓	✓
Big Data and Big Money: x-as-a-service pattern and its impact on costs	3.8	✓	✓
Big Data and Big Money: a real Marelli project end to end	1.5	✓	
External Workshop: Data Democratization	3.4	✓	
External Workshop: The Data Scientist Journey	3.2	✓	
External Workshop: A Deluge of IoT Data Sources: Blockchain SCM & Connected Vehicle	3.3	✓	✓
External Workshop: Google Cloud Platform workshop	3.4	✓	
External Workhop: Let's talk of Presto	3.4		✓
External Workshop: Ferrari & Epic Games & and Finra Data Projects	3.4		✓
External Workshop: How true is the Myth?	3.4		✓

Workshops represented one of the most important parts of the course: indeed, bringing in other perspectives from big companies is a pretty effective way to show that there is no "one-size-fits-all" solution, as well as that different opinions could exist on a given topic. This is why a big effort was spent to organize the Workshops calendar: so far, the topics listed in Table 5 have been presented.

Table 5

Workshops organised during the first two editions of ABD

2020/21	
D. Malagoli (Google)	Data Democratization and Data Driven company of the future
D. Malchiodi (UMIL)	A (partial) view on the Google cloud platform ecosystem
R. Tomasi (Marelli)	A Deluge of IoT Data Sources: Connected Vehicle & Blockchain
S. Rola (artea.com)	Why an Architecture could be a game changer even for the Machine Learning World
2021/22	
D. Colombatto (AWS)	Amazon Web Services Workshop
G. Martinelli (Microsoft)	Let's talk of Presto
R. Tomasi (Marelli)	A Deluge of IoT Data Sources: Connected Vehicle & Blockchain
F. Palladino (artea.com)	The Rise and Fall of Microservices, Chapter 2 – "To Kubernetes and Beyond!"

4.3. *Challenges, Workshop, and Exam Structure*

In both editions, several challenges were proposed to help students keep pace with the course. Each challenge awards student with some additional points for the final exam. During the first year, such challenges have been done pushing a lot on competition and gamification, probably a bit too much. Each challenge had a very strict deadline and the initial idea was to award only the first team with the correct solution. The following list describes each of the proposed challenges.

- **Challenge 1**:
  - write an algorithm to distribute the computation of mean and median over a dataset using MapReduce;
  - run the algorithm on the dataset using Apache Spark and Docker.
- **Challenge 2**: given a manufacturing plant where there are some machines that transform materials,
  - collect each machine status change event (i.e., when each machine change its status) in a big-data Data Lake;
  - compute the average duration per each status.
- **Challenge 3**: given a music listening dataset (i.e., a set of (play, user) interactions), write an end-to-end script to
  - compute the average number of replays for each user (i.e., how many times in average each user listens to songs) organizing them in a suitable data structure, e.g. a dictionary having users as key and averages as values; and
  - show the corresponding histogram.
- **Challenge 4**:
  - design and develop a simple CDC;
  - try to implement an abstract class for this CDC, containing all the needed abstract and concrete methods.
- **Challenge 5**: given a manufacturing plant where there are some machines that transform materials,
  - collect each cycle time (i.e., the time needed to finish a machine cycle) in a big-data Data Lake;
  - compute the average cycle time per machine;
  - propose an approach to identify anomalous machines.
- **Challenge 6**:
  - create a synthetic data generator to simulate log data (i.e., data that can be only inserted);
  - create a synthetic data generator to simulate registry data (i.e., data that can be inserted/updated/deleted).

After some complaints, to deal with working students and other kind of personal issues, the teacher decided to soften the initial constraints, awarding any student with a correct solution on less challenging deadlines. In view of this experience, during the second edition of the course the following challenges, in a smaller number and with smoother deadlines, have been proposed.

- **Challenge 1**:
  - write an algorithm to distribute the computation of mean and median over a dataset using MapReduce;
  - (not mandatory) run the algorithm on the dataset using Apache Spark and Docker.
- **Challenge 2**:
  - write an algorithm to compute an histogram with Apache Spark;
  - write an algorithm to compute the cosine similarity with Apache Spark.

