A Unified Vendor-Agnostic Solution for Big Data Stream Processing in a Multi-Cloud Environment

Thalita Vergilio (t.vergilio@leedsbeckett.ac.uk)
Leeds Beckett University
Ah-Lian Kor
Leeds Beckett University
Duncan Mullier
Leeds Beckett University

Research Article

Keywords:

Posted Date: January 20th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1253161/v1

A Unified Vendor-Agnostic Solution for Big Data Stream Processing in a Multi-Cloud Environment

Thalita Vergilio*, Ah-Lian Kor, Duncan Mullier
School of Built Environment, Engineering, and Computing
{T.Vergilio, A.Kor, D.Mullier@leedsbeckett.ac.uk}
*Corresponding Author

Abstract

The research field of cloud computing has witnessed tremendous progress as commercial cloud providers brought powerful distributed infrastructures within reach of small and medium enterprises (SMEs) through their revolutionary pay-as-you-go model. Simultaneously, the popularisation of containers has empowered virtualisation with seamless orchestration technologies for the deployment and management of large-scale distributed systems across different geolocations and providers. Big data is another research area which has developed at an extraordinary pace as industries endeavour to discover innovative and effective ways of processing large volumes of structured, semi-structured and unstructured data emitted at high velocity by an increasing number of internet-enabled devices. This research aims to integrate the latest advances within the fields of cloud computing, virtualisation and big data for a systematic approach to stream processing. The novel contributions of this research are: 1) MC-BDP, a reference architecture for big data stream processing in a containerised, multi-cloud environment; 2) a case study conducted with the Estates and Sustainability departments at Leeds Beckett University to evaluate an MC-BDP prototype within the context of energy efficiency for smart buildings.
"Keywords: multi-cloud; big data; containers; reference architecture; stream; vendor lock-in"

1. Introduction

1.1. Research Context

The advent of integrated yet pervasive systems such as the Internet of Things (IoT), the Internet of Everything (IoE), Fog/Edge and Cloud computing has brought big data to the forefront of technology research. Smartphones, tablets, GPS trackers, sensors, and video surveillance devices produce vast amounts of data, in varying formats, in real time. Consequently, this poses challenges not only in terms of storage, but also timely processing, and analysis for intelligence. Big data-related challenges are related to high volume, velocity, veracity, and variety. Historically, systems which focused on the volume aspect of big data appeared first. Those are known as batch architectures, as static finite data is stored in files and processed in batches. The most prominent and most highly utilised of such systems is Hadoop, an open-source distributed processing system based on the Map-Reduce algorithm developed by Google [2]. Hadoop is still currently popular, but has evolved to include a complete ecosystem of open-source applications for batch processing. However, the main critique of Hadoop and other batch systems is that they overlooked two important aspects of big data: velocity, and the fact that the data source is potentially infinite [3]. Batch systems were not designed to process streaming data and produce results in real-time, or close to real-time. Following the first generation of big data processing systems which focused on batch processing, stream systems were developed to process a potentially infinite source of data, arriving at high velocities, in real-time or close to real-time. The focus of these systems was different and therefore a new architectural design and approach to data processing was necessary. The concept of window was introduced which involved breaking up the data streams into manageable finite quanta for processing. This was facilitated by applying two functions: 1) assign a timestamp to arriving data; 2) based on the timestamp, assign data to a time window for processing. Thus,

time windows are abstracted groups (with a finite number of records) used for processing [4]. In order to address late and out-of-order streaming data, watermarks were introduced to represent the time by which all records for a given window are expected to have arrived [5]. It is used, in conjunction with a defined tolerance, to signal that a window is ready for processing.

The Lambda Architecture challenged the two main big data paradigms, namely, batch and stream, to propose a model where both technology stacks worked synergistically. Both batch and stream stacks were to be separately maintained, and data processing would be independently undertaken, with the results subsequently merged to obtain a combined view of the data. Although this architecture has been critiqued due to the need to create and maintain complex processing code in two separate places, thus incurring additional operational managing costs for two different technology stacks [6], it is still deployed in real-world applications (e.g. Facebook [7] and Twitter [8]). Though our proposed reference architecture does not completely address the Lambda Architecture overhead-related problem, it exploits the use of a super-framework to enable the same processing code to be run on batch or stream infrastructures, thus resolving the code duplication issue. Additionally, it advocates multi-tenancy so that both batch and stream frameworks share the same cluster. Currently, there are existing hybrid architectures where batch-based architectures have been adapted to process streaming data, as well as vice-versa, where stream-based ones have been adapted to process data from batch files.

Although, currently, a plethora of big data architectures is available, there is a discernible need for the development of systematic domain-specific approaches. Examples of numerous published domain-specific reference architectures for big data built to address this limitation are: BDWFMSs designed for the domain of data-intensive scientific workflows [9], Ta et al.'s (2016) healthcare-related reference architecture, and Klein et al.'s (2016) national security-related reference architecture. Approaches to develop an SME big data strategy tend to prioritise simplicity and convenience by focusing on big data as a managed service, whilst overlooking the ominous risk of vendor lock-in. As an example, Ardagna et al. (2016) highlighted the complexity of big data implementations as a significant barrier to SME adoption. They have proposed a Big Data Analytics as a Service (BDAaaS) model for the SME market. A similar cloud consumption model, Data as a Service (DaaS) [13], envisions big data cloud analytics as a crucial strategy to give SMEs a competitive advantage in the market. Both these models, however, are dedicated forms of SaaS which could incur a high risk of vendor lock-in. Liu and colleagues' research findings (2020) reveal that a significant barrier to the adoption of Big Data solutions by British manufacturing SMEs is the high cost of switching solutions. Big data challenges and potential within the SME market have been reviewed [15], while investigation of the opportunities for innovation created through big data has been conducted [16]. However, to date, a high-level vendor agnostic reference architecture specifically aimed at maximising economies of scale via commercial clouds does not exist. Consequently, this is an identified gap our research aims to address.

Currently, cloud computing business solutions are being made available to SMEs on a pay-as-you-go model. The cloud has brought with it economies of scale, making it more cost-effective for smaller companies to commission technology as services instead of purchasing hardware and maintaining their in-house IT staff. Such entry cost reduction has enabled more widespread adoption of computing resources to automate processes, leading to significantly increased data processing needs. Undeniably, more data means greater complexity, but also the potential for more accurate predictions and increased turnover. The profitability of big data initiatives is corroborated by an independent research study of 559 companies from various sectors conducted by the Business Application Research Centre [17]. The findings reveal that the participating companies reported an average eight percent increase in revenue and ten percent decrease in costs as a result of their use of big data [17]. However, most of them own in-house hosted technologies and are still sceptical about transferring their businesses to the cloud [17]. Given the elevated cost of big data implementations [18], [19], one could speculate that the cloud's economies of scale could potentially raise the reported profits.

The following three main service models describe how cloud resources are commissioned: IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service). IaaS is the most basic model with only infrastructure provision (e.g. servers, storage, virtual machines, and networking). PaaS, an intermediate model, adds an operating system, runtime, and middleware. SaaS is the most specialised model, with the entire package (including the applications and application data) managed by the cloud provider. The more specialised a model is, the greater the economies of scale, since the provider is responsible for commissioning and maintaining more resources at different levels. However, as a service model becomes more specialised, it also becomes less flexible and reliant on specific, if not proprietary, technology. However, a cloud consumer is understood to be vendor locked-in if they perceive better contract opportunities elsewhere in the market but is hindered from breaking an existing agreement with a provider. Vendor lock-in is a major cause of concern for potential cloud adopters [18], [20]–[22], and its mitigation is one of the motivating factors for the current research. Five different approaches to vendor lock-in mitigation have been reviewed in Section 2 of this paper. Recent, significant container-based virtualisation research has been conducted [23]–[26]. As containers provide decoupling between applications and the platforms to which they are deployed, they offer a level of mitigation against vendor lock-in. Furthermore, containers can be deployed on a PaaS (instead of IaaS) model because they represent a higher virtualisation level compared to virtual machines [23], [27]. When container and cloud computing technologies are deployed together, they afford greater economies of scale and facilitate greater collaboration amongst developers. As an example, images representing applications which are developed as containerised services have been shared in public repositories or libraries [28]. Such images can be deployed on any platform hosted on any infrastructure (cloud or non-cloud-based). Finally, the use of container and container orchestration technology also enables multi-cloud setups, which greatly mitigates the risk of vendor lock-in, giving cloud consumers the ability to compare and combine individual offers from providers to create the most appropriate and preferred configuration [19], [26]. Should changes in prices or service level agreements render it advantageous to do so, resources may be easily and seamlessly transferred between providers.

1.2. Aim and Objectives

To reiterate, the aim of this research is to propose a unified vendor-agnostic stream processing solution for big data in a multi-cloud environment. It is supported by the following research objectives: RO1 – to examine several existing architectures for big data processing and systematically evaluate the architectures of three well-known real-world companies (i.e. Facebook, Twitter and Netflix) with the purpose of understanding non-functional requirements for real-world big data systems. The selection criteria for these companies, methodology and results of this evaluation are explained in a separate publication [29]; RO2 – based on the findings in RO1, propose a new reference architecture based on industry's best practices and focused on vendor lock-in mitigation; RO3 - develop a prototype, based on a case study, to enable the empirical evaluation of the proposed reference architecture; RO4: to identify and implement a set of relevant performance metrics for the prototype in RO3; RO5 - to design and execute distinct sets of experiments to evaluate the proposed reference architecture in terms of the non-functional requirements implemented.

1.3. Rationale

The risk of vendor lock-in is perceived as one of the greatest obstacles to cloud adoption by companies [18], [22], [30], [31]. Consequently, as this research adopts a cloud consumer's perspective, the mitigation of vendor lock-in remains an important motivation throughout its development. The use of open-source technology is associated with a reduction in vendor lock-in risk [32], whilst simultaneously promoting

collaboration and reuse amongst the community [33]. Another way of mitigating vendor lock-in-related risks is by hosting the cloud infrastructure in multiple clouds. From a business perspective, implementers are less vulnerable to unilateral changes in price or SLAs by single providers when adopting a multi-cloud setup. Furthermore, they can achieve more flexibility by combining offers from different providers, or by changing providers for certain services, but not others, if a more advantageous offer is made available [34]. Motivated to achieve the greatest level of flexibility and interoperability of components across clouds, whilst at the same time taking advantage of the cloud's economies of scale, our research proposes a reference architecture based on the use of containers on a PaaS model. As containerised applications include all the environment configuration they need to run and can be deployed to any platform equipped with a container engine, they are fully portable across clouds, thus greatly reducing the risk of vendor lock-in and allowing implementers to shift away from the traditional SaaS model usually recommended for SMEs. This research therefore recommends that big data processing frameworks and other components of the proposed reference architecture be deployed as containerised services. From a technical perspective, a strong motivation of this research is to propose a reference architecture which has a discernible value for SMEs and is therefore, reliable and fault tolerant since high availability and business continuity have been identified as important requirements to these companies [13], [35]. Within a single commercial cloud, fault tolerance for containerised systems can be provided at three different levels: container, machine and region [36]. AWS has introduced the availability zones concept, which consists of isolated data centres hosted in the same region. Since no two availability zones share the same data centre, the resilience of the infrastructure is increased, even for single-region deployments [37]. An infrastructure that spans across multiple availability zones is indeed recommended by Netflix following the after-effects of AWS outages in the past [38]. However, zone failures involving multiple availability zones have indeed occurred in the past [39]. Likewise, a DNS disruption reported in 2016 affected Azure customers in all regions, effectively rendering the entire cloud unavailable (Dayaratna, 2016). Our research aims to explore the most resilient design for a cloud reference architecture and has thus identified an infrastructure distributed not only across multiple regions, but also across multiple clouds as a pattern to aim for. The relationship between different levels of fault tolerance and the cost and volume of data transfer from a cloud consumer's perspective is also investigated on in this research.

The rise in popularity of containers has led to significant developments in the discipline of distributed systems [22], [26], [28], [40]–[42]. In particular, the area of container orchestration and its support for hybrid and multi-cloud architectures has received increased attention [19], [24], [43]–[45]. Despite these advances, one related area which is yet to fully mature is that of resource consumption estimation for applications deployed in distributed container clusters. Currently, most cluster orchestration technologies allocate resources based on a user's initial request [26], [42]. Since users tend to overestimate the resources required by their applications, this approach leads to under-utilisation and, in the context of cloud computing, unnecessary costs [42]. Oversubscription is a possible solution, but a more systematic approach is called for in the literature [26]. Motivated by this gap in knowledge, the current research explores the relationship between the windowing function used in big data stream processing and the resource consumption of the cluster, with the aim of proposing a new approach to cluster size estimation for the domain of stream big data process.

1.4. Novel Contribution

In order to address the previously discussed challenges, this research proposes a new systematic and unified reference architecture for big data stream processing targeted at SMEs, organisations or departments characterised by devolved management, tight budgetary constraints and lack of in-house technical expertise (SMEODs). This new approach assumes a cloud consumer's perspective, and endeavours to facilitate the

adoption of stream big data analytics by: 1) promoting an infrastructure hosted on commercial clouds with pay-as-you-go model; 2) moving away from the traditional SaaS approach towards a standardised form of PaaS; 3) mitigating vendor lock-in risk by prescribing the use of portable, interoperable and vendor-agnostic components deployed to multiple clouds; and 4) providing a domain-specific reference architecture to guide implementers, thus alleviating concerns around complexity and skills shortage in the domain. In summary, the outputs of our research which constitute a novel contribution are: (i) the development, implementation and evaluation of Multi-Cloud Big Data Processing (MC-BDP), a new unified reference architecture for big data stream processing on a multi-cloud environment; (ii) a critical evaluation of the big data architectures of three well-known real-world companies: Facebook, Twitter and Netflix, with the purpose of understanding non-functional requirements for real-world big data systems; (iii) an innovative formula for adjusting the CPU and data transfer requirements of a distributed computer cluster for big data stream, based on the windowing function used to process the data.

The remainder of this paper is organised as follows; Section 2 Related Literature Review - situates the current work in the wider context of research in the fields of big data, cloud computing and virtualisation, and discusses related work; Section 3 Methodology - describes the methodology used in this research; Section 4 MC-BDP Prototype Evaluation - discusses the results of MC-BDP's evaluation; Section 5 – Conclusion and Future Work - considers the impact of our research within its field and presents recommendations for future work.

