Big Data and Cloud Computing for the Built Environment

Marcin Kosicki, Marios Tsiliakos, Khaled ElAshry, and Martha Tsigkari

Abstract Designing the built environment is by default a multidisciplinary endeavour, producing an abundance of data that needs to be analysed during the process. This data is associated with specific design solutions and driven by multiple, usually competing objectives that need to be taken into consideration during fast review cycles. Quantifiable data ranges from simple area measurements, to more elaborate metrics such as thermal performance, carbon footprint or contextual integration, derived by a plethora of time-consuming analyses. The need to create a built environment which is not only functional and elegant but also energy efficient and sustainable is making performance-oriented design one of the main driving forces in contemporary architecture. A by-product of this practice is the large data sets that it can produce, which in turn raises the question of how the industry can deal with all this data—not only in terms of production, but also classification and reuse. This has been a catalyst to investigating how other industries are dealing with similar issues. The shift, for example, towards big data and the adoption of cloud computing, has enabled IT companies to dramatically increase performance and efficiency of many industries over the past years. This runs contrary to contemporary tools used for architectural computing, traditionally built around a single workstation, and their respective workflows. This problem is firstly challenged by explaining the technology behind both big data and cloud computing while comparing them to state-of-the-art computer-aided design (CAD) software. Additionally, cloud-based software development and continuous delivery strategies are analysed and the potential improvement on design pipelines, based on experience. Then a prototype of a bespoke system called Hydra will be presented, which

M. Kosicki (✉) · M. Tsiliakos · K. ElAshry · M. Tsigkari
Foster + Partners, Applied Research & Development, London, UK
e-mail: mkosicki@FosterandPartners.com

M. Tsiliakos
e-mail: mtsiliak@FosterandPartners.com

K. ElAshry
e-mail: kelashry@FosterandPartners.com

M. Tsigkari
e-mail: mtsigkar@FosterandPartners.com

M. Bolpagni et al. (eds.), *Industry 4.0 for the Built Environment*,
Structural Integrity 20, https://doi.org/10.1007/978-3-030-82430-3_6

runs on a high-performance compute cluster combining multiobjective optimization, a popular parametric CAD system and a set of building performance analyses. The data produced by the system is stored in a database and visualized using a modern web-based interface. The system is being demonstrated and assessed on a large-scale master planning project case study, where Hydra's benefits are more evident due to project's complexity and size.

Keywords Big data · Cloud computing · Hydra · Parallelization · Performance speed-up · Real time · Performance-driven design · Optimization · Distributed computing

1 Introduction

Data has become the currency of the modern society. It is, in many ways, the most abundantly generated product of our century. Every single action in our life—asking directions from Google Maps, liking a post in social media, watching a movie in Netflix or even browsing on an online retailer—produces data that is being mined and used in a variety of imaginative and profitable ways. If, therefore, our small, daily actions can produce an avalanche of information, how much data can the design, construction and operation of a building produce? The answer is: a significant amount! It is therefore unfortunate that in the architecture, engineering, construction and operation industry (AECO) the use of big data and the adoption of big data technologies is substantially less developed than in other industries [1, 2].

Many people will argue that data is power, but the authors also argue that it is not one that can be wielded easily. Producing, collecting, processing, structuring and classifying data to be used in a meaningful manner is a non-trivial task, particularly for an industry as multidisciplinary and complex as the AECO is (see Fig. 1). The rise of performance-driven design has provided the perfect testbed for such an assertion. The requirement to produce design solutions that are driven not only by aesthetical criteria, but also by hard metrics such as their structural, environmental or social performance has posed many challenges. These include the complicated and time-consuming analyses that need to be run on a timely fashion and productively inform the design process [3], but also how the knowledge acquired in one project could become an input for the next one.

In these workflows, every single design choice (from conception to completion) can be based on a plethora of information derived from a multitude of specialist analyses. These, in turn, are driven by multiple, usually competing objectives that need to be taken into consideration during fast review cycles [4]. But how is all that information processed? How do the qualitative differences between options become quantifiable? And ultimately, how can the industry learn from it for future projects—how can it be gathered and used in a structured way further down the line? Obviously, the latter is not anything new in the profession: every experienced architect, engineer and contractor is doing this implicitly, building their knowledge

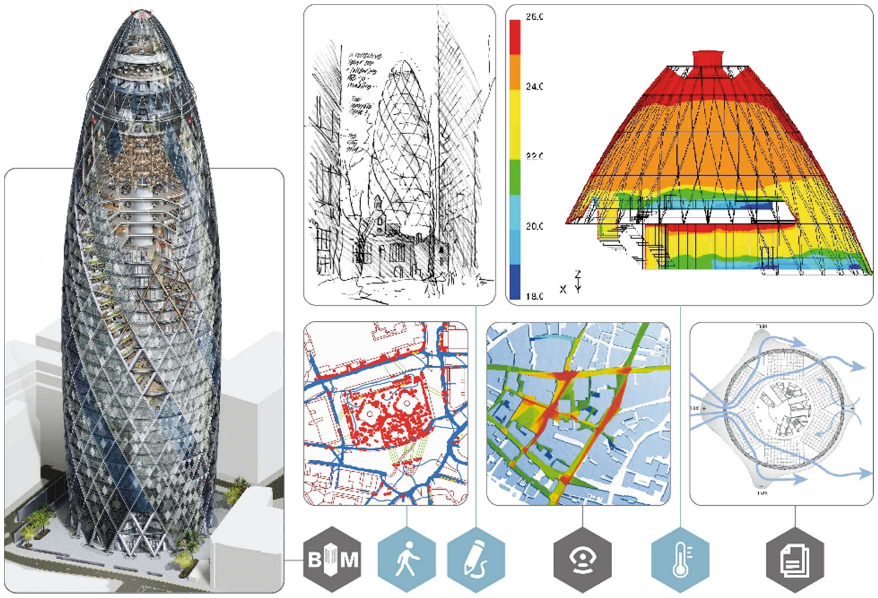

Fig. 1 Different types of *data associated with a building*

on top of the experience acquired on every project worked on. But this type of knowledge that lives in the head of every experienced professional is, unfortunately, not easily transferable.

So, the above beg the question: how can the industry run, manage and generally more effectively take advantage of the amounts of data each project yields in order to increase performance and efficiency in the design process? To that end, we can learn a lot by investigating how other industries deal with similar issues. The IT industry is a good example: their shift towards cloud computing to deal with big data has dramatically enhanced their processes both in terms of quality as well as efficiency, in the past years. Interestingly, when it comes to the creative industries, visualizers and animators are the most technologically advanced. That is a result of a close connection with the motion pictures industry and advertising. The internal competition for unusual visual effects and improved image quality has continuously driven software and pipeline development since the 70s and made them highly efficient.

Of course, AECO tools, processes and pipelines are different from those in IT or animation, not only because of their workflows, but also in that architectural computing is traditionally built around a single workstation. It is therefore interesting to first try and understand what is the software and hardware architecture that can allow us to run vast number of analyses in minimal timeframes or gather data from sensors, store that data in a meaningful and easily traversable manner and ultimately visualize and organize it.

To try and give some answers to the above questions, the problem will be first broken down to its fundamental parts: data, process, software and hardware. After its dissection the problem will be put back together and presented along an application which builds upon the understanding of how each piece works. More specifically, the nature of data (particularly within the AECO context) will be discussed and how it weaves in the design process and what are the capabilities and limitations of the current CAD software towards a more efficient data-driven design process. Then design workflows will be assessed against those in other industries, identifying relationships and parallels that our industry could take advantage of when dealing with big data issues. Various architectures for cloud computing will be presented along with their roles played in successful big data pipelines, and most recent software development practices for cloud environments will be discussed. Finally, everything will be brought together in a custom application called Hydra, that encapsulates all the above. Hydra is a distributed computing system that allows the user to vastly parallelize tasks for big design projects, dealing with gigabytes of data in a fraction of the time that traditional CAD software do and allowing the resulting data to drive project designs through an optimization pipeline. In addition, it is capable of classifying and visualizing its outcomes in a way that could allow us to use them in the future (e.g. as data sets to train machine learning systems). It is therefore a great example of how the authors are envisioning big data design computing to be incorporated in the design process.