The assessment of student knowledge, abilities and competences has been affected by the COVID pandemic during the first edition of the course. The simpler way to deal with this situation was that of running the exam from remote: each student got a one-hour oral test, where s/he was asked to write code (via screen sharing) involving the use of some of the patterns seen during the course. The list below shows some examples of the questions that were asked:

- Given a context (e.g., a production plant with some machines) compute the cosine similarity between items, or the average number of produced pieces.
- Show a situation where Spark laziness hides some bug, and show how to mitigate that risk.

After this first "practical" part, some theory-based questions were posed, exemplified in the following list:

- What are the Software Architecture pillars?

Table 6

An example of the practical part of the ABD exams during the second edition of the course

Each time a customer would like to post an order to our Company, this can be done through Electronic Data Interchange (EDI). EDI is a standardized way to communicate between companies. The received orders are stored in the `EDIQueue` table. Each EDI message contains, among other fields:

- *Number: unique code for the Order.*
- *CustomerNumber: unique code for the customer.*

There are two kinds of orders:

- *Open Orders: agreements between us and our customers to ship a total amount of pieces over a fixed time window.*
- *Closed Orders: one-shot orders.*

Answer to the following questions:

- *Define the type of each involved table (Log or Registry): which are the keys of these tables?*
- *Architect a strategy to integrate them through a CDC job. You can propose a pseudo-code solution for the CDC job, a SQL based solution, or a mix of the two.*
- *It might happen that an EDI message with a LoadTS in the future is created: how does this impact the CDC logic?*
- *How can we solve this issue?*
- *Compute the total money paid by our customers.*
- *Compute the total money paid by our customers in 2021.*
- *How many Open Orders has been completely shipped?*
- *How many FinishedGoodPN are sold to multiple customers?*
- *Compute the amount of Open quantities for each order.*
- *Estimate the amount of Closed Orders.*

- What does SOA mean? What are the advantages of such Architecture Style?
- What does the term «design pattern» mean?
- Which design patterns can be used in order to reduce the vendor lock-in risks?
- Which is the difference between task and stage in Spark?
- Describe the sequence diagram of a Spark Execution.

The second year was less impacted by COVID. The practical part was evolved in a written test, with a detailed description of the environment, and several questions about it. On top of Spark questions, students were asked to develop a CDC strategy for the given context. The theory part was either oral or written, as per student preference, with similar questions to the previous year. Table 6 shows an example of practical exam[7]. The assessment, expressed on a scale between 0 and 30, was done by the course instructor, taking into account the level of mastery of the topics, clarity, and language skills.

4.4. *Statistics and Feedbacks*

Computing statistics describing students attendance is pretty hard within the CS Master at UMIL, since students can decide to attend lectures without passing through a formal per-course enrollment process. Analogously, there are no constraints related to signing in for an exam, thus students can just decide to take it in any of six pre-planned sessions within an academic year, regardless they actually attended the lectures or not. For this reason, besides the analysis of evaluation surveys sent by student, which we will described later on, we considered the following KPIs, whose values are partly illustrated in Table 7:

Amount of interested students number of students who subscribed to the Web site of the course.

Amount of involved students number of students who had some kind of connection with the teacher through e-mail.

Amount of successful students number of students who have passed the exam, normalized w.r.t. the number of students who attended the lectures.

Table 7

Key Performance Indicators selected for the ABD course evaluation. The column "# Students" measures the amount of involved students, while "# Exams" and "% Exams" describe the amount of successful students. As an additional information, "Average Mark" and "# Surveys" illustrate the average mark of successful students and the number of received surveys for the course evaluation

	# Students	# Exams	% Exams	Average Mark	# Surveys
2020/2021	44	23	52.27%	28	40
2021/2022 (current data)	76	36	47.37%	27.38	50
2021/2022 (forecast)	76	45	59.21%	N/A	50
YOY Increase (forecast)	72.73%	95.65%	13.27%	–2.21%	25.00%

[7] For the sake of brevity, we choose to only show the text of one of the proposed practical parts, as UMIL has six examination sessions per year.