2. Related Literature Review

A comprehensive and appropriate review has been conducted on related work. Research in big data has flourished in recent years as the volume, velocity and variety of data generated by systems and devices connected to the internet escalates at an unprecedented rate and organisations become aware of the potential benefits of being able to harness and analyse this data. With so much data being generated at such a fast rate, existing technologies are stretched to their limits and sometimes fail to provide the level of performance expected to extract the maximum value from the data as it is generated. Within the field of virtualisation, research has flourished following the popularisation of Docker containers and the advent of sophisticated orchestration technologies enabling complex distributed architectures to be managed as a single cluster. Additionally, the popularisation of cloud computing and its associated pay-as-you-go model triggered a technological revolution by lowering the entry barrier for small and medium companies to commission complex and sophisticated systems. It was in the intersection of such rich and exciting research fields where new developments, both in the industry and in academia, appeared on an almost daily basis, that the current research project emerged. Reference architectures can be understood as an abstraction over concrete architectures aimed at a specific domain, or for a specific purpose, or both, based on which concrete architectures are derived. It is generally understood that every concrete system has an architecture, that is to say it is always possible to define its elements and their relationships in an abstract way [46]. The relationship between system and architecture can thus be abstracted as a one-to-one type, whereas a reference architecture holds a one-to-many relationship with specific implementations and concrete architectures. The objective of this research when proposing a reference architecture for stream big data processing in a multi-cloud environment was to provide a purely abstract template based on which specific architectures could be derived. MC-BDP is both domain-specific and purpose-specific, and although contributions were found that addressed one or even both aspects in the literature, an exact match was not encountered. This section takes a closer look

at recent work directly related to the products of this research and demonstrates the significance of this research's contributions to the wider scientific community.

A reference architecture for big data workflow management systems (BDWFMSs) was proposed by Kashlev et al. (2017) for processing data-intensive scientific workflows in the cloud. Although developed for processing big data using a distributed cloud-based infrastructure, BDWFMSs differs fundamentally from MC-BDP. Firstly, it was developed for the domain of scientific workflows, whereas MC-BDP is targeted at SMEODs entering the realm of stream big data analytics. Consequently, BDWFMSs does not specifically promote a multi-cloud environment, nor is it motivated by the desire to mitigate the risk of vendor lock-in. Virtualisation is provided at virtual machine level for BDWFMSs [9], whilst MC-BDP uses container-level virtualisation. Thus, BDWFMSs is more significantly affected by the lack of standardisation in virtual machine models and specifications offered by each provider than MC-BDP. When it comes to the experimental evaluation of BDWFMSs, its design shares some similarities with MC-BDP's such as the introduction of a computationally intensive calculation into some of the processing functions to observe how the system performs at higher loads [9]. Both BDWFMSs and MC-BDP used real historical data for their experimental setups: the former looked for patterns in driving data for fifty New York drivers over the course of an hour for its first case study, and analysed astronomical images for its second case study [9], while the latter calculated the energy efficiency of a data centre based on energy consumption records collected over the course of a year as part of its case study. An important difference between the experimental setups utilised to evaluate both reference architectures was that MC-BDP used a simulator to emit the data in real-time, whereas BDWFMS analysed static data in its case studies [9]. This difference can be explained by MC-BDP's focus on stream big data processing, whereas BDWFMS's main concern is with the processing of batch data.

The big data reference architecture developed by the National Institute of Standards and Technology (NIST) is a product of collaboration between academia, the industry and US-based governmental organisations [47]. NBDRA is composed of five different roles: system orchestrator, data provider, big data application provider, big data framework provider and data consumer, and two fabrics: security and management, each implemented independently based on functional requirements (NIST Big Data Public Working Group, 2019). One significant difference between NBDRA and MC-BDP is that the former has a wider scope, applying to big data in general, whilst the latter applies specifically to the domain of stream big data for the SMEOD market. As a result, the components or roles and fabrics identified by NBDRA are broader and more abstract. As an example, the big data application provider role defined by NBDRA is responsible for processing the data according to domain-specific business logic (NIST Big Data Public Working Group, 2019). In MC-BDP's proposal, a different emphasis was given to this role, represented as the processing code uploaded to the big data framework. One of the main concerns of this research was to investigate the overhead of running technologically agnostic code capable of processing both batch and stream data. Given that development time is expensive, particularly in the context of SMEODs at which MC-BDP is targeted, it was one of the objectives of this research to measure the impact of technology agnosticism on performance. Finally, the data provider and data consumer roles defined by NBDRA were not differentiated in MC-BDP's proposal. Since MC-BDP was designed for stream processing, it used the conceptual model of a directed acyclic graphic, common with stream architectures [48], [49]. Thus, the output of a process becomes the input of another and each node in the process is capable of being both provider and consumer of data. The biggest difference between NBDRA and MC-BDP is perhaps MC-BDP's focus on commercial clouds to provide the infrastructure required for big data processing, whereas NBDRA is neutral in this respect. This difference is explained by MC-BDP's focus on implementers whose presence in the big data arena is facilitated to a great extent by cloud computing.

The reference architecture proposed by Maier (2013) takes an approach similar to this research, starting with a thorough review of architectures and technologies for big data, following by the proposal of a reference architecture and subsequent evaluation. There are however some fundamental differences. Firstly, Maier's (2013) proposal focused on software only, whereas MC-BDP includes both hardware and software. In particular, MC-BDP offers an in-depth look at the advantages of leveraging container-based virtualisation technology to promote vendor lock-in mitigation and facilitate ingress into the big data market through cloud computing. Secondly, the reference architecture proposed by Maier looks mainly at batch processing frameworks. NoSQL databases and the Hadoop ecosystem are described as relevant big data technologies. However, stream-specific components of the Hadoop ecosystem such as Flink, Spark and Storm are not listed [50]. MC-BDP, on the other hand, focuses on stream processing. Finally, the evaluation of Maier's reference architecture was performed by analysing the real-world big data implementations of Facebook, LinkedIn and Oracle and retrofitting them to the reference architecture proposed [50]. While this research did perform a similar investigation of real-world big data implementations [29], its purpose was not to evaluate the MC-BDP proposal, but to gather requirements for it. The evaluation of MC-BDP was performed through a concrete case-study, with both quantitative and qualitative data collected and thoroughly analysed as part of a rigorous scientific process described in Section 3.

Despite being specifically aimed at the healthcare domain, the generic architecture proposed by Ta et al. (2016) has a number of elements in common with MC-BDP such as its reliance on existing open-source big data frameworks and a focus on stream data processing for providing real-time intelligence to a specific sector. One fundamental difference, however, is that while technologies such as Kafka and Storm are utilised by Ta et al. (2016) to prescribe a concrete architecture, MC-BDP is refers to them as mere implementation suggestions, given that it is a higher level abstraction. Another fundamental difference is that, even though Ta et al.'s (2016) architecture is focused on stream processing and real-time analytics, it contains a batch layer, implemented as a Hadoop cluster. This makes it an instance of the Lambda Architecture, as opposed to MC-BDP which is purely concerned with stream big data processing. One criticism which could be made when looking at Ta et al.'s (2016) contribution is that, although their architecture is aimed at the healthcare sector, little evidence was found of a quantitative or qualitative evaluation within this specific context. Improvements to existing medical systems and processes are merely alluded to, but not scientifically measured [10]. MC-BDP, on the other hand, provided a domain-specific case study evaluation using a mixed-methods approach to provide not only the quantitative measurements needed to rigorously assess the contribution from a positivist standpoint, but also a highly valued qualitative appraisal of its impact and potential applications within the target domain.

A reference architecture for big data in the cloud focused on cost and the pay-as-you-go model is offered by Heilig & Voß (2017). The authors propose three implementation possibilities for their reference architecture using stream processing technology offered by major commercial cloud providers, namely AWS, Google Cloud and Azure [51]. While some similarities can be drawn between this proposal and MC-BDP such as the focus on big data stream processing on infrastructure commissioned from commercial cloud providers, there are some fundamental differences in approach, the most important of which is regarding the preoccupation with vendor lock-in mitigation. Heilig & Voß's (2017) implementation proposals using AWS, Google Cloud and Azure are based on proprietary services offered by these providers which are neither portable nor interoperable with components on different clouds. The authors state that there was a requirement for each service proposed to be compatible with other services within the same provider. Thus, as an example, Amazon Kinesis Streams was selected to provide data to Kinesis applications, and Google Cloud Pup/Sub

was selected to provide data to Google Cloud Dataflow [51]. There was however no requirement for the services selected to be compatible with services from other clouds, thus reducing the risk of vendor lock-in. Since vendor lock-in mitigation is one of the motivations of this research, the use of managed services is avoided as discussed in Section 2.2.

Sang et al. (2016) proposed a reference architecture for big data based on five components:

- data source,
- data collection, processing and loading,
- data analysis and aggregation,
- interface and visualization,
- and job and model specification.

In an exercise similar to that performed in the early stages of this research [29], the architectures of real-world big data systems, namely Facebook, LinkedIn, Twitter and Netflix, were studied and their implementations mapped to the five components proposed [52]. As with the reference architecture proposed by Maier (2013), real-world implementations by lead big data companies were used to evaluate the authors' contribution. Our research took a different approach by using the information gathered from real-world implementations to derive a set of non-functional requirements for big data based on which the MC-BDP reference architecture was designed [29]. Additionally, a possible implementation was suggested by the authors using AWS technology. As with Heilig & Voß's (2017) proposal, the implementation offered by the authors was based on managed services specific to AWS [52] and would not have been portable to other clouds. MC-BDP, on the other hand, was designed to be portable and interoperable across multiple clouds in order to mitigate the risk of vendor lock-in.

The reference architecture for big data proposed by Klein et al. (2016) is similar to MC-BDP in that it was designed for a specific domain. Whilst MC-BDP focuses on the domain of stream big data for implementers looking to take advantage of the economies of scale brought about by cloud computing, Klein et al.'s contribution was developed for the domain of national security. Technology agnosticism is a main concern of the reference architecture proposed by Klein et al. (2016), and one which it shares with the current research. Klein et al.'s contribution is based on thirteen modules, further grouped into three main categories: big data application provider, big data framework provider and cross-cutting modules [11]. Whilst some parallels can be drawn between their modules and MC-BDP's layers, the most significant similarity between the two references architectures is the treatment of security as orthogonal to all other modules [11]. Considering that Klein et al.'s reference architecture was designed for the specific domain of national security, it is reassuring to find that its approach to systems security does not significantly diverge from MC-BDP's. There is however a fundamental difference when it comes to the underlying infrastructure to which derived architectures are deployed. While Klein et al. (2016) make no mention of a cloud infrastructure, MC-BDP advocates the use of container-level virtualisation on a PaaS model since it was designed to take advantage of the cloud's elasticity and pay-as-you-go model whilst minimising the risk of vendor lock-in. Finally, the evaluation of Klein et al.'s reference architecture was performed using a simple prototype implementation to analyse a data flow of Twitter sentiment data using stream processing and merge it with a news dataflow obtained through batch processing to produce intelligence [11]. Differently from MC-BDP's prototype evaluation, Klein et al.'s (2016) was based on a typical Lambda Architecture implementation where stream and batch data were processed separately before the results were merged [11]. There was no controlled variation of the velocity of data ingress as performed for the MC-BDP evaluation, since the authors' objective was to demonstrate the feasibility of the reference architecture, not measure its performance [11]. Case-specific implementations of Klein et al.'s (2016) reference architecture within the domain of national security were called for as future

work to enable a more thorough evaluation of their proposed reference architecture, similar to what was suggested by participants of the MC-BDP case study following its prototype evaluation within the domain of smart buildings.

Pääkkönen & Pakkala (2015) derived their reference architecture from studying publications related to four real-world big data implementations: Facebook, Twitter, Netflix and LinkedIn. This is a similar approach to the study conducted early in this research where the Facebook, Twitter and Netflix architectures for big data were analysed to gather non-functional requirements to inform the design of the MC-BDP reference architecture [29]. The use of an inductive process to arrive at a generalisation common to the observed cases and applicable to future cases is a known and widely used empirical methodology with its roots in Kant's (1781, p.137) philosophy of science. Within this review, a similar inductive process to derive a reference architecture from known big data implementations from real-world companies was carried out by Sang et al. (2016) and Maier (2013). One criticism which can be made to this approach is that the systems were not evaluated in-situ, nor were the source code and primary data available to the researchers. The aforementioned studies, including the one performed as part of this research, were carried out based on academic papers and other official publications made available by the target companies. Indeed, it would have been advantageous to conduct a case study with the companies and to access their data, systems and participants directly. However, as these are very large global companies, a case study such as the one proposed was not feasible. A different strategy altogether would have been the exclusive utilisation of first-hand materials obtained directly from case studies in the design of a reference architecture for big data. Given the difficulties involved in securing case studies with large-scale global organisations, the final product of such a project would not necessarily reflect the big data requirements of very large companies, nor would it benefit from their real-world experiences and lessons learned. In order to address this methodological shortcoming and balance the type of information used in the design of its reference architecture, this research conducted a case study with a real, albeit smaller, organisation in addition to researching real-world big data systems. The evaluation of Pääkkönen & Pakkala's (2015) reference architecture for big data systems was performed by retrofitting it to the real-world implementations at Facebook, Twitter, Netflix and LinkedIn. This approach provides insufficient validation for the proposed model since it was derived from these very same architectures by induction, so the generalisation should logically retrofit the particular observations. The need for further validation through a real-world use case was acknowledged by the authors as a limitation [53], and is addressed by the current research.