2 Data in the AECO Industry

2.1 Nature of Data

The multidisciplinary nature of the AECO industry poses some interesting challenges in terms of not only the nature but also the sheer volume of the data that can be produced during the design and construction process. This data is everywhere: created, assembled and embedded in the design of the built environment from conception to operation and beyond. Sketches and drawings, simulation and analyses, simple or building information models (BIM), spatial and geographical data from geographic information systems (GIS), construction logistics and procurement, post-occupancy data gathered by sensors or 3D scans (see Fig. 2), HVAC systems monitoring: there is a multitude of processes, means and ways by which data can be gathered, analysed and effectively drive the built environment.

If a rough grouping of this data has to be provided, it could be done in accordance with the various stages in the life of a building: design, construction and operation. *Construction data* can incorporate anything, from procurement analytics to monitoring of as-designed against as-built models. *Operational data,* on the other hand, has become more and more important, as it allows the owner to assess an improved performance of a building daily. In this vein, a building is not much

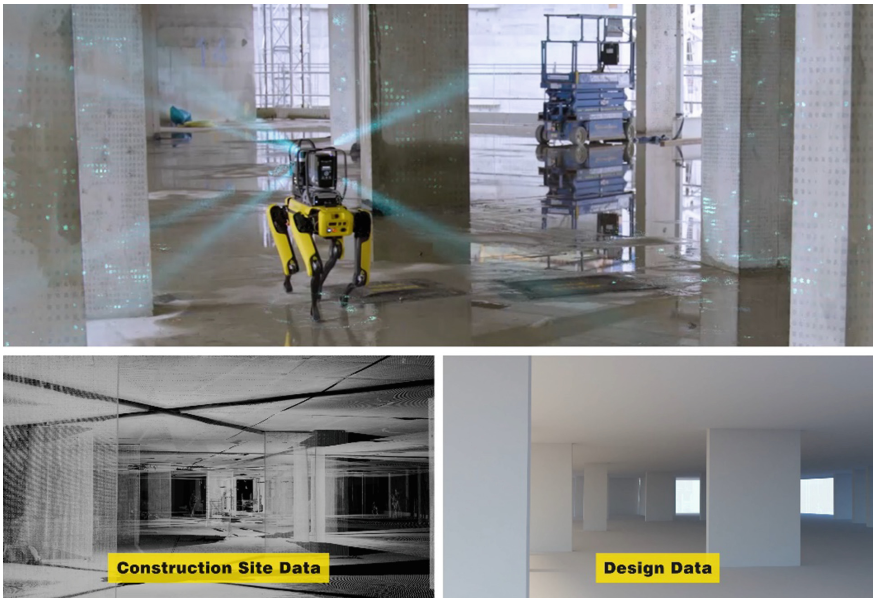

Fig. 2 3D Lidar data from a construction site gather*ed* by a scanner mounted on top of *Boston Dynamic's semi*-autonomous *Spot*

different to many businesses: its value is measured relatively to its ability to mine and use data. Therefore, the use of sensors and Internet of things (IoT) in conjunction to BIM and digital twin models has made the accessibility and management of data on operational building a very powerful tool towards their perceived success.

Although construction and operational data are very rich and interesting, it is important to also focus on the data produced during the design process. It is this performance-related data, produced during the design process, that is of interest to us, as the potential of optimization in project early phases is higher and the impacts of changes of the building and the construction costs are lower [5]. Design performance has long been recognized as an important issue in the built environment and has long been considered a seminal component in the value system of architectural design. A BIM model can be quite information-rich on its own right, but usually it is only the final step of a very nonlinear and convoluted design process.

2.2 Design Process

Most of the data produced during the design process is a by-product of complex and usually time-consuming simulations to assess objectives like structural viability, daylight, wind, views, connectivity, etc. All these analyses, albeit very different to one another, have some common characteristics: they require bespoke software, they are usually run as branched out processes rather than in tandem with the design progression, they are very time consuming and they provide data-rich results that require specialized knowledge in order to be incorporated into the design process. In addition, they may contradict each other, as they are often set to analyse conflicting objectives. The above have as a result a time-consuming and computationally intensive production of information that is difficult to incorporate as metadata within a design model.

Effectively, during the design stage, enormous amounts of data are produced whose results are stored in different models and delivered in such a compartmentalized fashion that it becomes hard to even meaningfully use them during very fast design cycles (not to mention storing them or even harnessing them for data mining purposes). This is a result of both the process and the software used to complete these tasks, which was for a long time reflecting the predigital workflow followed by the industry: that of the architect/decision-maker spending many lonely hours in front of the drafting board, while cross-discipline synchronization was left for later stages in the process. Technological advances and collaborative design platforms have made those lonely drafting days obsolete. However, those early-days CAD software architectures have had a ripple effect whose repercussions are still felt today. Namely, those platforms struggle to deal with the dramatic increase of speed at which new data can be produced and in the endless parade of non-interoperable data formats.

2.3 Design Software

CAD software is ever evolving, and each new generation promises new capabilities in relation not only of the creation and manipulation of geometry, but also of data (see Fig. 3). But to paraphrase Jane Austin, it is a truth universally acknowledged that a CAD software in possession of a good parametric platform, must be in want of simulation engines to interop with. If the replacement of hand drafting by CAD

Fig. 3 An evolution of design tools

software was the flint, then the rise of computational simulation engines was the steel that produced the spark which ignited the explosion of data in design workflows and the ultimate rise of performance-driven design.

Drafting CAD software, parametric and generative design platforms, BIM systems and ultimately algorithmic computational approaches are consequent and ever-evolving versions of design tools that strive to offer increased control not only of the design product, but also of the data associated to it (with promises of interoperability being made abundantly). This rise in control and connectivity is consequently—and unavoidably—directly related to a rise in the learning curve for these tools. Nevertheless, it is partly to their ability to offer these extra layers of control, that things like parametric and computational design have graduated from fringe design tools to mainstream software within the span of a decade. To take matters a bit further, the current implementation of BIM methodologies into various platforms has been shaping to be the ultimate (if occasionally glorified) database of the design industry: a data-rich 3D modeller, providing amazing drawing production capabilities albeit sometimes limited data-driven problem-solving agency to the designer.

Until recently, most of the above software had a common denominator: their limitation of the single workstation software architecture. In the past 15 years, there has been a shift in this paradigm with the use of central models in BIM, and with the development of new, usually programmable platforms (e.g. Rhino.Inside), that are trying to solve the interoperability issues the industry has long been burdened with. But apart from these, most classic CAD software were predominantly designed with the “one-computer-one-designer” approach in mind. This assumption, integral in the software architecture of these tools (and the overall design methodology), is a far cry from contemporary hardware/software paradigms, which are built to their core to deal with distributed computing problems (where one person can easily orchestrate hundreds of machines) and enhanced interoperability capabilities.

3 Big Data in the Cloud

3.1 What is Big Data?

According to NIST, big data refers to extensive data sets, whose characteristics are volume, variety, velocity and variability (also referred to as the “Vs” of big data) that require a scalable (software) architecture for efficient collection, storage, cleaning, processing and analysis [6]. Working with big data could not be pinpointed to a single framework or technology. Many tools have been developed to work together towards the main goal of discovering patterns and correlations in raw data and turning them into actionable insights. They belong to a large ecosystem collectively labelled as big data analytics.