The amount of involved students almost doubled in just one year, raising from 44 to 76, while the amount of students who subscribed to course Web site for the current academic year (as of June, 17th) is 128, a very interesting number compared to *the whole cohort of* 144 *new enrolled students* in the academic year 2021/22. The normalized amount of students who passed the exam has been more or less stable in the two years, as well as the average mark, which is more or less aligned with that of the other courses of the CS Master. More precisely, in the academic year 2020/21 the average mark of the ABD exams is slightly higher (around one point of absolute difference) than the average mark computed over all exams of all courses of the CS Master, in turn amounting to 26.49[8]. As a reference base for giving a meaning to these numbers, on the average there are 30 *students* attending an elective course in the CS Master. Taking in mind that students are free to decide whether attending lectures and taking exams of a course in their first or second year (cfr. Section 2), a more meaningful comparison should be done using the sum over the first two years w.r.t. the number of students in the corresponding cohorts. In 2021/22, there are 456 students enrolled in the Master[9]: 26% of them were somehow interested with this new – and non mandatory – course, while 14% has already passed the exam. An estimation of fresh students only could be obtained by considering a steady amount of new students per year, bringing the number to be used for computing this statistics to roughly 300 students: under this assumption, the aforementioned numbers grow to 40% and 22%, respectively. A last interesting point is the number of theses: as already said in Section 4.1, several students are involved in ongoing theses. It is however important to say that only one of them focuses on an ABD topic (namely, event-oriented architectures for the application of anomaly detection algorithms on the service execution of a Service Oriented Architecture). Indeed, most of the students are simply interested in working on "external" theses, and more precisely on projects taking place in industry/private firms. The feedback we had from the representatives of these firms are on the average firmly positive, and roughly in 40% of the cases the student evaluation was good enough to get a hiring proposal from top companies within the field of big data.

Quoting Brown *et al.* (2013), "there is no doubt about the importance of assessment: it defines what students regard as important, how they spend their time and how they come to see themselves – it is a necessary part of helping them to learn". In line with this statement, UMIL constantly monitors the educational offer in compliance with its quality policy, requiring that students willing to take an exam compile an evaluation survey for the corresponding course (Università degli Studi di Milano, 2022). The results of this evaluation procedure were particularly helpful in shaping a qualitative analysis of the course outcomes. In particular, we focused on the following questions[10]:

[8] In the Italian tertiary education system, positive marks are expressed in the scale 18–30, with the possibility of a full mark with honours.

[9] This number includes first year and second year students, as well as students who attended two years and did not pass all exams or defend their thesis (note that in the considered educational system, students can renew their enrollment as many years as they want, and take exams on any academic year after the one in which they attended a course).

[10] The item enumeration refers to the ranking used in the original evaluation survey: some numbers are missing because the corresponding questions were not included in our analysis.

1. Were the preliminary learnings sufficient for understanding the topics set out in the examination syllabus?
2a. Were the objectives and content of the course presented clearly?
3. In your opinion, did the course reflect the learning objectives set out in the Degree Programme?
4a. If the course syllabus includes topics already dealt with during other courses as part of the Degree Programme, did you find these repetitions helpful?
5. Are you interested in the topics dealt with during the course?
6a. Was the course load proportionate to the credits assigned, also in relation to the examination syllabus?
9. Were the teaching materials (suggested and available) suitable for studying the subject?
10. Were the examination methods set out clearly?
11. Were you satisfied, on the whole, with this course?
13. If teaching activities have been carried out online (video lessons, multimedia films, hypertext units...) on the whole were you satisfied?
14. Did the teacher stimulate/motivate students to take an interest in the subject?
15. Did the teacher set out the topics in a clear and comprehensive manner?
16. Was the teacher readily available to provide clarifications and explanations?
17. Did the teacher behave in a correct and helpful manner towards the students?