Belli et al. (2015) proposed an architecture for stream big data which shares some common aspects with MC-BDP such as its focus on stream processing, reliance on open-source technologies and evaluation using real-world data emitted by a purpose-built simulator. Amongst the main differences are Belli et al.'s (2015) aim to produce an architecture focused on the IoT domain, whilst MC-BDP is a reference architecture aimed at the SMEOD market. Additionally, and possibly due to the difference in focus, the former proposes the use of a bare-metal infrastructure whilst the latter relies on a multi-cloud infrastructure. The evaluation methodology of Belli et al.'s (2015) proposed architecture is comparable to that used for MC-BDP's empirical evaluation. A simulator was used to emit smart parking streaming data in real time, at different velocities, and the acquisition and processing times were independently measured to understand the overhead introduced by the proposed architecture [55]. The velocities used varied between one and one hundred messages per second [55], whilst in MC-BDP's experiments they varied between two and two thousand messages per second, making the latter a more comprehensive test of the prototype under heavy load. The focus of the experiments and the metrics observed were fundamentally different. While Belli et al were interested in the speed of data processing, this research monitored performance metrics such as CPU, memory and network utilisation to

understand the overhead introduced by different features of the proposed reference architecture such as its multi-cloud proposal, as well as scalability, fault tolerance, technology agnosticism, the use of windowing to process streaming data and container co-location. The speed of data processing was not addressed by the current research since it depends on the calculations being performed. In effect, the calculation of a large factorial was introduced as part of the processing for some experiments to observe the impact of scaling the cluster up or down on the performance metrics monitored. Belli et al. (2015) make it clear that their objective was to understand the overhead introduced by the actual architecture, so application-specific processing time was subtracted from their calculation. In MC-BDP's case, it would not have made sense to measure the overhead introduced by the actual architecture since MC-BDP is a reference architecture and therefore abstract. The prototype implemented as part of MC-BDP's empirical evaluation is an example of one of many possible compatible architectures. An interesting venture, although out of scope for this project, would have been to implement a different MC-BDP-compatible prototype and then compare the performance of the different prototypes in terms of data processing speed.

Pellegrini et al. (2017) proposed a specific technology stack for the utilisation of commercial cloud computing resources with minimal risk of vendor lock-in. Although not aimed at big data, their proposal shares some common aspects with MC-BDP such as the use of container-level virtualisation, an orchestrator, and promotion of a multi-cloud deployment as mitigation against the risk of vendor lock-in. In order to verify the viability of the technology stack proposed, a prototype was built whereby a simple browser game was deployed to three commercial providers as a containerised service [56]. Nevertheless, plans on how the prototype was going to be evaluated, as well as a full account of the research's methodology was not shared by the authors in this publication [56], making it difficult therefore to fully compare it with MC-BDP.

The architecture proposed by Scolati et al. (2019) for the domain of edge cloud computing is an example of a Lambda Architecture implementation based on Hadoop for batch and Spark for stream processing. Although aimed at very different domains, Scolati et al.'s (2019) research is relevant to the current study as it corroborates the empirical methodology followed in MC-BDP's evaluation. The authors evaluated the performance of their containerised architecture based on Docker and Swarm by:

- measuring CPU and memory utilisation at container and node level,
- disregarding disk I/O was, since it is not relevant to stream processing,
- monitoring the total processing time [57].

The three decisions above were independently reached by the current research as part of its case study evaluation. The differences, however, are more significant:

- different aims: Scolati et al.'s (2019) architecture was developed for the domain of edge computing where computation is performed by smaller edge devices before being transferred to the cloud. MC-BDP, on the other hand, was developed for SMEODs and aims to transfer as much of the computation as possible to the cloud to take advantage of its economies of scale.
- different infrastructure: as consequence of the above, Scolati et al.'s (2019) prototype was deployed to a cluster of eight Raspberry Pi 2 machines hosted locally. MC-BDP's used clusters of up to ten virtual machines hosted in the Azure, Google and OSDC clouds.
- different container allocation: Scolati et al. (2019) used a fixed cluster of one master, four workers and three data collection containers. MC-BDP used one master, a varying number of workers ranging from three to eight, and no dedicated data collection containers since data from Kafka was streamed directly to the jobmanager via a TCP socket connection.

- different container co-location: a fixed co-location of one container per machine was used by Scolati et al. (2019), while different co-locations were observed in MC-BDP's experiments.
- different velocities: data was emitted at a fixed velocity of one record per second in Scolati et al.'s (2019) evaluation, whilst the emission rate in MC-BDP's experiments varied from two records per minute up to a maximum rate of two records per millisecond.
- different windowing approach: whilst different windowing functions were observed in MC-BDP's evaluation, there was no mention of a windowing strategy being used in Scolati et al.'s (2019) experiments.

In conclusion, although Scolati et al.'s (2019) choice of metrics and experimental setup were similar to MC-BDP's and provided validation for the decisions taken earlier in this research, MC-BDP's evaluation was considerably more extensive, making it an important contribution to the field.

A security reference architecture (SRA) for big data based on the UML notation was proposed by Moreno et al. (2018). Their contribution builds upon other industry-standard reference architectures, particularly the NIST Big Data Interoperability Framework [47], by adding a security dimension defined using UML diagrams [58]. Although this proposition differs significantly from MC-BDP, it is included in this review because it highlights the importance of incorporating security aspects into the early design stages of a big data implementation [58]. This is the approach supported by this research and integrated into MC-BDP's security layer presented in Section 4.

A recent contribution by Chen et al. (2020) proposed a real-time scheduling algorithm for distributed big data stream processing in a multi-region cloud infrastructure. Using task duplication, the algorithm created by the authors schedules parallel tasks across different configured regions within the same cloud, aiming to maximise resource utilisation whilst minimising both costs and completion time [59]. The purpose of the Chen et al.'s (2020) work is rather different from that of the current research: whilst the former's main preoccupation was with achieving an efficient way of running parallel co-dependent workflows by scheduling (and sometimes duplicating) their component tasks, this research delegates the scheduling implementation to the orchestrator and focuses, instead, on the underlying architecture. Nevertheless, both studies are concerned with stream big data processing in a heterogeneous cloud environment where price variation is particularly significant to implementers, so the publication of Chen et al.'s (2020) contribution serves to reinforce the relevance of this research's topic to the field of big data.

This concludes the background and literature survey provided in this section, which discussed specific academic work directly related to this research's contributions. The next section provides a detailed presentation and in-depth discussion of our research's methodology.

3. Methodology

From a wider perspective, six strategies for carrying out academic research are identified by Oates (2005): survey, design and creation, experiment, case study, action research and ethnography. Our research adopts the design and creation strategy, since it proposes a completely new artefact to address gaps in the literature identified by previous research work. A case-study involving the Estates and Sustainability departments at Leeds Beckett University was used to evaluate the product of this research. As part of this case-study, a prototype was implemented using energy consumption data from historic readings and evaluated using a combination of methods underpinned by distinct philosophical paradigms, namely post-positivism and

interpretivism. Specifically, the empirical evaluation of the prototype follows a post-positivist methodology with falsifiable hypotheses and strictly controlled experiments, whilst the focus group evaluation of the same prototype follows an interpretivist methodology where the qualitative data collected is coded and analysed using a process derived from grounded theory.

Using a classification based on traditional methodological standpoints (Venkatesh et al., 2013), this research is neither purely quantitative nor purely qualitative, but falls under the category of mixed-methods research since both quantitative and qualitative methods are utilised, albeit at different stages. The reason for this approach was to provide a multifaceted and more extensive evaluation of the products of this research as evidenced in studies that examine the use of mixed methods in the field of information systems (Orlikowski & Baroudi, 1991; Landry & Banville, 1992; Mingers, 2001; Venkatesh et al., 2013).

3.1. Component-Based Software Engineering Lifecycle

This section describes the phases in a macro-level component-based software engineering lifecycle (depicted in Fig. 1). A gap was initially identified in the literature reflecting a shortage of systematic academic studies where container technology was applied to the domains of big data processing and cloud computing (Naik, 2017). Additionally, reference architectures with a focus on cloud consumers based on a model other than SaaS are under-studied. To address this gap, our research aims to leverage the latest advances in containers and container orchestration technology for a new PaaS-based multi-cloud reference architecture targeted at big data stream processing.

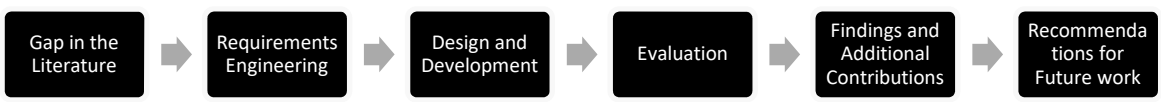


Fig. 1. Macro-Level Component-Based Software Engineering

As part of the requirements engineering phase, we have conducted a critical review of published work by three major big data corporations: Facebook, Twitter and Netflix [1], [29]. Ten non-functional requirements were identified and discussed in the context of these companies' architectures: batch data, stream data, late and out-of-order data, processing guarantees, integration and extensibility, distribution and scalability, cloud support and elasticity, fault tolerance, flow control, and flexibility and technology agnosticism. Based on these requirements and industry's best practices gathered from real-world implementations, a new MC-BDP reference architecture for big data stream processing in a multi-cloud environment was designed and developed. Subsequently, this architecture was implemented as a prototype in an energy efficiency case study involving the Estates and Sustainability Departments at Leeds Beckett University. This case study used real data obtained by a previous data centre energy efficiency study (Pattinson et al., 2012) to calculate its PUE (Power Usage Effectiveness). Since the mechanisms in place to emit consumption data and calculate the PUE had not evolved since the original study took place, the data collected remained relevant. The case study targeted by our research has been selected because it has characteristics which are appropriate for the testing of our prototype [60]. A needs analysis interview revealed that the Estates and Sustainability departments at Leeds Beckett University managed data which had inherent big data characteristics: volume (log files with operational data that spans several years), velocity (data from meters, that is sampled and streamed in real-time), and variety (device log exports, for example, contain unstructured data, as the format is specific to each manufacturer). The real-time sampling rate used for the data centre energy efficiency study is, however,

limited by the technology available at the time. For the purpose of this research, the data granularity has been enhanced to meet the higher volume and velocity requirements, together with MC-BDP's inherent strategy for dealing with late data (Vergilio & Ramachandran, 2018b). Due to operational and availability constraints, a simulation exercise has been carried out for the evaluation. Instead of using the real EGX300 server located at one of Leeds Beckett University's data centres, a simulator has been written to emulate the behaviour of this server. The simulator used real energy consumption data, obtained from the previous Power Usage Effectiveness study [61] and readings were transmitted at desired frequencies using an interpolation algorithm. While using a simulator carries lower risks and is time and resource-efficient, it has the disadvantage of not providing real-time insight into live data. To reiterate, the main purpose of this case study was to conduct a technical evaluation of the proposed reference architecture. Three out of the ten non-functional requirements have been selected: scalability, fault tolerance, and technology agnosticism (details are found in Section 4). This selection is supported by current research in the field, as scalability and fault tolerance have been identified as key issues in a systematic literature review of big data stream analytics [62]. Correspondingly, a major factor of concern, the vendor lock-in risk, was mitigated through technology agnosticism. The seven remaining functional requirements: batch data, stream data, late and out-of-order data, processing guarantees, integration and extensibility, cloud support and elasticity and flow control were not evaluated due to time and scope limitations of this project.

A set of performance metrics have been identified which include CPU, Memory and Network utilisation, and experiments were designed to evaluate the prototype implementation of the proposed reference architecture. The experiments were run with varying parameters: single and multi-cloud environments, different cluster sizes, as well as different velocities for incoming data. Based on initial findings, two additional sets of experiments have been designed to verify the following: (i) the relationship between resource consumption and selected windowing rate for stream processing; (ii) the impact of container co-location on resource consumption. Details of the findings are found in Section 4. The experimental procedures for the simulations are described in the next section.

3.2. Experimental Procedure for all Simulations

The experimental procedure common to all simulations is summarised in Fig. 2. It starts with booting all participating machines, which are connected to the internet via SSH secured by public key authentication. The decision to use a public key instead of password authentication was informed by security considerations at the design stage of the prototype, as recommended by the Cloud Computing Adoption Framework (CCAF) proposed by Ramachandran and Chang (2016). Public key authentication is more secure than password authentication given that, in the event of the server being compromised, password authentication confirms a valid username/password combination to a potential attacker [64].

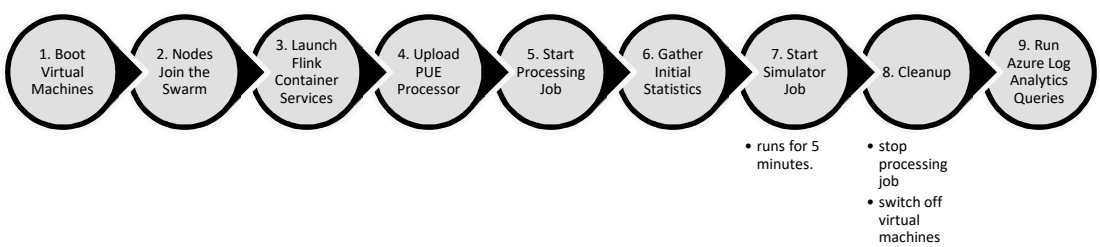


Fig. 2. Experimental Procedure for all Simulations

The next step is to run a command in each machine to register with the Swarm manager as a worker. As part of this step, the machine's IP address is sent to the Swarm manager via SSH. Since the Azure machines have static IP addresses, this step must be completed only once for all experiments. The Google Cloud machines are configured with ephemeral IP addresses, as static IP addresses are not available for purchase. For this reason, they must be registered with the Swarm manager after each reboot.

The Flink container services were launched from the Azure-swarm-manager virtual machine via the Docker Swarm interface, accessible through the SSH console. The Flink jobmanager service used the multi-cloud network created using the Weave plugin, and was constrained to run on a dedicated machine called azure-flink-jobmanager. This constraint was introduced to isolate the jobmanager service from the taskmanager service running on worker nodes, thus allowing for more granular monitoring and gathering of performance statistics. The image used to create the jobmanager service was obtained from the Docker Hub repository, and was minimally modified. The Flink taskmanager service, distributed across different containers, also used the network created using the Weave plugin. A constraint was added to run this service on any machine but the one dedicated to run the jobmanager service. Once the taskmanager service was up and running, it was scaled to the desired level of parallelism using the Docker Swarm interface.

The jobmanager service displays a graphical user interface that can be used to upload the JAR file containing the PUE Processor implementation. The desired parallelism for the job is set to match the parallelism of the taskmanager service, i.e. how many instances of the service are running on separate containers distributed across the cluster. The job is automatically started once submitted.

Once the data processing job is running, initial statistics are gathered using the Weave Cloud service [65]. CPU and memory utilisation metrics are verified to check for discrepancies such as readings which are abnormally high or low. Any reading above 2% or below 0.1% for CPU utilisation was considered abnormal, and the setup process was restarted with a fresh reboot of the machine where the abnormality has been observed. Similarly, any reading above 600MB or below 200MB for memory utilisation was considered abnormal and warranted a reboot of the machine and restart of the setup process.