The V's characteristics can centrally be delineated to the data produced by AECO. This data has huge potential, which could only be realized if only a step back is taken and examined how other industries have dealt with their data-related challenges. The broad definition of big data has two inherent characteristics relating to both its size and structural complexity. Data is classified as "Big" when traditional database approaches of storing, cataloguing and querying fail to process it—both in terms of data analysis as well as data visualization. These data sets are becoming the norm, as the rate of data generation has been growing drastically over the past decades. This is primarily credited to the extended use of smart mobile devices and the ease of access to the Internet. Back in 2017, it was estimated that by the beginning of 2020, the total amount of digitally stored data in the world would be around 44 zettabytes [7]—a number which is almost impossible to comprehend and yet could be already rendered obsolete due to the exponential rate of data production. For comparison, the total compressed size of articles in the English version of Wikipedia is 18.9 Gigabytes. This means that there could be roughly 2328 billion Wikipedias worth of data out there [8]—and growing!

Most of this data is unstructured, meaning that there is no predefined data model or schema by which this data is being created, hence maintaining and processing it is far from straightforward. This type of data, (e.g. point clouds from Lidar scans or photogrammetry data) could be saved in data lakes which are systems designed to store it in its natural format such as Google Cloud Storage [9], Amazon S3 [10] and Azure Data Lake [11]. On the other hand, structured or semi-structured data adheres to specific formatting that can be relatively easier to query and evaluate through relational databases in data warehouses (e.g. Oracle [12], Amazon Redshift [13], Azure Synapse [14]). Data fitting in this category can be for instance collected from Internet of things (IoT) sources in a building through its lifecycle, as these usual correspond to specific formatting as provided by the fabricator of the sensory systems. Another metric in combination with the other Vs [15] that plays a detrimental role in the meaningful analysis of data is the "veracity", which relates both to the quality of the stored data but more importantly on its accuracy [16] and the credibility of the generating source.

We have therefore two major issues: quantity and traversability of data, both of which started booming at the beginning of the new millennium. These forced major IT players and tech companies to present efficient ways of tackling problems of storage and data processing power, in order to deal with the exponential increase of data coming their way. Hardwarewise, there was simply too much data to fit even the largest hard drive on a single machine. Additionally, the pace at which data was produced was much faster than the growth of hard-drives' sizes, so information had to be divided in chunks and placed on multiple machines. This solved the storage problem but automatically created maintenance issues. Eventually, keeping track of where relevant chunks were physically located (the computers could be in different geographical locations) and what was stored and processed became a major issue. Depending on the size and turnaround of new data batches, there are two common approaches. One is called *batch processing* and deals with large data chunks when slower turnaround between collection and analysis is acceptable. On the other hand, if much faster decision-making is needed, then *stream processing*, which is more complex, could work with smaller data blocks.

For example, to meet their own rapidly growing needs and based on some earlier work, Google developed in 2013 the Google File System (GFS) [17]. The system automatically divided their machines into storage and server nodes. Raw data was split into chunks and placed on storage nodes, while the server nodes were only responsible for keeping track of metadata. Additionally, they decided to use low-spec machines, which despite having a higher failure rate, were much cheaper to replace. Therefore, it was easier and more economically viable to make backup copies of the data on many redundant machines (to cope with the percentage of machines they anticipated would fail), rather than to invest in higher-end hardware. This was followed by a distributed storage system also designed specifically to store Google's Big Data from applications such as Google Earth and Google Analytics, focusing on latency-sensitive servicing called BigTable [18]. The above two systems immediately demonstrated the necessity of distributed or cloud-based solutions to deal with the new normal regarding the pace we generate, store and analyse data.

Data acquisition is usually a unique process for every organization and is tightly related to the nature of data they produce or use. Raw data is usually incomplete and could produce incorrect insight. Therefore, before drawing any conclusions, it always requires some preprocessing which reshapes raw data into an understandable format. That is where *big data analytics* come to play. Steps must be taken to eliminate duplicates, handle inconsistencies, integrate information coming from different sources or annotate and assign labels. When there is too much data, it could be discretized or sampled. Once data is sanitized and formatted, it could finally be analysed using both analytical and statistical tools to uncover useful information.

Historical data could be used to make predictions and forecast future behaviour using predictive analysis. Anomalies and data clusters can be identified by data mining tools. The abundance of data also gave rise to more efficient ways of mining data by utilizing machine learning (and even more advanced deep learning) methods to derive patterns in data sets [19]—a practice in which companies like Google saw enormous potential. Leveraging the power of collated data is what made Alphabet Inc., the multinational conglomerate it is today. In retrospect, it comes as no surprise that Google provided its users almost unlimited storage to host their e-mails or imagery very early during this whole endeavour. In 2004, in their new e-mail client—Gmail—they offered 1 GB of free data storage, almost 500 times more than similar vendors offered at the time, as they have foreseen the dominance of big data and their key role in predicting trends based on the user's decision-making process. In a similar fashion, Facebook, Twitter, Amazon and even Netflix have chosen their own platform-specific big data analytics [20] trying to understand the landscape of their user base, while improving their services with more relevant content.

At the time of writing this chapter, the AECO industry is still trying to catch up with this big data frenzy, not only because of the difficulty of collecting and organizing its data in a useful way, but also due to the nature of the AECO data itself.

3.2 Cloud Computing

As previously mentioned, to mine, analyse and meaningfully visualize big data we need great compute power, which can be found amply in cloud resources. cloud computing (CC) describes the process by which users may store data and applications on a remote cloud location (in contrast to on-premise or on-prem) and then traverse them in an effective way [21]. CC is intertwined with the existence of data centres, as it still requires a physical location for storing the data and performing computational processes on/with it. A data centre can be defined as an actual physical premise that organizations such as Google or Amazon use to store their data and to run their applications.

The nature of data centres has shifted from the initial on-prem resource model—where individual organizations had to provide a physical space to host computing hardware, storage devices and a robust networking infrastructure—to completely off-site ones [22]. The idea of those decentralized locations is to provide cheap computing and storage resources *as a service* which could seamlessly integrate with the clients' basic infrastructure via fast Internet connections, as if all resources were on site. Furthermore, they can be easily accessed by virtualization technology, where an image of the behaviour of a typical computer is used to run applications or perform tasks like a physical computer would normally do. These virtual machines (VMs) are essentially applications that act as computers and are easily manageable, widely available and capable of running different operating systems on environments from the same physical endpoint, for example our personal computer at home [23]. This gives the users the ability to take advantage of massive compute and storage resources (and be charged on a per-use basis) without being responsible for their maintenance. It is worth noting that cloud computing, where resources such as compute power are available via a service system, differs from systems such as grid computing, where machines are connected in a grid structure and try to perform a coordinated goal by accessing resources directly [24].

Cloud computing vendors provide different services, but in principle these usually come under three distinct classifications: infrastructure as a service, platform as a service and software as a service (see Fig. 4).

Infrastructure as a service (IaaS) is a very common model which provides the whole hardware infrastructure to users without any further action required for maintenance, back up or recovery [25]. However, the responsibility for the software applications or the choice of platform lies with the user. In addition, the system provides a very detailed model for monitoring, load balancing and billing.