Table 8 summarizes the results of these surveys for the ABD course, comparing them with the averages for the other courses in the CS Master. It is evident that most of the results are aligned to those at the Master level. Note, however, that the number of answers is relatively low, and this implies a relatively high uncertainty in the predictive value of

Table 8

Students' feedback on evaluation survey for the ABD course vs. the average of other CS Master courses. Values in the **ABD** and **Master** columns are expressed in the 1–10 scale

	2020/21			2021/22			
	ABD	Master	ABD/Master	ABD	Master	ABD/Master	YOY ABD
# Questionnaires	26	-	-	35	-	-	34.62%
Question 1.	7.69	7.91	97.22%	7.94	7.95	99.87%	2.65%
Question 2.	8.04	8.06	99.75%	7.71	8.23	93.68%	-6.07%
Question 3.	8.58	8.48	101.18%	8.26	8.34	99.04%	-2.13%
Question 4a.	8.55	8.27	103.39%	8.27	8.2	100.85%	-2.53%
Question 5.	8.46	8.61	98.26%	8.31	8.41	98.81%	0.55%
Question 6a.	8.42	7.27	115.82%	7.51	7.72	97.28%	-18.53%
Question 9.	7.73	7.91	97.72%	7.2	8.03	89.66%	-8.06%
Question 10.	8.77	7.88	111.29%	6.46	8.03	80.45%	-30.84%
Question 11.	8.31	7.72	107.64%	7.46	7.6	98.16%	-9.48%
Question 13.	8.14	8.04	101.24%	7.08	8.13	87.08%	-14.15%
Question 14.	9.19	8.04	114.30%	8.54	7.85	108.79%	-5.51%
Question 15.	7.96	7.93	100.38%	7.34	7.91	92.79%	-7.58%
Question 16.	9.31	8.55	108.89%	8.74	8.79	99.43%	-9.45%
Question 17.	9.35	8.71	107.35%	9.03	8.88	101.69%	-5.65%

averages. For instance, a small set of unhappy students could greatly impact the final results. The most interesting interpretations of the results are listed here below:

- Q9 No standard teaching material has been provided. The main material has been the lectures themselves, as well as any clarification meeting needed. On top of that, only slides and exercises have been shared, no textbooks or other kind of traditional materials.
- Q10 The teacher commitment was not to impose strict bureaucratic rules. So, to deal with different situations over time, some rule has changed to make the students life easier. This low result probably is a mix of students with bad results in the exam and students which lost some time during exam preparation.
- Q15 To stress out the importance to live the ABD course in real time and cooperate with the professor, no lecture has been recorded. This decision created a lot of discussion, but it has not changed (this is also related to Q9).
- Q14 The ABD course has been designed with this mission in mind. Several strategies have been put in place, from offering candies as prize for any student question/answer, to allowing students to get extra points for the exam through some challenges, i.e., homework (in Italian universities there is no such habit of giving and correcting homeworks).

These feedbacks are beyond any expectation but aligned with other similar works (Valiente Bermejo *et al.*, 2021): all the experiments done in the first year allowed to correct some structural issues, to provide students with a better experience. A lot of focus has been put in designing the concept of homeworks: to move them from a *"boring high-school-like task"*, they were called *challenges* and they were not mandatory, although they contributed to the final course mark through additional points given as a reward. To make them even *sexier*, an anonymous leader-board has been shared during lectures to add a *gamification* taste (Swacha, 2021) to the process, "to motivate and engage students in their learning process" (Martí-Parreño *et al.*, 2016): during the first ABD edition, these additional points were awarded to the quickest solution or to the smarter ones. To further increase competition, *best students* were given the opportunity to show their solution to the class during lectures. However, we realized that this approach was probably too much pushing for several students, as they were not used to perform in a highly challenging environment and a lot of complaints arose (e.g., *"It's not fair because I'm attending other X classes"*, *"I'm a part-time student"*, and so on.) For these reasons, during the second year deadlines were relaxed, no additional points were awarded, and the leader-board was no more shown: this lighter version of gamification made students more comfortable as per their feedbacks.

5. Conclusions

This paper described how UMIL and Marelli designed, implemented and conducted two editions of a CS Master course on «Architectures for Big data», expressly tailored for a *Machine Learning and Data Science* learning path, yet available for all students. This course was motivated by the need of teaching software architectural patterns for data at