With the data processing job running and ready to receive data, the Energy Consumption Simulator job is started using the graphical user interface (discussed in Section 3.5). It runs for five minutes transmitting data at the desired frequency, which varies depending on the experiment being performed. Once the Energy Consumption Simulator finishes transmitting data, it displays the start and end transmission times. These are noted and used to set the times the Azure Log Analytics queries are run 'from and to'. The setup is then cleared to prepare it for the next experiment: the PUE Processor job is stopped via the Flink jobmanager interface, the Flink jobmanager and taskmanager services are stopped using the Docker Swarm interface, the nodes are removed from the swarm, and the virtual machines are switched off. The final step in the process is to run the Azure Log Analytics queries using the Azure Portal. The 'from and to' times are set to the start and end transmission times. The queries executed are shown in Fig. 3.

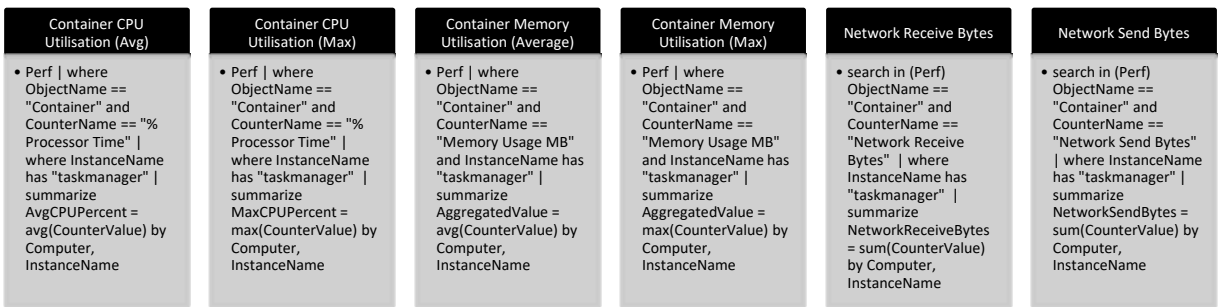


Fig. 3: Azure Log Analytics Queries to Extract Metrics

For the Container Co-Location experiments, because the number of containers per node was no longer fixed at one, in order to compare the performance of clusters with different co-location distributions, the container measurements were aggregated, as per **Error! Reference source not found.**4, to generate node-level values. For metrics which constitute a sum (node network receive bytes, and node network send bytes), the aggregation was calculated as the sum of the container-level measurements. For metrics which constitute an average (average node CPU utilisation and average node memory utilisation), the aggregation was calculated as the sum of the average utilisation of each container running on a node. Finally, for metrics which constitute a maximum, the aggregation of measurements at container level to derive metrics at node level (CPU utilisation max, and memory utilisation max) is inherently more complex.

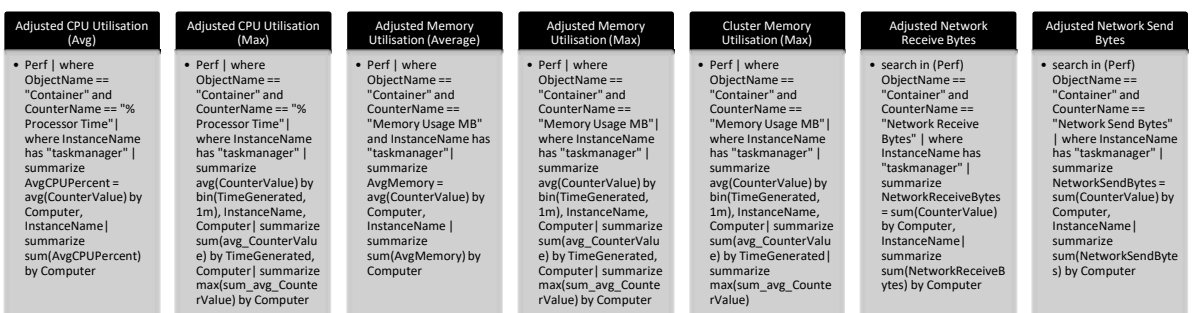


Fig. 4: Azure Log Analytics Queries to Extract Metrics – Adjusted to Aggregate by Node

The experimental procedures common to all simulations have been described in this section. Due to limited cloud computing funding, data collected for each experiment is based on a single run of the procedure described. Whilst discrepant results were discarded and anomalous experiments repeated on an ad-hoc basis, it is believed that a more systematic approach to scheduling multiple runs for each experiment would have benefitted this research by reducing experimental error and increasing the accuracy of the results observed.

3.3. MC-BDP Reference Architecture for Big Data Stream Processing

MC-BDP is an evolution of the PaaS-BDP architectural pattern originally proposed by the authors. While PaaS-BDP introduced a framework-agnostic programming model for batch and stream processing and enabled different frameworks to share a pool of resources [66], MC-BDP focuses on stream processing and expands the previous model by explicitly prescribing a multi-tenant environment where nodes are deployed to multiple clouds [1]. To reiterate, the rationale for proposing a multi-cloud model is to mitigate the risk of vendor lock-in, perceived as a major obstacle to the adoption of cloud computing.

3.4. MC-BDP Reference Architectural Layers

The MC-BDP Reference Architecture is depicted in Fig. 5. The model comprises 6 horizontal and 3 vertical layers which are described in more detail in Table A1. At the lowest level is the persistence layer, used in different ways and to varying degrees by components in the nodes, containers, services, and messaging layers. The next layer represents nodes or machines which can be physical or virtual. It is followed by the containers layer, which provides an additional layer of virtualisation, isolation, and abstraction on top of each node. Networking is represented next since, in a distributed system used for big data processing, all processing units must be capable of communicating in order to enable parallel data processing. Networked containers working as part of a single distributed system must be centrally managed and coordinated, which is why orchestration is shown as the next layer in the diagram. Finally, a services layer contains all the application services deployed in the infrastructure described, including those responsible for processing big data as a stream and those used for monitoring and analytics. Security, monitoring and messaging are represented as vertical layers as they permeate every other layer in the diagram.

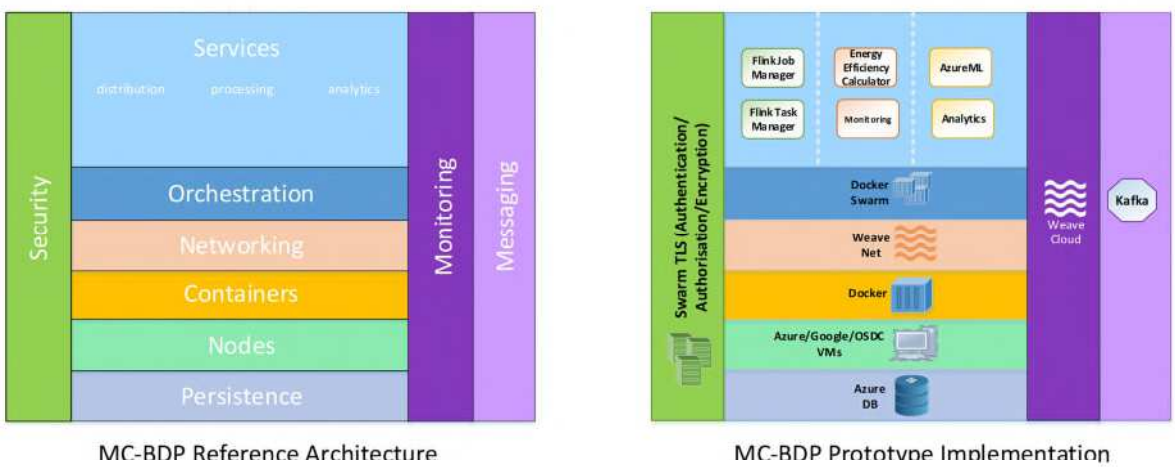


Fig. 5. MC-BDP Architectural Layers (Source: Vergilio et al., 2019)

3.5. MC-BDP Prototype Implementation

This section summarises the case study based on which a prototype was created to evaluate the MC-

BDP reference architecture. It also describes the results of MC-BDP's quantitative evaluation. A needs analysis exercise conducted at the start of our research revealed the existence of previous research conducted in 2012 with the Sustainability Centre at Leeds Beckett University in to assess the energy efficiency of a data centre by calculating its Power Usage Effectiveness (PUE) [61]. It was decided that the current case-study would build upon the previous study by leveraging more advanced technology with an aim of providing close to real-time data processing.

The PUE, a measurement of an IT building's power efficiency, is calculated by dividing the total energy consumed by the facility by the total energy consumed by the IT equipment running in that facility. In the case of Leeds Beckett University's original study, it was calculated as the total energy consumed by each of the 6 rooms in the Woodhouse Data Centre divided by the total energy consumed by the IT equipment installed in these rooms [61]. Fig. 6 summarises the original PUE calculation process: energy consumption readings from IT equipment and from the rooms in the data centre building were sent to an EGX300 Integrated Gateway Server. The data was then extracted into Excel reports as an ad-hoc process. Note how the readings were emitted every 30 minutes, and the log extraction into Excel took 43 minutes to run [61]. Moreover, the reporting was produced retrospectively, on a pull basis.

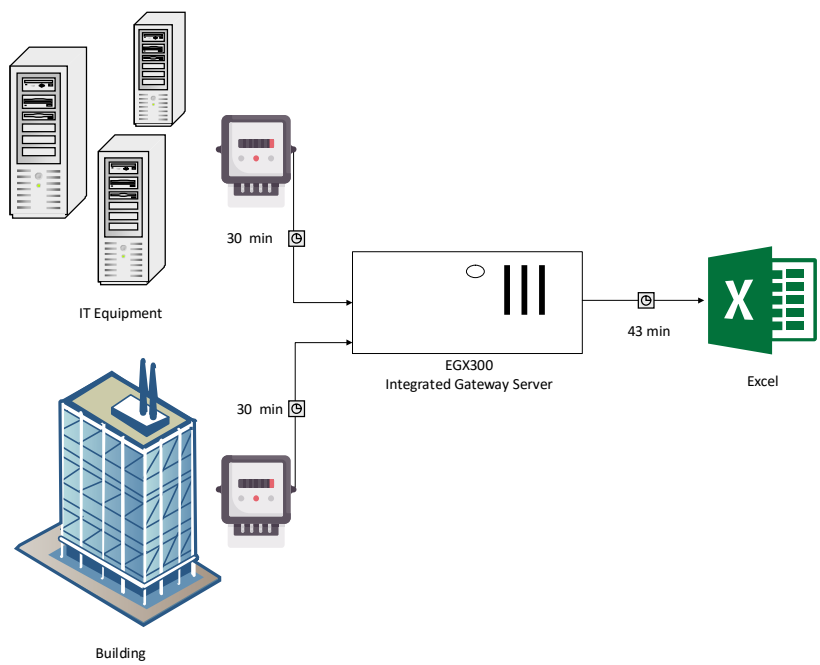


Fig. 6. Original PUE Calculation Process at Woodhouse Data Centre

It was decided that the PUE calculation for the Woodhouse Data Centre at Leeds Becket University would be a good case for a prototype implementation of MC-BDP, as the original experiment's volume and velocity of data processing were limited by the technology available at the time. MC-BDP uses a scalable distributed

cluster and a streaming framework designed for big data, so the frequency of the reading emissions could theoretically be increased more than a hundredfold and still achieve close to real-time processing. A simulator was therefore developed to emulate the behaviour of the EGX300 Integrated Gateway Server at the Woodhouse Data Centre at Leeds Beckett University.

The components involved in the prototype implementation of MC-BDP and their interfaces are shown in Fig. 7. The energy consumption simulator provides a simple web-based interface through which a user can launch a simulation (e.g. transmit energy consumption records every 5 seconds for 10 minutes). It interfaces with the energy consumption producer component which gets historical readings from a database hosted in the Azure cloud, uses the data interpolator to generate realistic readings based on the historical readings, and emits the results to a Kafka server in the frequency requested. The PUE processor component is a distributed big data stream processing pipeline which runs on the Flink framework. It consumes the energy readings from Kafka, calculates the PUE, then exposes the results which, in the example depicted in the diagram, are posted to an Azure Event Hub. Different configurations were used in the development, including logging the results to a file to verify accuracy and posting it back to a different Kafka topic. Microsoft Power BI integrates easily with the Azure Event Hub and was used to create a simple dashboard to display the PUE calculations in real-time. The formatted PUE results from the Azure Event Hub were also stored using Azure Blob Storage as proof-of-concept. The framework and orchestrator components shown on the bottom right-hand side of the diagram show how the processing job is launched and the parallelism is set through the job manager interface. The work is then distributed across several parallel task managers. The number of job managers and task managers available is controlled by the orchestrator, since these components run as containerised services, and the interface to scale these up or down is also exposed.

The EGX300 Integrated Gateway Server simulator is used to transmit data for the prototype implementation. The simulator comprises 3 modules: a data interpolator, a transmitter to transmit readings at desired frequencies, and a front-end user interface. The Data Interpolator component is a simple Maven project with a single class and a single static method. The static method takes an original Consumption record for a given duration as a parameter, and outputs simulated Consumption records for a shorter duration contained in the initial duration. Instead of using a simple average to calculate how much energy was consumed for smaller durations contained within the original duration, the consumption values generated vary randomly, up to a maximum variation percentage passed in as a parameter. The Energy Consumption Producer component was developed to gather energy consumption records from the Original Readings Database, generate interpolated records using the Data Interpolator, and emit generated readings to the Kafka Server in the desired time interval. Finally, the Energy Consumption Simulator is a very simple out-of-the-box MVC application designed to gather runtime values from the user and pass them to the Energy Consumption Producer. It captures user input for how many records to retrieve from the database, the maximum number of records to query per request, the duration that each original record corresponds to, the desired duration, the desired variation and the credentials to access the database. The full implementation code, unit tests and commit history for all components are available in public repositories [67]–[69].