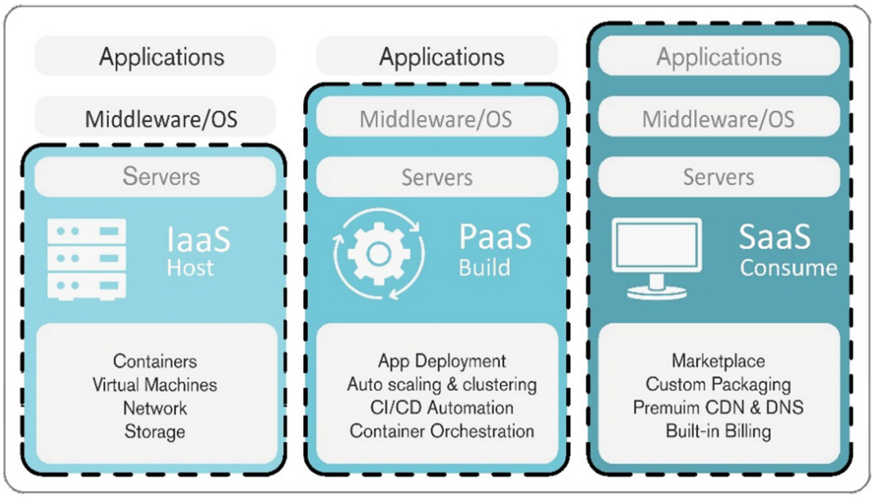

Fig. 4 Diagram showing the three basic categories of services offered by cloud computing providers—IaaS, PaaS and SaaS

Platform as a service model (PaaS) offers the ability to build, test, deploy and maintain applications by providing all the relevant hardware and software infrastructure, such as runtime, storage and queuing but the configuration of the system lies with the user so it can be tailored to specific needs. This model favours not only the users who can run their services or applications without managing an actual network, but also the cloud infrastructure providers, as they can have many diverse applications and systems running on their platform [26].

Lastly, the software-as-a-service (SaaS) model can also be described as a licensing model. It contributed greatly to the explosion of cloud computing as it removes the need of investing on locally based compute or storage resources to install and run a specific software application. With SaaS, part of the software, such a lightweight UI system, can reside on-prem, however all the major functionality happens on the cloud, while enabling collaboration between many users or machines.

3.3 *Microservices*

This tremendous increase in compute power available on demand (feasible and directly dependent on Internet connection speeds) allowed millions of machines and even more applications to be connected to the Internet, all of them ready to process user requests or talk to each other while performing various tasks. This created massive coordination issues. Traditional applications were developed in a monolithic way (see Fig. 5). They were self-contained with each part tightly

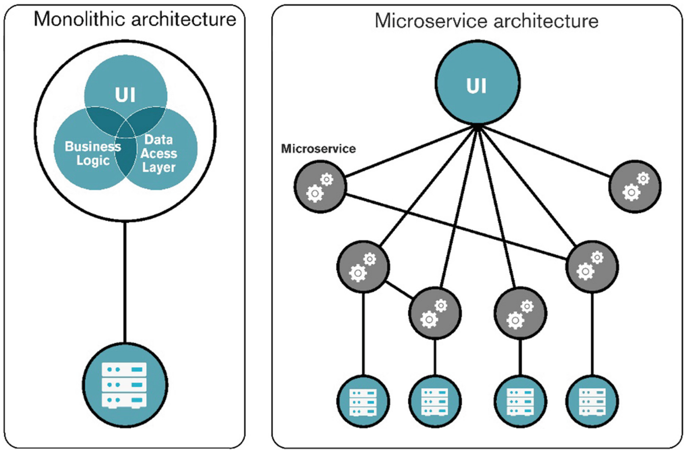

Fig. 5 Monolithic versus microservice-based software architecture

interconnected and interdependent. This turned out to be highly inefficient because scaling-up such systems to accommodate the servicing of more users (and data) required the scale-up of the entire application, instead of just scaling individual function which were experiencing surges in demand. Additionally, it also turned out to be incredibly difficult to make small updates without having to rewrite the entire application.

In time, developers started breaking the monoliths into small, independent, autonomous and loosely connected parts called microservices (see Fig. 5) [27]. In short, microservices can be described as the approach where a fleet of smaller independent processes communicate with each other via a lightweight data and information exchange mechanism, working together to form a bigger application and achieve a single goal [28]. The microservice approach has many significant advantages both in terms of the flexibility of development of the applications themselves, as well as its scalability, as each individual service can be executed on many machines at the same time. The additional benefit is that, similarly to Google's GFS examined previously, multiple instances of each service allow some of them to fail safely, without bringing down the whole system. On the other hand, they are a bit more complex to structure than simple monolithic software applications, as the different services need to talk to each other efficiently. For instance, a case of deficiency would be if the communication mechanism of one of the microservices is changed in such a way that is not backwards compatible to other services talking to it. The lightweight information exchange mechanism which glues all those microservices together is called application programming interface (API). APIs are based on predefined set of rules which allow certain actions to be taken between individual services. The most popular standard of APIs for building microservices is known as representation state transfer (REST) as introduced by Roy Fielding in 2000 [29]. The application of REST services is particularly successful because only a representation of the *state* of the resource accessed by the REST-based API is returned to a client—which can be either a person or a program. This state is usually in a light data format such as JSON or XML and its independent of the actual details of the implementation of the resource. For example, if we need to access some entries in an e-mail address book, we do not care on how the contact card is implemented, but only about the name and the e-mail of the entry that we are looking for.

3.4 Cloud-Based Software Development

Creating efficient cloud-based applications as well as deploying and maintaining thousands of their instances simultaneously have led to the development of new coding standards and practices (see Fig. 6). The traditional monolithic applications were developed in a highly sequential way and split into distinct phases. These phases had a strict linear sequence, like a water cascading down a hill, christening this type of systems architecture the *waterfall* approach. On the other hand, a highly dynamic nature of the cloud environments, allowed for a more flexible and adaptable approach. Software requirements could change overnight, often rendering the careful planning, so valued by the waterfall approach, entirely outdated. The ability to be *agile*—respond quickly to requests with targeted, small updates within days rather than waiting for the entire waterfall to flow—proved to be more efficient (Facebook, for example, is making updates to its production code every day [30]).

The necessity of reducing delivery times while still providing high-quality products has led to increased automation and coordination of both the software development side (Dev) and the IT operations side (Ops) of the process.