scale, using an approach that mixed disciplinary and methodological contents with the use of tools of widespread use within working environments. The main end was that of trying to contribute to train experts having a long-time theoretical knowledge coupled with practical skills that are desirable from the perspective of employers. The course was grounded on architectural patterns, although also business and cost-related topics were considered, in the idea that the latter can promote a rapid integration of graduated students in the working processes when they get hired. As a distinctive feature, the course also offered several workshops held by representatives of big companies and private firms, with the aim of bringing in also the perspectives of employers. The obtained results are more than encouraging: the number of involved students doubled from the first edition to the second one, ant it is currently above the average w.r.t. the CS Master, as well as the academic performance in terms of obtained marks. Moreover, several students exploited the above-mentioned connections with private firms, engaging with them for external theses. Therefore, at least in part the initial expectations were met. At the current date, the course has been offered for two academic years, giving the instructor valuable feedbacks that allowed a fine-tuning process concerning both disciplinary content and course organization. As further improvement, we will work on a tighter integration between the course and the remaining activities within the *Machine learning and data science* learning path. For instance, we are convinced that the collaboration between UMIL and the industrial context should be strengthened, for instance putting in a different light the above mentioned workshops/seminars, so that they become an activity devoted to a broader student audience. We also envisage the establishment of an *external board* offering advice w.r.t. structure and contents of the learning path, as well as evaluating to which extent it meets the expectations of firms willing to hire graduates skilled in managing and analyzing data at scale.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. `https://www.tensorflow.org/`

Asamoah, D.A., Sharda, R., Hassan Zadeh, A., Kalgotra, P. (2017). Preparing a Data Scientist: A Pedagogic Experience in Designing a Big Data Analytics Course. *Decision Sciences Journal of Innovative Education*, 15(2), 161–190.

Brown, G.A., Bull, J., Pendlebury, M. (2013). *Assessing Student Learning in Higher Education*. Routledge, London.

Camacho, B., Alexandre, R. (2019). Design Education. University-industry collaboration, a case study. *The Design Journal*, 22(sup1), 1317–1332.

CC2020 Task Force (2020). *Computing Curricula 2020: Paradigms for Global Computing Education*. Association for Computing Machinery, New York, NY, USA. 9781450390590. `https://doi.org/10.1145/ 3467967`

Chung, P.W.H., Cheung, L., Stader, J., Jarvis, P., Moore, J., Macintosh, A. (2003). Knowledge-based process management – an approach to handling adaptive workflow. *Knowledge-Based Systems*, 16(3), 149–160. `https://doi.org/10.1016/S0950-7051(02)00080-1` `https://www.sciencedirect.com/science/article/pii/S0950705102000801`

Demchenko, Y. (2019). Big Data Platforms and Tools for Data Analytics in the Data Science Engineering Curriculum. In: *Proceedings of the 2019 3rd International Conference on Cloud and Big Data Computing*, pp. 60–64.

Dooley, L., Kirk, D. (2007). University-industry collaboration: Grafting the entrepreneurial paradigm onto academic structures. *European Journal of Innovation Management*, 10(3), 316–332.

European Commission (2022). European Credit Transfer and Accumulation System (ECTS). Accessed 18 June 2022: https://education.ec.europa.eu/education-levels/higher-education/higher-education-initiatives/inclusive-and-connected-higher-education/european-credit-transfer-and-accumulation-system

Feng, A., Gardner, M., Feng, W.-c. (2017). Parallel programming with pictures is a Snap! *Journal of Parallel and Distributed Computing*, 105, 150–162. Keeping up with Technology: Teaching Parallel, Distributed and High-Performance Computing. https://doi.org/10.1016/j.jpdc.2017.01.018 https://www.sciencedirect.com/science/article/pii/S0743731517300242

Fu, G., Zhang, Y., Yu, G. (2020). A Fair Comparison of Message Queuing Systems. *IEEE Access*, 9, 421–432.

Galster, M., Angelov, S. (2016). What Makes Teaching Software Architecture Difficult? In: *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pp. 356–359. IEEE.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995). *Design Pattern. Elements of Reusable Object-Oriented Software*. Addison-Weslsy, Boston.

Garg, N. (2013). *Apache Kafka*. Packt Publishing, Birmingham, UK.

Ghemawat, S., Gobioff, H., Leung, S.-T. (2003). The Google File System. In: *Proceedings of the nineteenth ACM Symposium on Operating Systems Principles*, pp. 29–43.