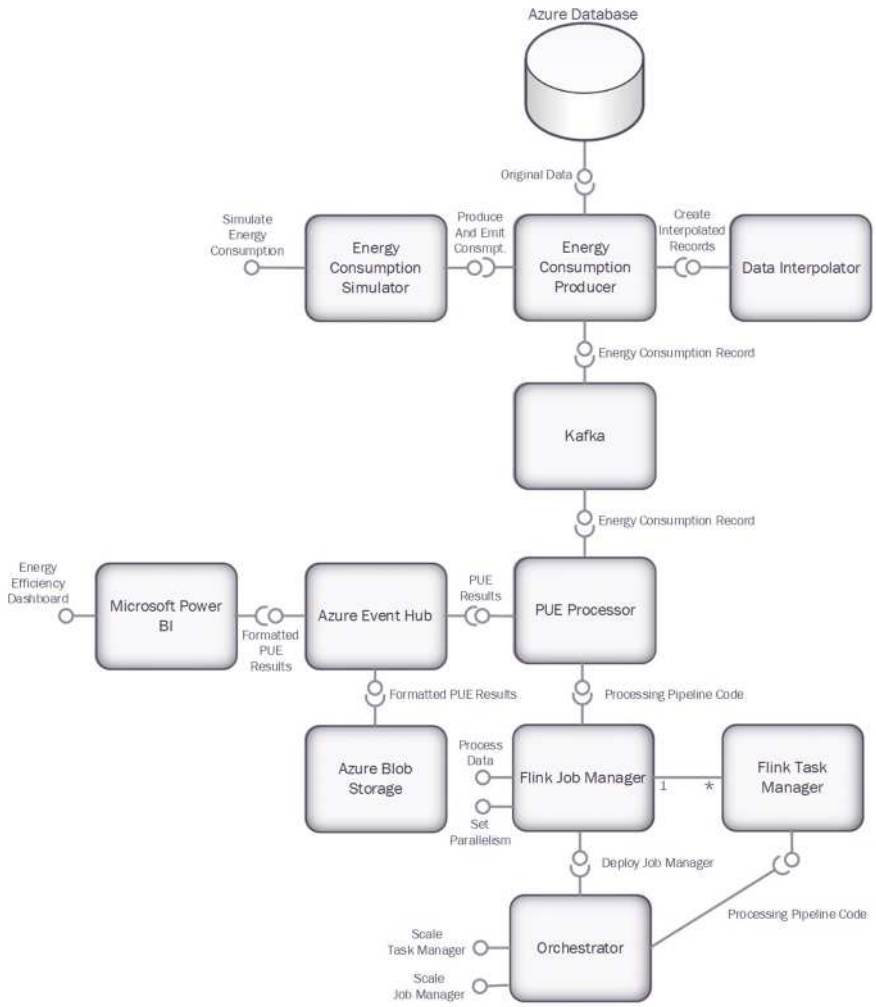


Fig. 7. MC-BDP Proto type Implementation for the Data Centre Energy Efficiency Case Study

4. MC-BDP Prototype Evaluation

4.1. Experimental Design

This section describes the results of the quantitative evaluation of the MC-BDP reference architecture for big data stream processing in a multi-cloud environment. The evaluated dimensions are: scalability, fault tolerance, technology agnosticism, windowing rate versus resource utilisation, and container co-location. A total of 110 experiments have been conducted for this research. The details of the experimental design have been tabulated in Tables A2-A4.

4.2. Experimental Results

The experimental results for the evaluation of the MC-BDP reference architecture are described in this section. Three categories of experiments were run where each metric was monitored for different velocities, windowing rates and cluster sizes. The velocity for all experiments was measured as 2 records per time unit and represents the speed of emission for: 1) total energy consumption of the data centre, and 2) energy consumption by the IT equipment in the data centre. The windowing rate represents the period (how often a window of processing starts), divided by the duration (how long each window lasts). For example, the Scalability, Fault Tolerance and Technology Agnosticism experiments used a sliding window of five seconds, starting every second, for data processing. Non-parametric Mann Whitney U tests (two independent samples) have been employed for hypotheses testing due to the very small sample sizes [70]. The significance level chosen for all tests was $\alpha = 0.05$. The null hypothesis (Ho) was that the means of the two samples were not significantly different, while the alternative hypothesis, Ha, stated otherwise.

4.2.1. Average Container CPU Utilisation

Based on the summary results for average CPU utilisation presented in Table 1, the following conclusions were drawn: (i) average container CPU utilisation by velocity for scalability – means of three or six workers (single and multi) are not significantly different from each other. The same conclusion was drawn for all the possible combinations of three workers (single and multi) compared to six workers (single and multi); (ii) average container CPU utilisation by velocity for fault tolerance – same conclusions for all combinations in (i); (iii) average container CPU utilisation by velocity for technology agnosticism – the means for three or six workers (Beam, Flink) x (single, multi) were found not to be significantly different from each other. The same conclusion was drawn for three workers [(Beam, Flink) and (single, multi)] x six workers [(Beam, Flink) and (single, multi)]. In terms of number of workers (3, 6, 10), the means were also not significantly different from each other for average container CPU utilisation by windowing rate.

Table 1

Average CPU Utilisation Hypotheses Testing Results	Scalability Metrics (for Workers)					
Average Container CPU Utilisation By Velocity	Velocity	Three Workers (n=3)		Six Workers (n=6)		Mann Whitney U Test Results
	2 records per time	Single %	Multi %	Single %	Multi %	3 Workers and 3 Workers (Single and Multi)
	1 min	3.52	3.44	2.51	3.61	p-value = 0.8857, accept Ho
	1 sec	37.94	40.21	18.22	23.17	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	42.25	36.39	25.46	26.24	p-value = 0.1905, accept Ho (S,S)
	10 ms	41.58	45.46	23.02	29.27	p-value = 0.2857, accept Ho (S,M) p-value = 0.1905, accept Ho (M,S) p-value = 0.2857, accept Ho (M,M)
	1 ms	NA	NA	36.26	33.02	6 Workers and 6 Workers (Single and Multi)
	Average	31.32	31.38	17.30	20.57	p-value = 0.5476, accept Ho
	Fault Tolerance Metrics					
	Velocity	Three Workers (n=3)		Six Workers (n=6)		Mann Whitney U Test Results
	2 records per time	Single %	Multi %	Single %	Multi %	3 Workers and 3 Workers (Single and Multi)
	1 min	10.08	3.58	4.97	5.02	p-value = 0.3429, accept Ho
	1 sec	7.11	5.92	6.25	8.30	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	11.09	14.83	8.44	7.17	p-value = 0.3429, accept Ho (S,S)
	10 ms	22.68	8.83	16.55	16.90	p-value = 0.4857, accept Ho (S,M) p-value = 0.8857, accept Ho (M,S) p-value = 0.8857, accept Ho (M,M)
	1 ms	NA	NA	NA	NA	6 Workers and 6 Workers (Single and Multi)

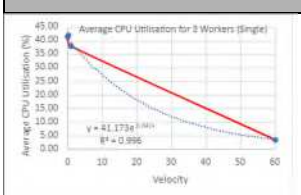
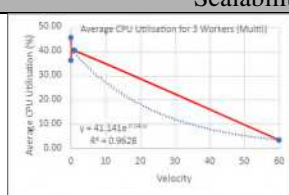
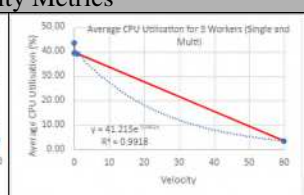
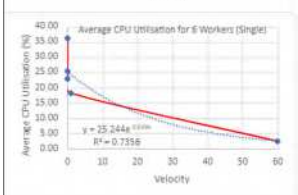
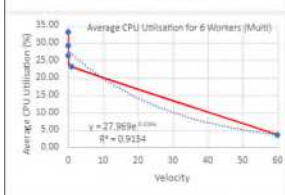
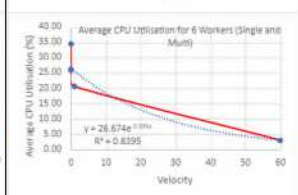
	Average	12.74		8.29		9.05		9.35		p-value = 0.8857, accept Ho	
	Technology Agnosticism Metrics (for Beam and Flink SDK)										
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results	
	2 records/time	Beam		Flink		Beam		Flink		Beam	Flink
		S%	M%	S%	M%	S%	M%	S%	M%	3 Workers and 3 Workers (Single and Multi)	
	1 min	3.4	2.9	3.4	1.9	1.5	2.6	2.1	2.6	p-value = 0.6905, accept Ho	p-value = 0.4633, accept Ho
	1 sec	33.8	27.9	36.5	34.3	19.7	17.3	20.0	16.6	3 Workers and 6 Workers ([S,S] x [M,M])	
	100 ms	44.7	36.5	38.5	36.1	23.2	23.2	18.7	20.2	p-value = 0.1425, accept Ho (S,S) p-value = 0.1508, accept Ho (S,M)	p-value = 0.09524 , accept Ho (S,S) p-value = 0.09524, accept Ho (S,M)
	10 ms	46.4	35.9	37.4	36.8	23.2	29.0	18.6	20.8	p-value = 0.1425, accept Ho (M,S) p-value = 0.4206, accept Ho (M,M)	p-value = 0.1508, accept Ho (M,S) p-value =0.1508, accept Ho (M,M)
	1 ms	64.8	66.7	35.2	36.5	35.6	37.2	17.7	18.4	6 Workers and 6 Workers (Single and Multi)	
	Average	36.82	33.98	30.20	29.12	20.64	21.86	15.42	15.72	p-value = 0.8325, accept Ho	p-value = 0.8413 , accept Ho

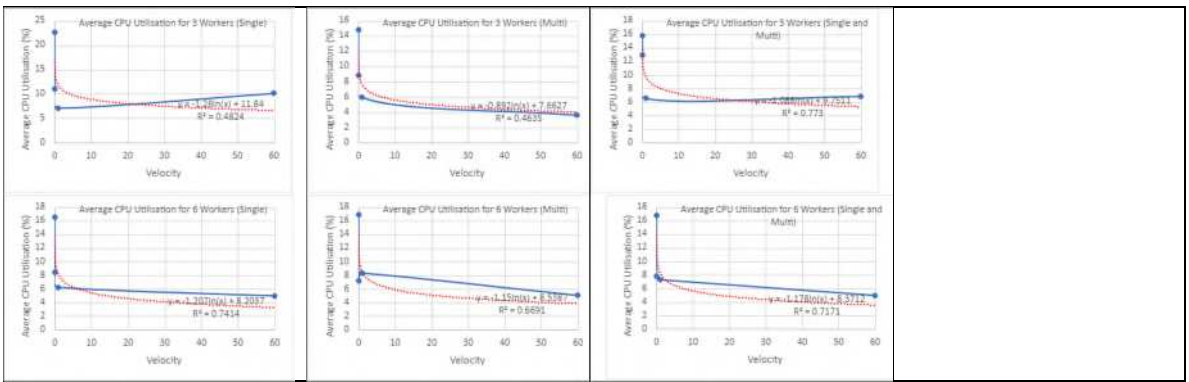
Average of Container CPU Utilisation by Windowing Rate (Period/ Duration)	Windowing Rate versus Resource Utilisation Metrics				
	Windowing Rate (Period/Duration)	Three Workers (n=3) (%)	Six Workers (n=6) (%)	Ten Workers (n=10) (%)	Mann Whitney U Test Results
	0.1	80		29	3 Workers and 6 Workers
	0.2	39	19	16	p-value = 0.4206, accept Ho
	1.0	16	9	6	3 Workers and 10 Workers
	1.5	10	6	6	p-value = 0.1719, accept Ho
	2.0	8	7	5	6 Workers and 10 Workers
	Average	30.60	10.25	12.40	p-value = 0.3976, accept Ho

Average node CPU utilisation by node Cluster	Container Co-Location Metrics			
		Cluster (n = 1) (%)	Cluster (n = 2) (%)	Cluster (n = 4) (%)
		13.39	15.96	14.47

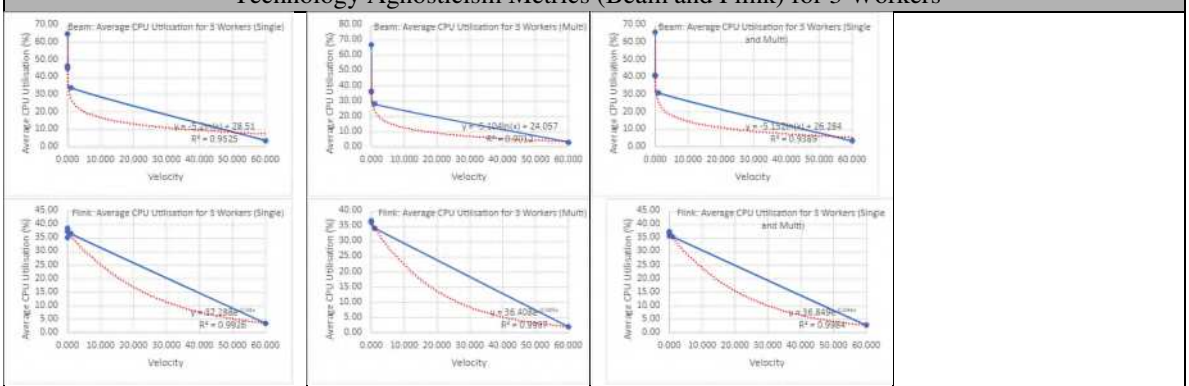
Results (in the form graphs and regression models) for average CPU utilisation by velocity, windowing rate and cluster are tabulated in Table 2. The velocity is measured in, 2 records per time, where the time units in Table 1 have been converted to seconds (0.001, 0.01, 0.1, 1.0, 60.0). Due to the inconsistent time interval, a linear trend is not appropriate. Thus, a more appropriate model would either be an exponential (when initial decrease is gradual followed by a sharp decrease), logarithmic (when initial decrease is sharp followed by a gradual decrease), power, or polynomial regression model. Note that the type of regression model has been chosen based on the highest R-squared value. Graphical displays for all the regression models cannot be included in this paper due to space constraints.