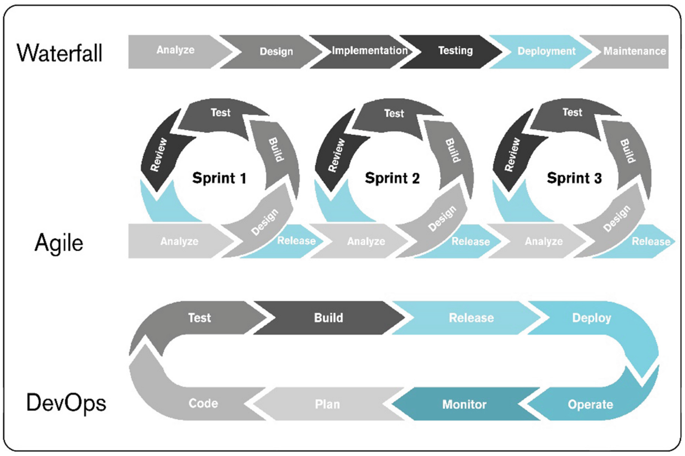

Fig. 6 Comparison between three most common software development methodologies

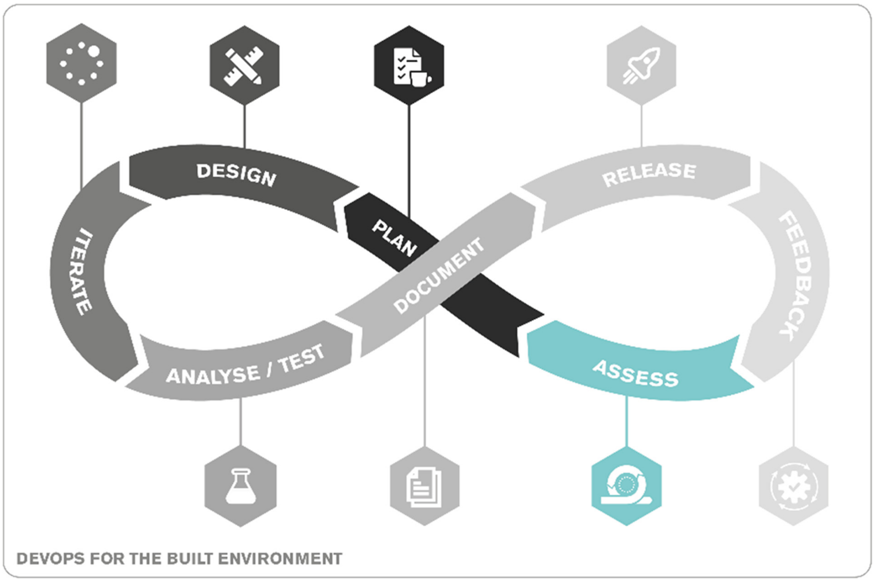

Fig. 7 Conceptual design pipeline based on to the DevOps methodology

The resulting methodology called *DevOps* is not a rival methodology to agile. Its aim is to compliment agile's costumer–client coordination and expand it to a seamless communication between the developers and the IT infrastructure. The primary goal of DevOps is automation. Especially when it comes to limiting potential failures, minimizing bottlenecks and ensuring maximum performance when deploying the solutions. It is comprised by a series of key components, namely continuous integration (CI), continuous delivery (CD) and continuous deployment. While CI is entirely focused to the development of code and its regular testing and merging to the main software or application, the other two components are sometimes interchangeable terms. Their key difference in a simplified way, is that in continuous delivery the continuous updates delivered to the user are manually triggered by the development team whereas in continuous development, the whole pipeline is completely automated. A similar development practice can be a potential candidate for the build environment from the planning, engineering to the documentation aspect of it where iterations are delivered to the relevant stakeholders (see Fig. 7).

3.5 Cloud Computing in AECO

As designers we can learn from the workflows of the IT and tech industry, as they may differ in size, but draw some interesting parallels in the principles of continuous deployment in delivering solutions, where the DevOps CD component comes into play. Design teams are always required to frequently deliver iterations of their designs to the respective stakeholders, whether these are internal reviews, the client, building authorities or consultants. This involves a very time-consuming iterative process associated with the design cycle and its derivatives—such as documentation, costing or visualization.

Imagine now, employing discrete design workflows or procedures as microservices in the design delivery process. This could work like this: whenever a new design iteration is created, a series of automated pipelines could immediately check for building regulation compliance, cost or performance-driven metrics. In parallel, the system could fire-off a rendering pipeline to create visuals of preselected views to be audited by the designers. Then, the design could be approved through a sequence of design "pull requests", tailored to each organization and its existing workflows. After the approval stage, the new design option could be automatically delivered to the relevant stakeholders. The system would be able to run and assess various options in parallel, while simultaneously storing all the derivative data in a structured way, so it could be accessed on demand, or even used as a training set for predictive models.

Although the above is still a remote target, the AECO industry has slowly started implementing some of these concepts within its workflows. CAD software vendors such McNeel and Associates have moved part of their applications (Rhinoceros3d [31]) towards a SaaS model, where users are able to access instances of the software over a cloud and perform all the computation there. This opened more possibilities for tech companies stemming from the AECO industry to perform their computation on VMs. Such examples include Hypar.IO [32], a cloud-based CAD service deploying both Rhino.Compute [33] and its own bespoke functions and Thornton Tomasetti's Swarm [34], that works purely with Rhino.Compute instances running parametric definitions (Grasshopper3d) on the cloud. A similar approach which lies closer to the microservice architecture is adopted by the developers of Ladybug tools [35] for environmental analysis in their newest cloud-based platform called Pollination. In their latest iteration, all the computation happens on cloud servers in the form of small distinct functions that are parallelizable. In this way, failures occurring while the processes are running can be easily traced back and rescheduled without having to rerun the entire analysis from scratch.

On the other hand, there is a current trend for applications talking in between software such as Speckle [36] and BHoM [37], which is open source and implement the principles of RESTful APIs communicating stateless messages through servers. Continuing the story on trends, start-ups and spin offs like Spacemaker.io [38], Giraffe [39], TestFit.io [40]. Archistar.ai [41] and Google Alphabet's SidewalkLabs with project Delve [42] are developing applications that run on web browsers and deal mainly with design space exploration, harnessing the power of data mining and predictive modelling through machine learning. The important thing here is that these systems are built in such a way so that the computation is independent of software solutions and can be accessed by any device capable of connecting to the Internet. This concept of central consolidated models over clouds is one of the basic functionalities of Autodesk's BIM360, and the notion of distributing tasks has been used by designers, even unknowingly, whenever they are rendering imagery over render farms.

4 Harnessing the Power of Distributed Computing and Big Data in Performance-Driven Design

4.1 Performance-Driven Design

Applying cloud-based workflows in AECO and harnessing the power of its data could become a game changer in the way architects work and the quality of buildings they produce. But as mentioned earlier, the focus will be predominantly on workflows and data produced during early stages of the design process, as this can have a cascading effect in further stages of the life cycle of a building. Furthermore, the importance of early stage design has also to do with the plethora of possible solutions (and its consecutive data) that can be explored during the relative freedom that the conceptual stages of a project can offer.

In traditional design processes, early stage decisions are predominantly based on the architects' experience and their ability to make educated guesses about the repercussions of their design decisions. While those early decisions are crucial to the success of the project [43], it may be increasingly difficult to manually navigate through the multiple criteria that drive them, particularly as the projects scale up in size and complexity. However, there exists a feasible set of design options which can extend beyond the limited manually produced and tested ones and which could satisfy basic brief requirements while having vastly different performance values. This set of possible options can be easily visualized as a physical landscape with hills and valleys. In this analogy, the basic building features (like its orientation or height) could be compared to the x, y components of geographic coordinates, with their z value representing the performance score of each design option. Of course, this is an oversimplification, as these landscapes of solutions spaces are usually multidimensional, but the analogy is still applicable. When a project starts, this landscape is usually an uncharted territory covered in dense fog. However, based on the architect's prior knowledge and experience, some promising locations of the summits could be (sometimes almost unconsciously) identified. Thus, one of the primary goals of the design process is to uncover and map this landscape, while finding the best trade-offs between quantifiable, functional and aesthetic requirements.

The exploration of the aforementioned landscape (and its potential for data mining) can be assisted by fast, efficient and automated methods for creating and analysing options in order to drive the solution to the right direction early on in the design process.

4.2 A Multitude of (Meaningful) Design Options

The most widespread method for automatic generation of design options is currently based on parametric and generative modelling environments like Grasshopper and Dynamo. The process of building such models requires a lot of expertise and domain knowledge. In every model, both requirements and hard constrains from the project's brief must be combined with a set of chained geometric rules. The rules reflect a certain design intent which is then translated into a piece of software capable of generating variations of building geometry in respect to a given context. Every project has unique characteristics, thus requiring a bespoke approach. Additionally, during early stages, a design direction changes very often, sometimes even overnight. Thus, many different typologies, derived from geometrically different rule sets, need to be tested. The rules usually have nothing in common and need custom development, resulting in multiple models per project. Therefore, capturing a design intent becomes a challenge in all building scales, but particularly for urban scale master plans, where many different building typologies need to be designed and tested.

Striking a balance between flowing brief constrains and geometrical rules is more art than science when handcrafting a model definition. Well-structured generative models should output design option with enough complexity and variety to make the exploration interesting while still complying with building code regulations and a desired aesthetic intent. For urban scale projects, this rule-based approach is often the backbone of procedural city generators (e.g. City Engine [44]) dedicated to large-scale, high-level 3D city modelling. This technique is used heavily in the gaming and visualization industries to produce vast landscapes and cityscapes. Procedural modelling can produce more diverse and detailed results with less effort and in a shorter period of time. Nevertheless, for these options to be meaningful and compliant to project requirements, an extensive amount of rigour needs to be applied to the definition of the rules applied.