Johansson, L., Dossot, D. (2020). *RabbitMQ Essentials: Build Distributed and Scalable Applications with Message Queuing Using RabbitMQ* (2nd ed.). Packt Publishing Ltd, Birmingham, UK.

Komoda, N. (2006). Service Oriented Architecture (SOA) in Industrial Systems. In: *2006 4th IEEE International Conference on Industrial Informatics*, pp. 1–5. IEEE.

Lee, K.-H., Lee, Y.-J., Choi, H., Chung, Y.D., Moon, B. (2012). Parallel Data Processing with MapReduce: A Survey. *SIGMOD Rec.*, 40(4), 11–20. https://doi.org/10.1145/2094114.2094118

Martí-Parreño, J., Méndez-Ibáñez, E., Alonso-Arroyo, A. (2016). The use of gamification in education: a bibliometric and text mining analysis. *Journal of Computer Assisted Learning*, 32(6), 663–676.

NSQ Authors (2022). NSQ Docs 1.2.1 -A realtime distributed messaging platform. Accessed 18 June 2022: https://nsq.io/

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., Red Hook, NY, pp. 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Perry, D.E., Wolf, A.L. (1992). Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4), 40–52.

Pittarese, T. (2009). Teaching fundamental business concepts to computer science and information technology students through enterprise resource planning and a simulation game. *Journal of Computing Sciences in Colleges*, 25(2), 131–137.

Quevedo, W. (2018). *Introduction to NATS*. Apress, Berkeley, CA, pp. 1–18. 978-1-4842-3570-6. https://doi.org/10.1007/978-1-4842-3570-6_1

Reichlmay, T.J. (2006). Collaborating with Industry: Strategies for an Undergraduate Software Engineering Program. In: *Proceedings of the 2006 International Workshop on Summit on Software Engineering Education*. SSEE '06. Association for Computing Machinery, New York, NY, USA, pp. 13–16. 1595934073. https://doi.org/10.1145/1137842.1137848

Rupakheti, C.R., Chenoweth, S.V. (2015). Teaching Software Architecture to Undergraduate Students: An Experience Report. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (Vol. 2), pp. 445–454. IEEE.

Salloum, S., Dautov, R., Chen, X., Peng, P.X., Huang, J.Z. (2016). Big data analytics on Apache Spark. *International Journal of Data Science and Analytics*, 1(3), 145–164.

Stal, M. (2006). Using architectural patterns and blueprints for service-oriented architecture. *IEEE Software*, 23(2), 54–61.

Swacha, J. (2021). State of Research on Gamification in Education: A Bibliometric Survey. *Education Sciences*, 11(2), 69.

Thönes, J. (2015). Microservices. *IEEE Software*, 32(1), 116–116.

Università degli Studi di Milano (2022). Quality assurance. Accessed 18 June 2022: `https://www.unimi.it/en/university/quality-assurance`

Valiente Bermejo, M.A., Eynian, M., Malmsköld, L., Scotti, A. (2021). University–industry collaboration in curriculum design and delivery: A model and its application in manufacturing engineering courses. *Industry and Higher Education*, 09504222211064204.

White, T. (2012). *Hadoop: The Definitive Guide*. O'Reilly Media, Inc, Beijing.

Ying, L. (2008). How industry experience can help in the teaching of entrepreneurship in universities. *Sunway Academic Journal*, 5(1), 48–64.

A. Condorelli is adjunct professor at the Computer Science Department of the Milan University and Sr. Manager, Digital Transformation Lead, Head of Enterprise Architecture, Data Science, and Data Governance Marelli. His main areas of focus are Enterprise Architectures, Big Data, Machine Learning, Data Governance, Natural Language Processing and its applications, Recommender Engines. He is involved in several dissemination and education activities, both internally to his company, and externally, working with Universities, other companies, and making speeches in – mostly Industrial – Conferences.

D. Malchiodi is associate professor at the Computer Science Department of the Milan University. His research activities, documented in around one hundred publications, are focused on the treatment of uncertainty in machine learning, with particular focus to data-driven induction of fuzzy sets, compression of machine learning models, mining of knowledge bases in semantic Web and negative example selection in bioinformatics. He has been longtime involved in computing education for primary and secondary schools and in teacher training, as well as in several popularization activities in the computing field.