Table 2
Summary of Regression Models for Average CPU Utilisation

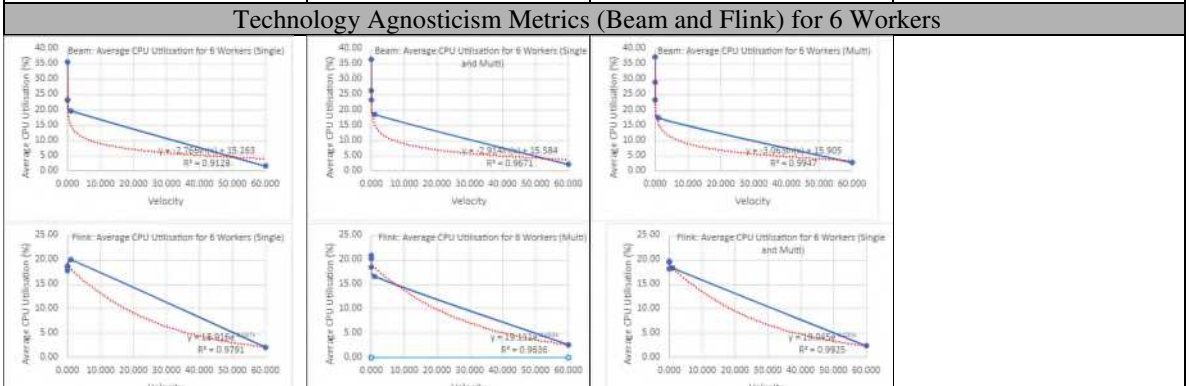
Average CPU Utilisation by Velocity		
Scalability Metrics		
		
		
Fault Tolerance Metrics		



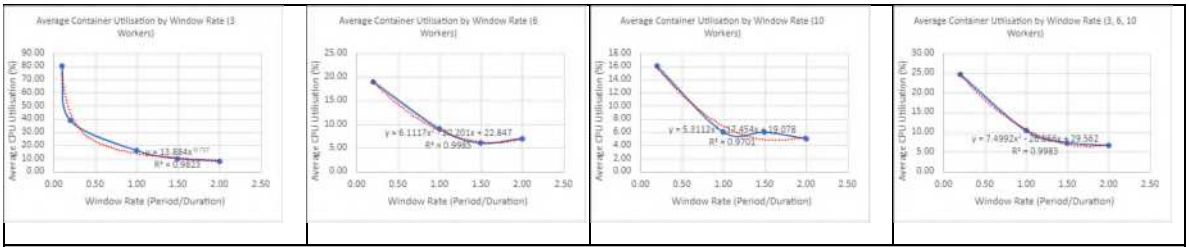
Technology Agnosticism Metrics (Beam and Flink) for 3 Workers



Technology Agnosticism Metrics (Beam and Flink) for 6 Workers

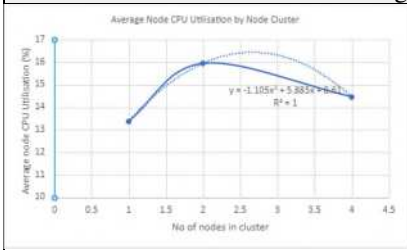


Windowing Rate versus Resource Utilisation Metrics



Container Co-location Metrics

Average Node CPU Utilisation by Node Cluster



Note: Velocity is 2 records per time (0.001s, 0.01s, 0.1s, 1.0s, 60.0s)

4.2.2. Maximum Container CPU Utilisation

Based on the summary results for maximum CPU utilisation presented in Table 3, the following conclusions were drawn: (i) maximum container CPU utilisation by velocity for scalability – means of three or six workers (single and multi) are not significantly different from each other. The same conclusion was drawn for all the possible combinations of three workers (single and multi) compared to six workers (single and multi); (ii) maximum container CPU utilisation by velocity for fault tolerance – same conclusions for all combinations in (i); (iii) maximum container CPU utilisation by velocity for technology agnosticism – all means for three or six workers (Beam, Flink) x (single, multi) were compared and found not to be significantly different from each other. The same conclusion was drawn for three workers [(Beam, Flink) x (single, multi)] and six workers [(Beam, Flink) x (single, multi)]. In terms of number of workers (3, 6, 10), the means were also not significantly different from each other for maximum container CPU utilisation by windowing rate.

Table 3
Experimental Results for the Evaluation of MC-BDP Reference Architecture in Terms of Maximum CPU Utilisation

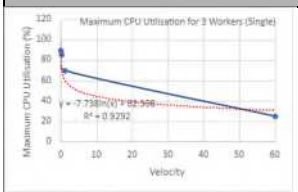
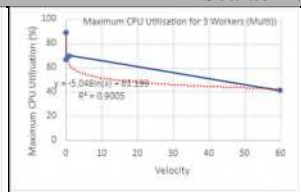
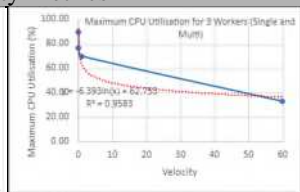
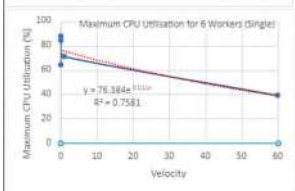
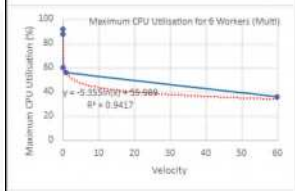
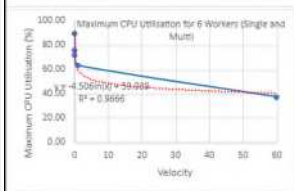
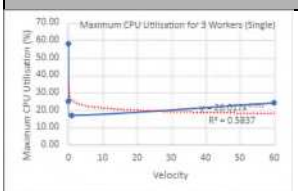
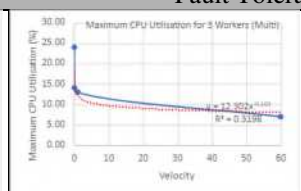
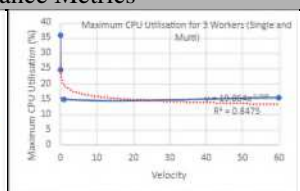
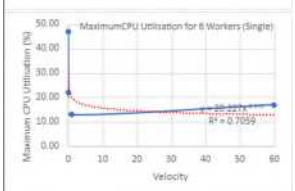
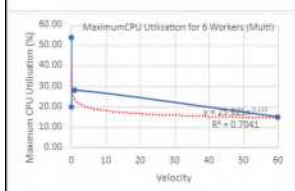
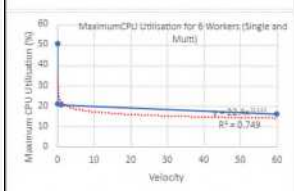
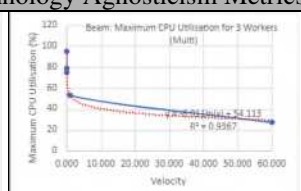
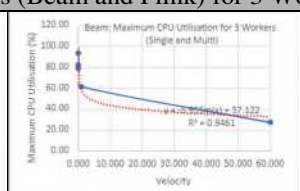
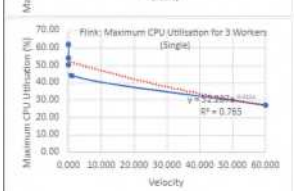
Maximum Container CPU Utilisation By Velocity	Scalability Metrics					
	Velocity	Three Workers (n=3)		Six Workers (n=6)		Mann Whitney U Test Results
		Single %	Multi %	Single %	Multi %	3 Workers and 3 Workers (Single and Multi)
	1 min	25	41	39	36	p-value = 0.7715 , accept Ho
	1 sec	70	70	71	56	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	86	67	84	60	p-value = 0.9048 , accept Ho (S,S)
	10 ms	90	89	64	88	p-value = 1.0952, accept Ho (S,M) p-value = 1.0952, accept Ho (M,S) p-value = 0.9048, accept Ho (M,M)
	1 ms	NA	NA	88	92	6 Workers and 6 Workers (Single and Multi)
	Average	67.75	66.75	64.50	60.00	p-value = 0.9166 , accept Ho
	Fault Tolerance Metrics					
	Velocity	Three Workers (n=3)		Six Workers (n=6)		Mann Whitney U Test Results

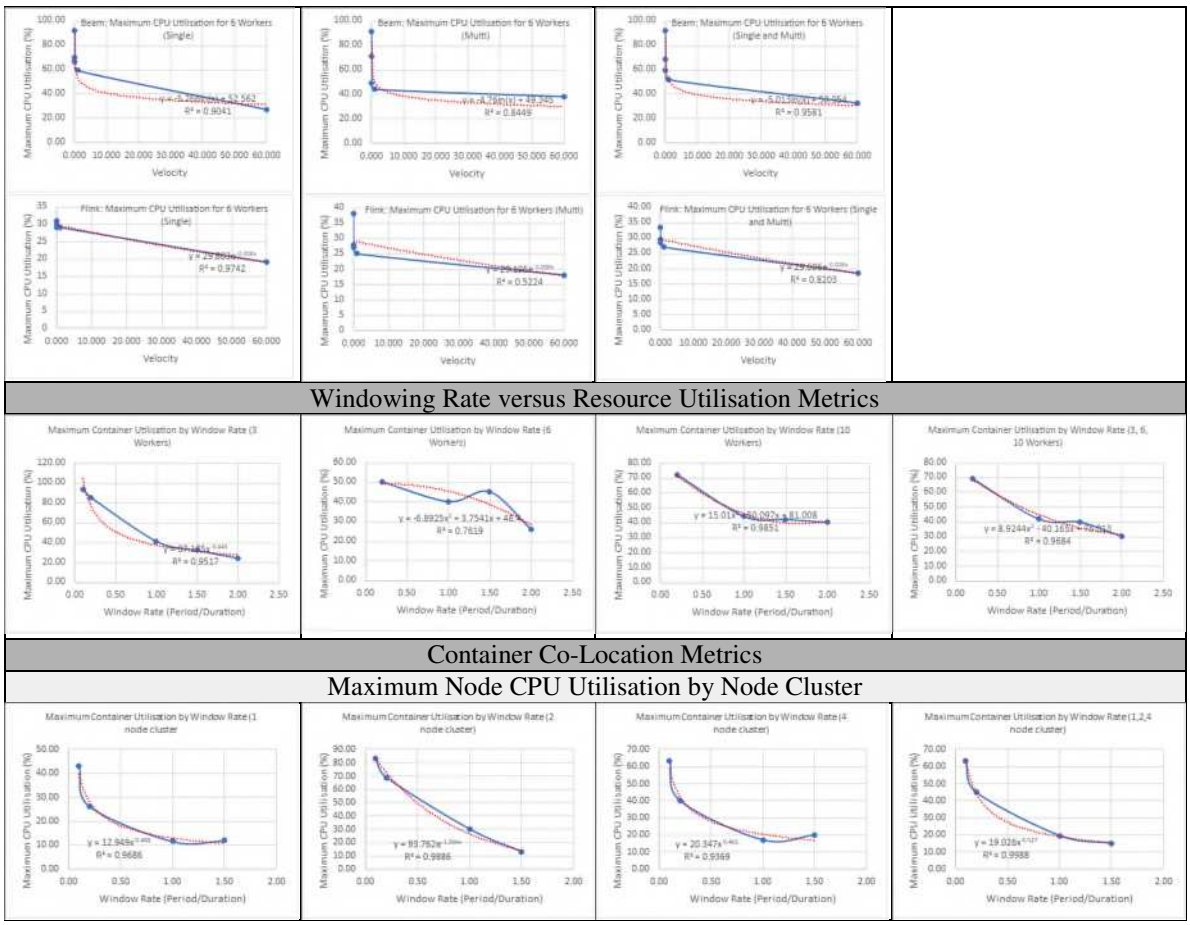
		Single %		Multi %		Single %		Multi %		3 Workers and 3 Workers (Single and Multi)	
	1 min	24		7		17		15		p-value = 0.08143, accept Ho	
	1 sec	17		13		13		28		3 Workers and 6 Workers ([S,S] x [M,M])	
	100 ms	25		24		22		20		p-value = 0.3836, accept Ho (S,S)	
	10 ms	58		14		47		54		p-value = 0.8857 , accept Ho (S,M) p-value = 0.3836, accept Ho (M,S) p-value = 0.1143, accept Ho (M,M)	
	1 ms	NA		NA		NA		NA		6 Workers and 6 Workers (Single and Multi)	
	Average	31.00		14.50		24.75		29.25		p-value = 0.6857 , accept Ho	
	Technology Agnosticism Metrics (for Beam and Flink SDK)										
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results	
		Beam		Flink		Beam		Flink		Beam	Flink
		S%	M%	S%	M%	S%	M%	S%	M%	3 Workers and 3 Workers (Single and Multi)	
	1 min	27	27	27	20	27	38	19	18	p-value = 0.8335, accept Ho	p-value = 0.5476, accept Ho
	1 sec	70	53	44	41	59	44	29	25	3 Workers and 6 Workers ([S,S] x [M,M])	
	100 ms	79	79	62	47	70	49	30	27	p-value = 0.4606, accept Ho (S,S)	p-value = 0.09369, accept Ho (S,S)
	10 ms	90	75	54	55	66	71	29	38	p-value = 0.5476 , accept Ho (S,M) p-value = 0.7533 , accept Ho (M,S) p-value = 0.5476, accept Ho (M,M)	p-value = 0.04653, accept Ho (S,M) p-value = 0.09369, accept Ho (M,S) p-value = , accept Ho (M,M)
	1 ms	92	95	50	46	92	91	31	28	6 Workers and 6 Workers (Single and Multi)	
	Average	71.60	65.80	47.40	41.80	62.80	58.60	27.60	27.20	p-value = 0.8413, accept Ho	p-value = 0.09524, accept Ho
Maximum Container CPU Utilisation by Windowing Rate (Period/ Duration)	Windowing Rate versus Resource Utilisation Metrics										
	Windowing Rate (Period/Duration)	Three Workers (n=3) (%)				Six Workers (n=6) (%)				Ten Workers (n=10) (%)	Mann Whitney U Test Results
	0.1	93				92				86	3 Workers and 6 Workers
	0.2	85				50				72	p-value = 1.0000, accept Ho
	1.0	41				40				44	3 Workers and 10 Workers
	1.5	32				45				42	p-value = 0.6905, accept Ho
	2.0	24				26				40	6 Workers and 10 Workers
	Average	55.00				50.60				56.80	p-value = 0.9166, accept Ho
Maximum node CPU utilisation by node Cluster	Container Co-Location Metrics										
		Cluster (n = 1) (%)				Cluster (n = 2) (%)				Cluster (n = 4) (%)	
		43.00				83.00				64.00	
Maximum node CPU utilisation by Windowing Rate and node Cluster	Windowing Rate (Period/Duration)	Cluster (n = 1) (%)				Cluster (n = 2) (%)				Cluster (n = 4) (%)	Mann Whitney U Test Results
	0.1	43.00				83.00				63.50	3 Workers and 6 Workers
	0.2	26.17				69.00				40.00	p-value = 0.2000, accept Ho
	1.0	11.58				30.00				17.00	3 Workers and 10 Workers
	1.5	11.93				13.00				20.00	p-value = 0.4857, accept Ho
	Average	23.17				48.75				35.13	6 Workers and 10 Workers
											p-value = 0.6857, accept Ho

Note: NA means data is not available

Results (in the form graphs and regression models) for maximum CPU utilisation by velocity, windowing rate and cluster are tabulated in Table 4.