Although handcrafting rule-based generative models, to capture the complexity of urban fabric, is considered state of the art, it is worth noting that the tech industry is, based on recent advancements in artificial intelligence and machine learning, investing a lot into systems which could create such models in other domains automatically. Those advancements have been possible because of the access to vast amounts of structured data and are discussed in detail in Chapter "Artificial Intelligence for the Built Environment" on machine learning.

4.3 Performance Analysis

Earlier in this chapter, we have discussed the principles of performance-driven design and their importance in the design process. Driving a design solution based on performance, means that each option needs to be checked against a series of analytical factors that define its fitness. So, moving on from creation to analysis, each of the hundreds of options created based on the generative tools presented above, needs to be assessed against a set of criteria representing the project's performance indicators.

These indicators cover a vast array of analyses, from simple gross floor area calculations to environmental analysis (daylight potential, total annual solar radiation, number of sunlight hours, wind flow, etc.), sustainability (e.g. embodied and operation carbon footprint), structural stability, intervisual and special connectivity, financial models, perceptual or well-being such as quality of view (e.g. visibility of certain context features like sea or green spaces) and many more.

Running each of those analyses require dedicated software and hardware (e.g. high-end graphic cards) as well as considerable compute time. In most cases, basic geometric data, which represent a mass of a single design option to be analysed, must be preprocessed (sometimes tediously and manually) and converted into a specific format before starting a simulation. Additionally, analysis engines usually produce vast amounts of temporary data which needs storing and is required to produce aggregated scores. Calculation times for each analysis could vary from minutes to hours which makes a massive difference when a big number of design option needs to be evaluated. This poses a great challenge both in terms of investment in the infrastructure which could support that many different engines and shear compute time. Many of those issues could be effectively mitigated, if both cloud resources and parallel computing are used.

4.4 Big Data Challenges When Searching for Good Solutions

Exploring a solution space defined by a generative model against a set of key performance indicators is a non-trivial problem. Finding a relationship between the vast input combinations of the generative model and their respective effect in the solution's performance against all set objectives is a task whose complexity increases exponentially along with the number of inputs and analyses required. One could try to make observations regarding input–output correlations, by manually manipulating input values while observing the resulting calculating performance, but this is not efficient, and it certainly is not scalable.

With the limited time that fast design cycles provide, an automatic search mechanism is required. Nature has already created one of the most effective search and optimization mechanism—biological evolution. Evolution promotes organisms' adaptation to their natural environment which is usually described in literature as a combination of variation, heredity and selection [45]. Variation is realized by the existence of multiple individuals at any given time in a population. Heredity allows to pass down certain traits from parents to children in discrete chunks iteratively from one generation to another working as a form of cache of good traits.

Competition for natural resources creates a selection pressure which drives the entire phenomenon. Biological evolution works on large timescales, but its basic principles can be computationally replicated. From the computational perspective, evolution could be seen as a distributed and parallel mechanism for constantly creating organisms which are better set to adapt and survive in an ever-changing natural environment [46].

Appling the evolutionary principle to the problem of finding an optimal building massing expressed as a combination of input parameters (like building orientation or height) driven by a single objective (e.g. daylight potential) is quite straightforward. The single performance score can be directly used to drive selection pressure as a fitness value. As long as the fitness pressure exists, new solutions bred (mainly but not solely) from the best performing individuals of previous ones will have good chance of being better than their parents. This process becomes significantly more difficult when more than one objective is being taken into consideration, which is usually the case in any real-world scenario. There are various strategies to deal with this issue but the most popular and widely accepted is to keep selecting solutions where you cannot improve one objective without deteriorating the performance of any other. A set of such solutions is called non-dominated or Pareto-optimal after Vilfredo Pareto, an Italian economist. Finding and being able to traverse this set by decision-makers is the main goal of multiobjective optimization and multicriterion decision-making [47, 48]. Multiobjective optimization is a mature field in computer science with many real-world application [47, 49]. On the other hand, a high number of objectives could negatively influence the overall performance of an optimization study. More data to process not only increases computational cost but also might affect the stability of the whole simulated evolution [50].

Despite almost 30 years of research on this topic, the applications in AECO industry have been mostly academic or focused on simple or single domain problems due to significant computational cost. As described, in the previous paragraphs, architectural software is mostly outdated and does not fully take advantage of recent advances in cloud computing. Performance-oriented design on large-scale, real-world projects which could benefit from evaluating multiple performance criteria is challenging. To put into perspective, using current tools a generation of a single option for a large masterplan project and its subsequent performance evaluation could take approximately 15 min [51]. This might not seem long, but a single optimization study would need to process at least 100 generations each with 100 options processing 10,000 solutions in total. Multiplying each option by 15 min results in staggering 150,00 min (almost 104 days)! At an early stage where a design intent changes weakly or even daily this timeframe would not be acceptable. By the time the results are ready, the design would have moved on to a point where the outcome of the study would be of no use to a design team. The problem with this approach is that all work is done sequentially by a single machine. It is a significant bottleneck, as it limits the applicability of this method to simple or single domain studies, excluding most of the real-world problems. As described in the previous paragraph, individual solutions which exist in each population are completely independent from each other. Thus, they could be processed on different machines in parallel, dramatically reducing compute time even up to a single day.

5 Hydra Tool

Hydra is an example of a bespoke tool developed at Foster + Partners by the Applied Research + Development group, which aims at addressing the aforementioned issues and tap on to the potential of big data and cloud computing [51]. In its basic principle, it takes large geometry generation and simulation tasks and splits them into smaller parts which are automatically executed in parallel on multiple machines using an on-prem cloud. Following the previously discussed software developed patterns, Hydra's components are developed in the agile way and broken down into microservices which communicate using a custom REST API (see Fig. 8). Similar to Google's GFS system, it uses two high-level machine types: the first schedules and monitors execution of subtasks or controls the optimization process; the second picks pending subtasks from a queue and runs them simultaneously. The tasks range from creation of geometry using generative models or procedural city generators, various performance analysis and both image and video rendering. This software pattern, known as the controller–worker model, is extremely efficient, especially for evolutionary-based optimization studies [52, 53].

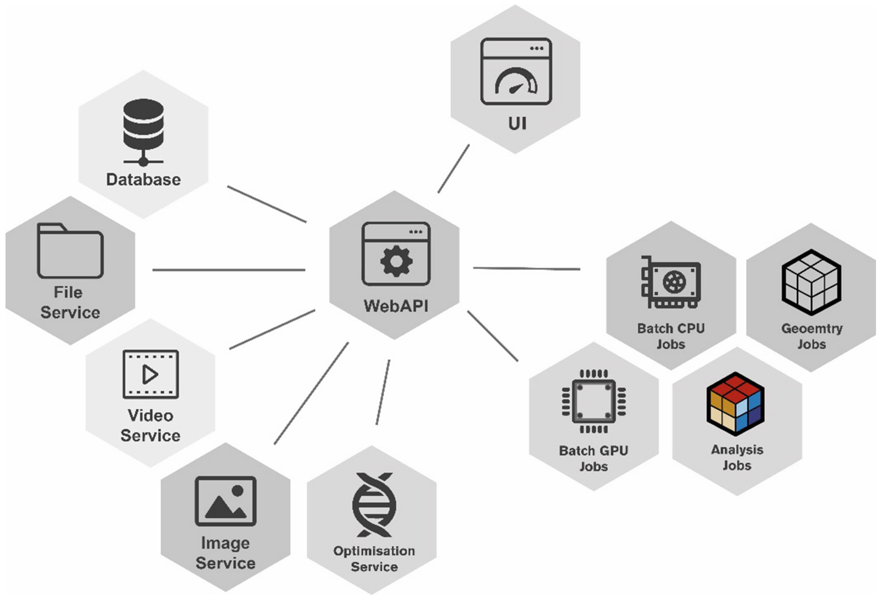

Fig. 8 Diagram showing Hydra's microservice-based software architecture

Metadata associated with design solutions generated by Hydra is stored in an organized database which could be queried by designers. The database also stores pointers to temporary geometries and images generated during the execution of the sub-task, which could be pulled and reviewed on demand. Tens of thousands of solutions generated by Hydra could be visualized in an interactive web-based dashboard which helps with exploring and understanding the relations between various design parameters and trade-offs between design objectives.

Hydra's development has been driven by an extensive use in real-world projects in the practice. It has been used on 15 real-world projects for the past 2 years, especially large-scale masterplans. One of such examples is the project which won the design competition for Guangming Hub, a new transport-oriented development situated on the high-speed rail link that connects Hong Kong, Shenzhen and Guangzhou. Based on the brief, an integrated team of architects and specialists identified a set of key objectives and constraints which would drive the initial placement of urban massing to positively influence urban growth in the region. The challenge was to find an optimal combination of buildings based on site-specific design typologies, while maximizing daylight potential, cutting down average walking time to a high-speed rail station, ensuring that as many buildings as possible have a view to the central park while keeping the 2 M sqm GFA target (see Fig. 9).

Base on the project brief, a parametric model was developed and fed along with the constraints to Hydra, which processed 10,000 options in total in 49.14 h providing approximately 37 times speed up (each option would take otherwise on average 11 min to process). The final Pareto-front of 100 options with the best trade-offs between the objectives was then uploaded into the web-based interface (see Fig. 10) and discussed with the design team. After choosing the most

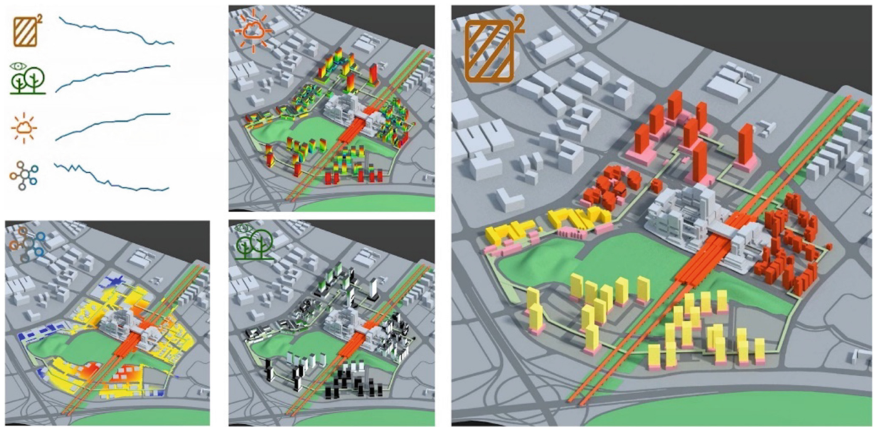

Fig. 9 A single design option from the Guangming Hub project analysed against four performance criteria

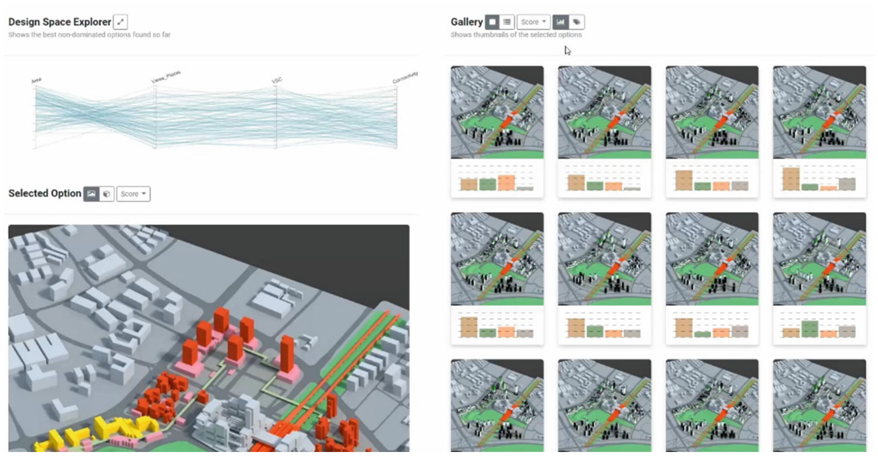
Design Space Explorer
Shows the best non-dominated options found so far
Area
Views_Plazas
VSC
Connectivity
Selected Option
Score
Gallery
Score
Shows thumbnails of the selected options

Fig. 10 A screenshot of Hydra's web-based UI showing the interactive dashboard which allows easy exploration of the project's solution space

promising solution, a Hydra-generated massing was used as an inspiration and starting point for the team's design explorations towards the production of the final model.

6 Conclusions

As demonstrated in the Hydra-based case study, the current advancements in cloud computing could dramatically reduce computing time for performance analysis simulations and complex multiobjective optimization studies. By cutting down the processing time from months to days or even hours, those tools could be effectively used in the context of early stage design. This in turn, gives architects and all stakeholders involved in the process, a powerful tool which allows to make more informed decisions and significantly influence the design at its most critical stage. It also allows for the immense amounts of data produced to be collected, processed, structured and classified in a meaningful manner, ready for further reuse, e.g. in machine learning pipelines.

This change has been possible because current cloud computing technology provides access to massive compute clusters and data storage for little money as a service without a need to invest in expensive hardware and maintenance. This, as we showcased, led to an explosion of cloud-based distributed applications and facilitated the creation of new software development standards. The easiness with which it is possible to develop such applications and whole systems (as demonstrated in the Hydra example) gives architects a new perspective and an opportunity to rethink current design pipelines and workflows.

Additionally, data which is currently being created by the AECO industry contains a great amount of unstructured information such as CAD models, images or written documents, which makes it inherently difficult to query or effectively analyse. However, the increasing input of technology within the building industry profession and in combination with more schema-specific data from IoT devices and smart buildings can potentially lead to a point where data coming out of the design board or collected from buildings will be suitable for analysis and further processing. On the other hand, more compute power available on demand and vast storage capacity could facilitate the creation of huge and more structured data sets. Such sets are critical for building data-based predictive which could lead to a next generation of more sophisticated design tools.