Table 4
Experimental Results for the Evaluation of MC-BDP Reference Architecture in Terms of Maximum CPU Utilisation

Maximum CPU Utilisation by Velocity			
Scalability Metrics			
			
			
Fault Tolerance Metrics			
			
			
Technology Agnosticism Metrics (Beam and Flink) for 3 Workers			
			
			
Technology Agnosticism Metrics (Beam and Flink) for 6 Workers			



4.2.3. Average Container Memory Utilisation

Based on the summary results for average memory utilisation presented in Table 5, the following conclusions were drawn: (i) average container memory utilisation by velocity for scalability – the means of three or six workers (single and multi) are not significantly different from each other. The same conclusion was drawn for all the possible combinations of three workers (single and multi) x six workers (single and multi); (ii) average memory utilisation by velocity for fault tolerance – same conclusions for all combinations in (i); (iii) average memory utilisation by velocity for technology agnosticism – all means for three or six workers (Beam, Blink) x (single, multi) were found not to be significantly different from each other. The same conclusion was drawn for three workers [(Beam, Flink) x (single, multi)] and six workers [(Beam, Flink) x (single, multi)]. In terms of number of workers (3, 6, 10), the means were also not significantly different from each other for average container memory utilisation by windowing rate.

Table 5
Experimental Results for the Evaluation of MC-BDP Reference Architecture in Terms of Average Memory Utilisation

Average Container Memory Utilisation (MB) By Velocity	Scalability Metrics									
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results
		Single		Multi		Single		Multi		3 Workers and 3 Workers (Single and Multi)
	1 min	824		865		790		812		p-value = 0.2000, accept Ho
	1 sec	955		976		933		1015		3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	957		995		941		992		p-value = 0.4217, accept Ho (S,S)
	10 ms	968		1002		942		998		p-value = 0.1905 , accept Ho (S,M) p-value = 0.1905, accept Ho (M,S) p-value = 0.5556, accept Ho (M,M)
	1 ms	NA		NA		964		1058		6 Workers and 6 Workers (Single and Multi)
	Average	926.00		959.50		901.50		954.25		p-value = 0.09524, accept Ho
	Fault Tolerance Metrics									
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results
		Single		Multi		Single		Multi		3 Workers and 3 Workers (Single and Multi)
	1 min	837		758		882		684		p-value = 0.6857, accept Ho
	1 sec	839		842		717		758		3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	833		829		879		776		p-value = 0.2000, accept Ho (S,S)
	10 ms	699		849		897		904		p-value = 0.05714, accept Ho (S,M) p-value = 0.3429 , accept Ho (M,S) p-value = , accept Ho (M,M)
	1 ms	NA		NA		NA		NA		6 Workers and 6 Workers (Single and Multi)
	Average	802.00		819.50		843.75		780.50		p-value = 0.5614 , accept Ho
	Technology Agnosticism Metrics (Azure)									
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results
		Beam		Flink		Beam		Flink		Beam / Flink
		S	M	S	M	S	M	S	M	3 Workers and 3 Workers (Single and Multi)
	1 min	869	861	914	911	783	764	880	876	p-value = 1.0000, accept Ho / p-value = 1.0000, accept Ho
	1 sec	965	971	894	927	941	949	929	903	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	957	978	926	909	931	913	926	912	p-value = 0.09524 , accept Ho (S,S) / p-value = 0.8335 , accept Ho (S,S)
	10 ms	959	955	886	910	948	899	914	891	p-value = 0.09524, accept Ho (S,M) / p-value = 0.4206, accept Ho (S,M) p-value = 0.09524, accept Ho (M,S) / p-value = 0.9163, accept Ho (M,S) p-value = 0.1508, accept Ho (M,M) / p-value = 0.402 , accept Ho (M,M)
	1 ms	1000	987	924	911	946	956	909	916	6 Workers and 6 Workers (Single and Multi)
	Average	950.0	950.4	908.8	913.6	909.8	896.2	911.6	899.6	p-value = 1.0000, accept Ho / p-value = 0.3095 , accept Ho

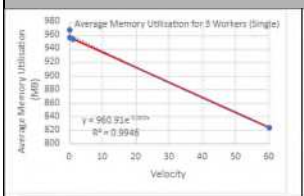
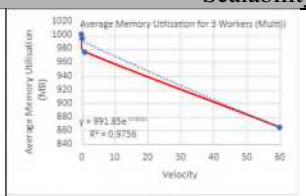
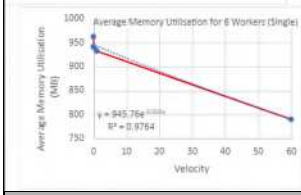
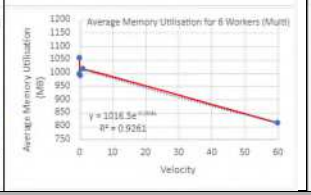
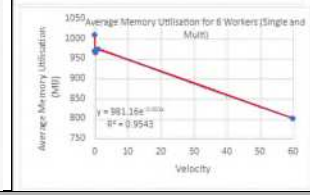
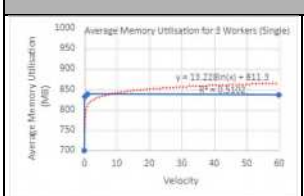
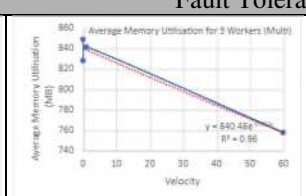
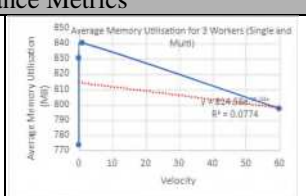
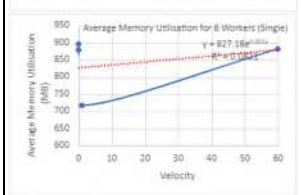
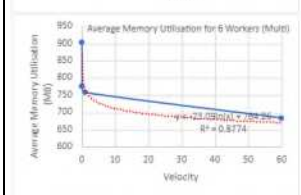
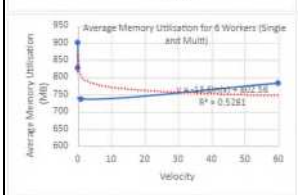
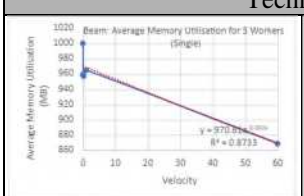
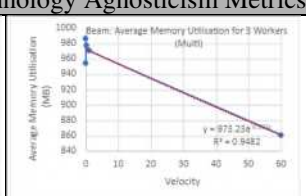
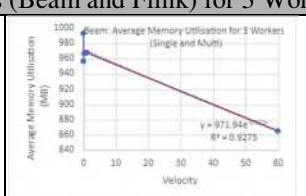
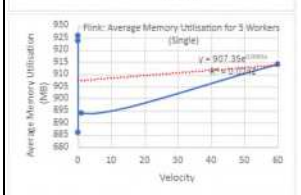
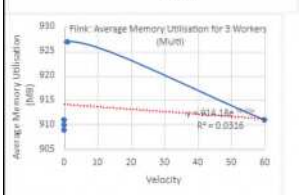
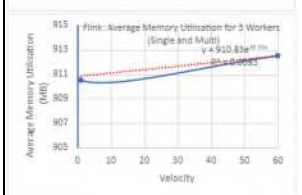
Average Container Memory Utilisation by Windowing Rate (Period/ Duration)	Windowing Rate versus Resource Utilisation Metrics				
	Windowing Rate (Period/Duration)	Three Workers (n=3) (MB)	Six Workers (n=6) (MB)	Ten Workers (n=10) (MB)	Mann Whitney U Test Results
	0.1	1008	986	980	3 Workers and 6 Workers
	0.2	974	1002	988	p-value = 0.5476, accept Ho
	1.0	947	962	903	3 Workers and 10 Workers
	1.5	968	943	892	p-value = 0.4206, accept Ho
	2.0	972	910	815	6 Workers and 10 Workers
	Average	973.8	960.6	915.6	p-value = 0.3095, accept Ho

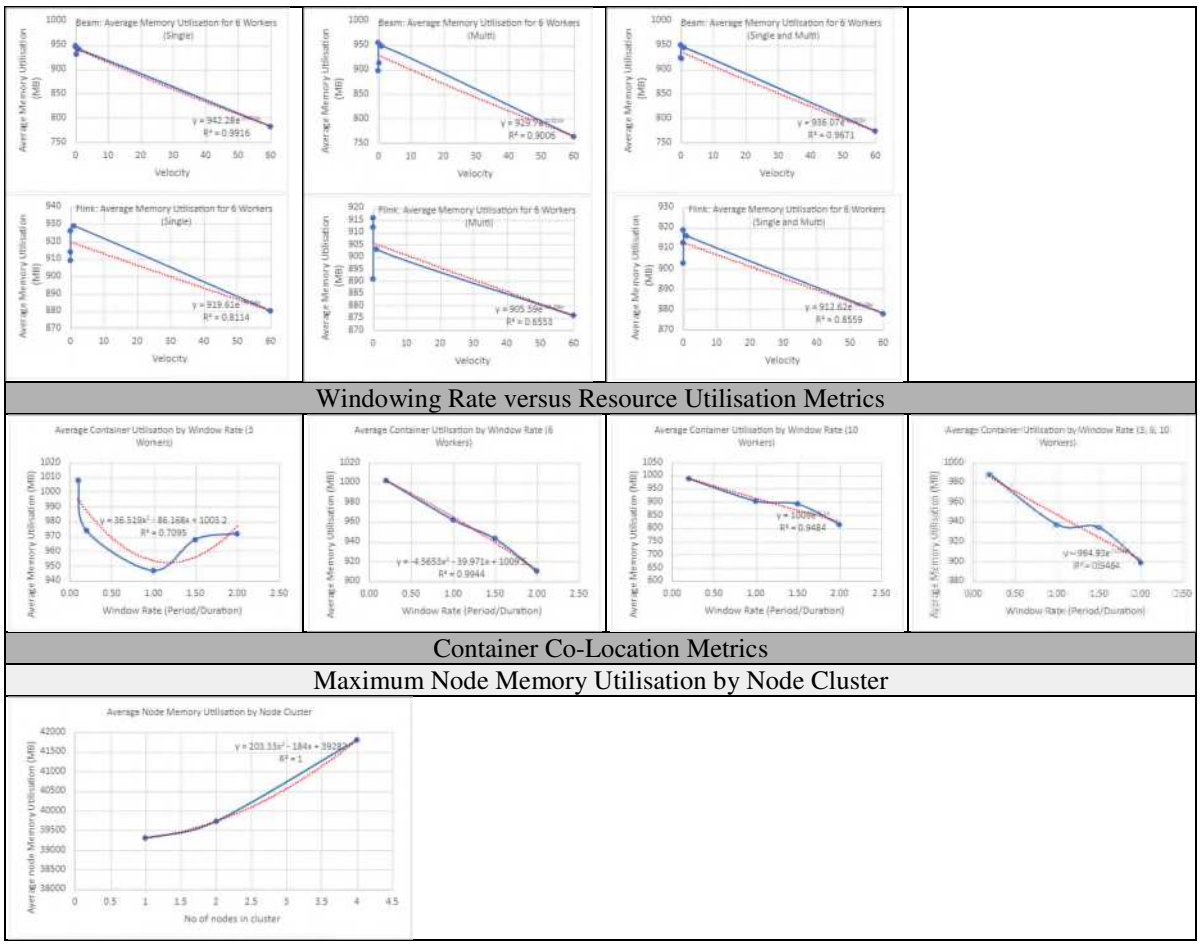
Average memory utilisation by node Cluster	Container Co-Location Metrics			
		Cluster (n = 1) (MB)	Cluster (n = 2) (MB)	Cluster (n = 4) (MB)
		39301	39727	41799

Note: NA means data is not available

Results (in the form graphs and regression models) for average memory utilisation by velocity, windowing rate and cluster are tabulated in Table 6.

Table 6
Average Memory Utilisation Graphs and Regression Models

Average Memory Utilisation by Velocity			
Scalability Metrics			
			
			
Fault Tolerance Metrics			
			
			
Technology Agnosticism Metrics (Beam and Flink) for 3 Workers			
			
			
Technology Agnosticism Metrics (Beam and Flink) for 6 Workers			



4.2.4 Maximum Container Memory Utilisation

Based on the summary results for maximum memory utilisation presented in Table 7, the following conclusions were drawn: (i) maximum container memory utilisation by velocity for scalability – means of three or six workers (single and multi) were not significantly different from each other. The same conclusion was drawn for all the possible combinations of three workers (single and multi) x six workers (single and multi); (ii) maximum container memory utilisation by velocity for fault tolerance – same conclusions for all combinations in (i); (iii) maximum container memory utilisation by velocity for technology agnosticism – all means for three or six workers (Beam, Flink) x (single, multi) are not significantly different from each other. The same conclusion was drawn for three workers [(Beam, Flink) x (single, multi)], and six workers [(Beam, Flink) x (single, multi)], except for 3 workers (Beam, multi) and 6 workers (Beam, multi). Maximum memory utilisation was therefore slightly higher in multi-cloud clusters than it was in single-cloud clusters in the Beam SDK setups The reason for this discrepancy has not yet been established, and it may be related to the use of the Beam SDK and not to the infrastructure being single or multi, since the same was not observed when

using the Flink SDK. It is believed that repeated runs of the experiment over time would provide more relevant data to better understand the relationship between the cloud provider selected, the SDK selected, and the maximum memory utilisation metrics gathered. The means of different numbers of workers (3, 6, 10) are not significantly different from each other for maximum container memory utilisation by windowing rate.