References

1. Bilal, M., Oyedele, L.O., Qadir, J., et al.: Big data in the construction industry: a review of present status, opportunities, and future trends. Adv. Eng. Inform. **30**, 500–521 (2016). https://doi.org/10.1016/j.aei.2016.07.001
2. Loyola, M.: Big data in building design: a review. J. Inf. Technol. Constr. **23**, 259–284 (2018)
3. Tsigkari, M., Chronis, A., Conrad Joyce, S., et al.: Integrated design in the simulation process. Proc. Symp. Simul. Archit. Urban Des. **28**, 1–28:8 (2013)
4. Chronis, A., Tsigkari, M., Giouvanos, E., et al.: Performance driven design and simulation interfaces: a multi-objective parametric optimization process. Proc. SimAUD SCS SpringSim' **12**, 81–88 (2012)
5. Bragança, L., Vieira, S.M., Andrade, J.B.: Early stage design decisions: the way to achieve sustainable buildings at lower costs. Sci. World J. **2014**, 1–8 (2014). https://doi.org/10.1155/2014/365364
6. NIST Big Data Public Working Group.: NIST Big Data Interoperability Framework: vol. 1, Definitions. Gaithersburg, MD (2015)
7. Sh. Hajirahimova, M., Aliyeva, A.S.: About big data measurement methodologies and indicators. Int. J. Mod. Educ. Comput. Sci. **9**, 1–9 (2017). https://doi.org/10.5815/ijmecs.2017.10.01
8. Wikipedia. Wikipedia:Size_of_Wikipedia 20/11/20 (2020). Accessed 20 Nov 2020
9. Google Storage Cloud. https://cloud.google.com/storage. Accessed 1 Feb 2021
10. Amazon Simple Storage Service. https://aws.amazon.com/s3/. Accessed 1 Jan 2021
11. Azure Data Lake. https://azure.microsoft.com/en-gb/solutions/data-lake/. Accessed 1 Jan 2021
12. Oracle Database. https://www.oracle.com/uk/database/. Accessed 1 Jan 2021
13. Amazon Redshift. https://aws.amazon.com/redshift/. Accessed 1 Jan 2021
14. Azure Synapse. https://azure.microsoft.com/en-gb/services/synapse-analytics/. Accessed 1 Jan 2021
15. Zikopoulos, P., Eaton, C., IBM.: Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, 1st edn. McGraw-Hill Osborne Media (2011)
16. Lukoianova, T., Rubin, V.L.: Veracity roadmap: is big data objective, truthful and credible? Adv. Classif. Res. Online **24**, 4 (2014). https://doi.org/10.7152/acro.v24i1.14671
17. Ghemawat, S., Gobioff, H., Leung, S.-T.: The Google file system. SIGOPS Oper. Syst. Rev. **37**, 29–43 (2003). https://doi.org/10.1145/1165389.945450
18. Chang, F., Dean, J., Ghemawat, S., et al.: BigTable: a distributed storage system for structured data. In: OSDI 2006—7th USENIX Symposium Operating Systems Design and Implementation, pp. 205–218 (2006)

19. Witten, I.H., Frank, E., Hall, M.A.: References. In: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn, pp. 587–605. Elsevier, Boston (2011)
20. Sivarajah, U., Kamal, M.M., Irani, Z., Weerakkody, V.: Critical analysis of big data challenges and analytical methods. J. Bus. Res. **70**, 263–286 (2017). https://doi.org/10.1016/j.jbusres.2016.08.001
21. Wang, L., Von Laszewski, G., Younge, A., et al.: Cloud computing: a perspective study. New Gener. Comput. **28**, 137–146 (2010). https://doi.org/10.1007/s00354-008-0081-5
22. Bilal, K., Khan, S.U., Kolodziej, J., et al.: A comparative study of data center network architectures. In: Proceedings—26th European Conference on Modelelling and Simulation, ECMS 2012 (2012). https://doi.org/10.7148/2012-0526-0532
23. Smith, J.E., Nair, R.: The architecture of virtual machines. Computer (Long Beach Calif) **38**, 32–38 (2005). https://doi.org/10.1109/MC.2005.173
24. Fox, G., Gannon, D., Thomas, M.: Editorial: a summary of grid computing environments. Concurr. Comput. Pract. Exp. **14**, 1035–1044 (2002). https://doi.org/10.1002/cpe.734
25. Shahzadi, S., Iqbal, M., Qayyum, Z.U., Dagiuklas, T.: Infrastructure as a service (IaaS): a comparative performance analysis of open-source cloud platforms. In: IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD 2017-June (2017). https://doi.org/10.1109/CAMAD.2017.8031522
26. Keller, E., Rexford, J.: The "Platform as a Service"; Model for Networking (2010)
27. Gribaudo, M., Iacono, M., Manini, D.: Performance evaluation of replication policies in microservice based architectures. Electron Notes Theor. Comput. Sci. **337**, 45–65 (2018). https://doi.org/10.1016/j.entcs.2018.03.033
28. Garriga, M.: Towards a Taxonomy of Microservices Architectures, pp. 203–218 (2018)
29. Fielding, R.T.: Architectural Styles and the Design of Network-Based Software Architecturese. University of California, Irvine (2000)
30. Facebook. Rapid release at massive scale. In: Facebook English (2017). https://engineering.fb.com/2017/08/31/web/rapid-release-at-massive-scale/#:~:text=Althoughwepushtoproduction,orsoAndroidbetatesters. Accessed 1 Dec 2020
31. Rhinoceros3d. McNeel Association. https://www.rhino3d.com (2020). Accessed 20 Nov 2020
32. Hypar.io. Hypar. https://hypar.io (2020). Accessed 20 Nov 2020
33. Rhino.Compute. McNeel Association. https://developer.rhino3d.com/guides/#compute (2020). Accessed 20 Nov 2020
34. Swarm. Thort. Tomasseti Core. http://core.thorntontomasetti.com/announcing-swarm/. Accessed 20 Nov 2020
35. Roudsari, M.S., Mackey, C.: Pollination. https://pollination.cloud/ (2020). Accessed 20 Nov 2020
36. Matteo, S.D.C.: Speckle. https://speckle.systems (2015). Accessed 20 Nov 2020
37. The buildings and habitats object model. In: Buro Happold. https://bhom.xyz/. Accessed 20 Nov 2020
38. Spacemaker.ai. Spacemaker. https://www.spacemakerai.com. Accessed 20 Nov 2020
39. Giraffe. https://www.giraffe.build/ (2020). Accessed 20 Nov 2020
40. Harness, C.J.: TestFit.io. https://testfit.io/ (2020). Accessed 20 Nov 2020
41. Coorey, B.: Archistar. https://archistar.ai/. Accessed 20 Nov 2020
42. Sidewalklabs. Delve. https://hello.delve.sidewalklabs.com/. Accessed 20 Nov 2020
43. Li, S., Liu, L., Peng, C.: A review of performance-oriented architectural design and optimization in the context of sustainability: dividends and challenges. Sustain 12 (2020). https://doi.org/10.3390/su12041427
44. Müller, P., Wonka, P., Haegler, S., et al.: Procedural modeling of buildings. ACM Trans. Graph. **25**, 614–623 (2006). https://doi.org/10.1145/1141911.1141931
45. Dawkins, R.: The selfish gene. In: 40th Anniversary. Oxford University Press, Oxford, United Kingdom (2016)
46. Flake, G.W.: The Computational Beauty of Nature. MIT Press, Cambridge, MA, USA (1998)

47. Emmerich, M.T.M., Deutz, A.H.: A tutorial on multiobjective optimization: fundamentals and evolutionary methods. Nat. Comput. **17**, 585–609 (2018). https://doi.org/10.1007/s11047-018-9685-y
48. Bandaru, S., Ng, A.H.C.C., Deb, K.: Data mining methods for knowledge discovery in multi-objective optimization: part B—new developments and applications Sunith. Expert Syst. Appl. **70**, 119–138 (2017). https://doi.org/10.1016/j.eswa.2016.10.016
49. Coello, C.: A short tutorial on evolutionary multiobjective optimization. Evol. Multi-Criterion Optim. **1993**, 21–40 (2001). https://doi.org/10.1007/3-540-44719-9_2
50. Bhattacharya, M., Islam, R., Abawajy, J.: Evolutionary optimization: a big data perspective. J. Netw. Comput. Appl. **59**, 416–426 (2016). https://doi.org/10.1016/j.jnca.2014.07.032
51. Kosicki, M., Tsiliakos, M., Tsigkari, M.: Hydra distributed multi-objective optimization for designers. In: Impact: Design With All Senses. Springer International Publishing, Cham, pp. 106–118 (2020)
52. Branke, J., Schmeck, H., Deb, K., Reddy, S.M.: Parallelizing multi-objective evolutionary algorithms: cone separation. In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), pp. 1952–1957. IEEE (2005)
53. Van Veldhuizen, D.A., Zydallis, J.B., Lamont, G.B.: Considerations in engineering parallel multiobjective evolutionary algorithms. IEEE Trans. Evol. Comput. **7**, 144–173 (2003). https://doi.org/10.1109/TEVC.2003.810751