Table 7
Experimental Results for the Evaluation of MC-BDP Reference Architecture in Terms of Maximum Memory Utilisation

Maximum Container Memory Utilisation (MB) By Velocity	Scalability Metrics									
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results
		Single		Multi		Single		Multi		3 Workers and 3 Workers (Single and Multi)
	1 min	947		987		946		934		p-value = 0.02857, accept Ho
	1 sec	959		983		939		1015		3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	969		1004		947		1025		p-value = 0.4606, accept Ho (S,S)
	10 ms	977		1022		967		1037		p-value = 0.1905, accept Ho (S,M) p-value = 0.06349, accept Ho (M,S) p-value = 0.2957 , accept Ho (M,M)
	1 ms	NA		NA		990		1083		6 Workers and 6 Workers (Single and Multi)
	Average	963.00		999.00		949.75		1002.75		p-value = 0.1508 , accept Ho
	Fault Tolerance Metrics									
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results
		Single		Multi		Single		Multi		3 Workers and 3 Workers (Single and Multi)
	1 min	906		831		764		815		p-value = 0.1143 , accept Ho
	1 sec	897		879		762		847		3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	928		911		764		872		p-value = 0.0294, accept Ho (S,S)
	10 ms	1003		876		825		1021		p-value = 0.3429 , accept Ho (S,M) p-value = 0.0294 , accept Ho (M,S) p-value = 0.6857, accept Ho (M,M)
	1 ms	NA		NA		NA		NA		6 Workers and 6 Workers (Single and Multi)
	Average	933.50		874.25		778.75		888.75		p-value = 0.05907 , accept Ho
	Technology Agnosticism Metrics (Azure)									
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results
		Beam		Flink		Beam		Flink		Beam / Flink
		S	M	S	M	S	M	S	M	3 Workers and 3 Workers (Single and Multi)
	1 min	974	1012	934	923	961	953	935	909	p-value = 0.5476, accept Ho / p-value = 0.7533, accept Ho
	1 sec	1006	1014	909	956	993	968	966	914	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	995	1001	941	939	973	969	961	935	p-value = 0.2222, accept Ho (S,S) / p-value = 0.3457, accept Ho (S,S) p-value = 0.05556, accept Ho (S,M) / p-value = 0.7533, accept Ho (S,M) p-value = 0.05556 , accept Ho (M,S) / p-value = 0.3457 , accept Ho (M,S)
	10 ms	992	982	900	934	1000	973	930	946	p-value = 0.01587, reject Ho (M,M) / p-value = 0.7533, accept Ho (M,M)
	1 ms	1079	1064	962	935	991	999	941	944	6 Workers and 6 Workers (Single and Multi)
	Average	1009	1014	929	941	983	972	946	929	p-value = 0.3457, accept Ho / p-value = 0.3457 , accept Ho
Maximum Container Memory Utilisation by Windowing Rate (Period/ Duration)	Windowing Rate versus Resource Utilisation Metrics									
	Windowing Rate (Period/Duration)			Three Workers (n=3) (MB)			Six Workers (n=6) (MB)		Ten Workers (n=10) (MB)	Mann Whitney U Test Results
	0.1			10176			1011		991	3 Workers and 6 Workers
	0.2			994			1003		1011	p-value = 0.1412, accept Ho
	1.0			999			1010		1005	3 Workers and 10 Workers
	1.5			986			1003		983	p-value = 0.5476 , accept Ho
	2.0			998			1011		952	6 Workers and 10 Workers
	Average			2830			1007		988	p-value = 0.2031, accept Ho
Maximum node Memory utilisation	Windowing Rate (Period/Duration)			Cluster (n = 1) (MB)			Cluster (n = 2) (MB)		Cluster (n = 4) (MB)	Mann Whitney U Test Results

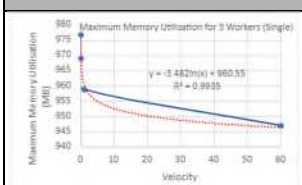
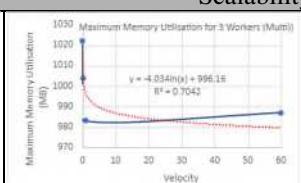
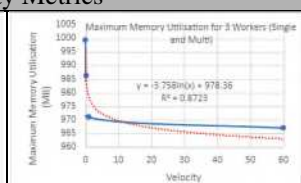
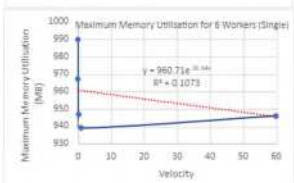
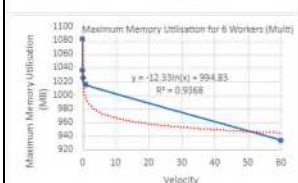
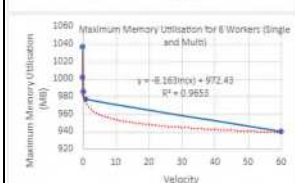
by Windowing Rate and node Cluster	0.1	1050	2161	4474	3 Workers and 6 Workers
	0.2	1085	2186	4517	p-value = 0.01193, reject Ho
	1.0	1043	2174	4475	3 Workers and 10 Workers
	1.5	1039	2171	4475	p-value = 0.01193, reject Ho
	2.0	1078	2161	4441	6 Workers and 10 Workers
	Average	1059	2170	4476	p-value = 0.01167, reject Ho
Maximum node Memory utilisation by node Cluster	Container Co-Location Metrics				
		Cluster (n = 1) (MB)	Cluster (n = 2) (MB)	Cluster (n = 4) (MB)	
		8091	8537	8824	

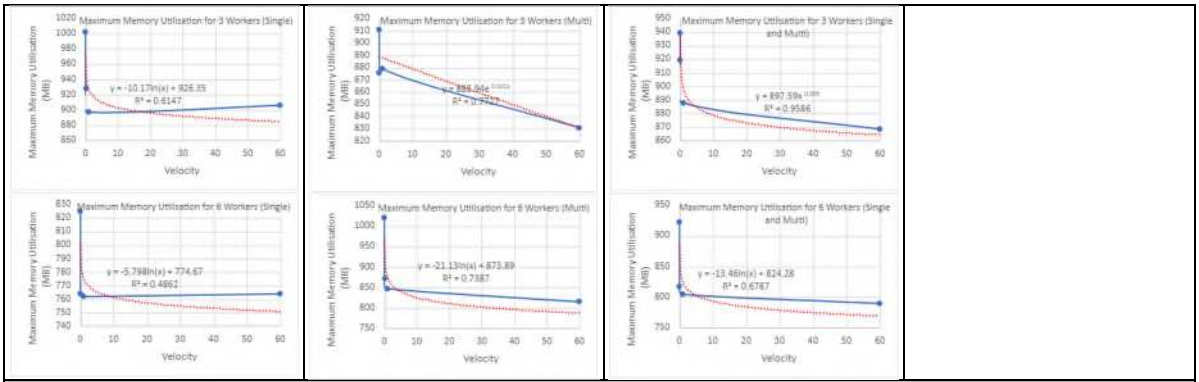
Note: NA means data is not available

The means for maximum node memory utilisation by windowing rate and node cluster sizes (1,2,4) were significantly different from each other, denoting a clear difference between the three co-locations observed. The nodes with two co-located containers had a maximum reading roughly twice as high as the nodes with no co-locater containers. Similarly, the nodes with four co-located containers had a maximum reading roughly twice as high as the nodes with two co-located containers. This was to be expected, however, since nodes with no co-located containers were in a cluster of eight, nodes with two co-located containers were in a cluster of four, and nodes with no co-located containers were in a cluster of two. In order to compare memory utilisation and co-location, it was necessary to take the entire cluster into consideration and aggregate the results. The Cluster Memory Utilisation (max) formula, depicted in Fig. 4 and explained in Section 3.2, was utilised in order to obtain maximum memory utilisation results for the cluster. The last three rows of Table 7 summarise the maximum memory utilisation findings from examining the entire cluster.

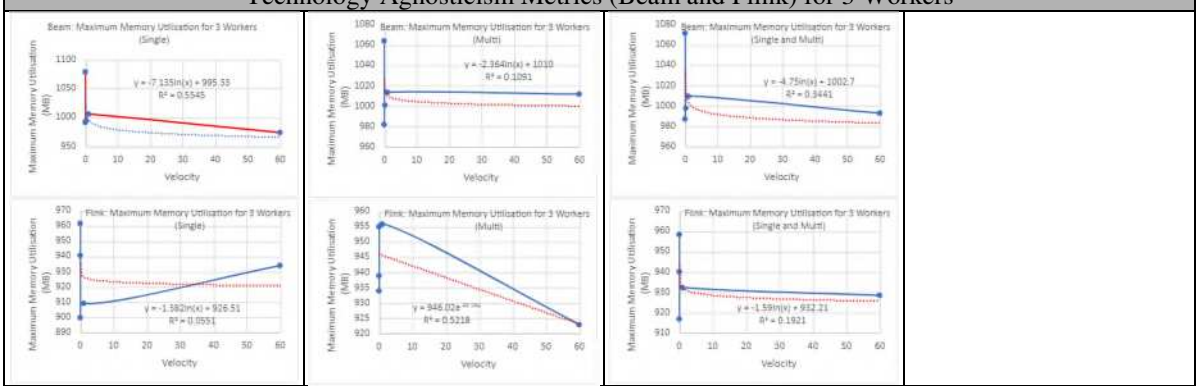
Results (in the form graphs and regression models) for maximum memory utilisation by velocity, windowing rate and cluster are tabulated in Table 8.

Table 8
Maximum Memory Utilisation Graphs and Regression Models

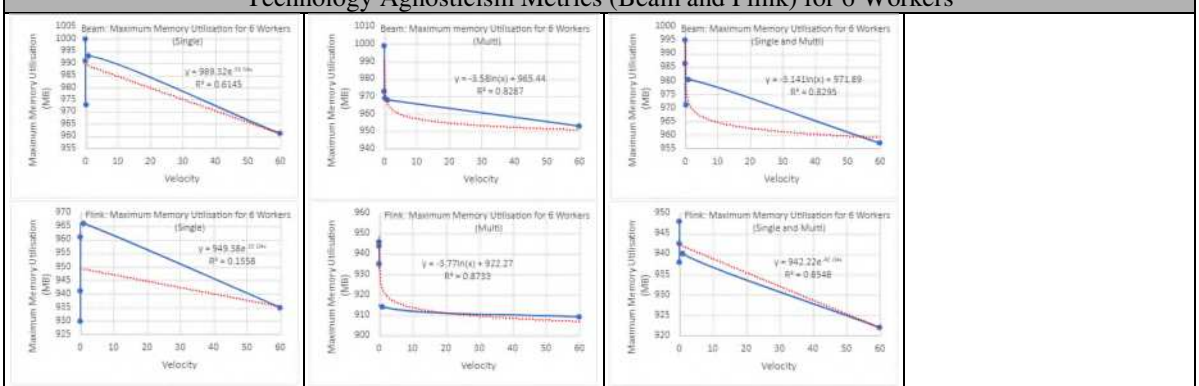
Maximum Memory Utilisation by Velocity			
Scalability Metrics			
			
			
Fault Tolerance Metrics			



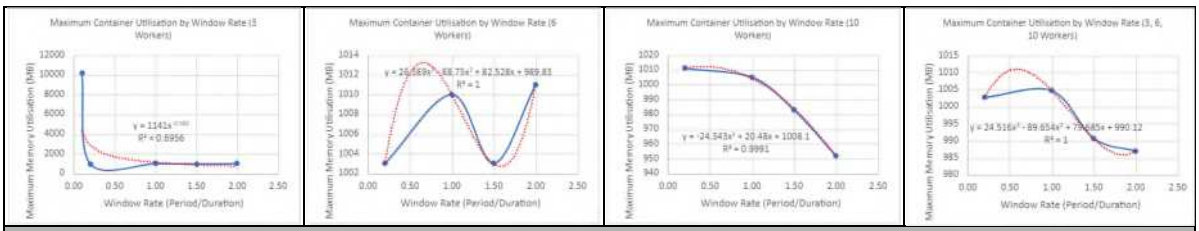
Technology Agnosticism Metrics (Beam and Flink) for 3 Workers



Technology Agnosticism Metrics (Beam and Flink) for 6 Workers

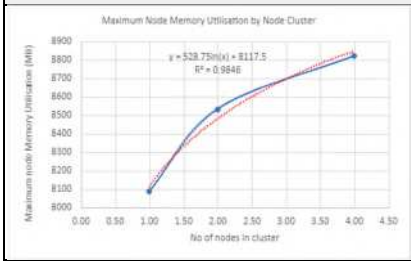


Windowing Rate versus Resource Utilisation Metrics



Container Co-Location

Maximum Node Memory Utilisation by Node Cluster Metrics



4.2.5. Container Network Utilisation (GB Sent by Workers)

Based on the summary results presented in Table 9 for total number of GB sent over the network, the following conclusions were drawn: (i) total number of GB sent over the network by velocity for scalability – means of three or six workers (single and multi) are not significantly different from each other. The same conclusion was drawn for all the possible combinations of three workers (single and multi) x six workers (single and multi); (ii) total number of GB sent over the network by velocity for fault tolerance – same conclusions for all combinations in (i); (iii) total number of GB sent over the network by velocity for technology agnosticism – all means for three or six workers (Beam, Flink) x (single, multi) are not significantly different from each other. The same conclusion is drawn for three workers [(Beam, Flink) x (single, multi)], and six workers [(Beam, Flink) x (single, multi)]. The means of number of workers (3, 6, 10) as well as number of containers per node (1, 2, 4) were also not significantly different from each other for total number of GB sent over the network by windowing rate.

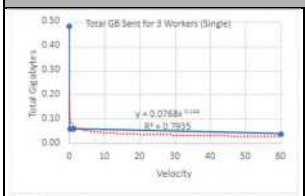
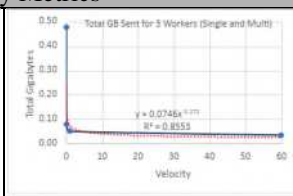
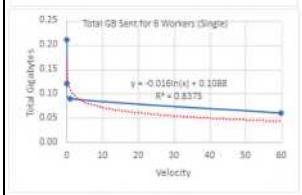
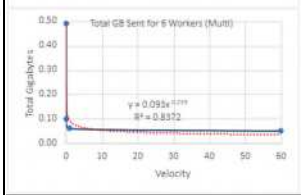
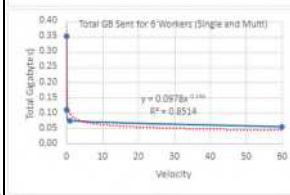
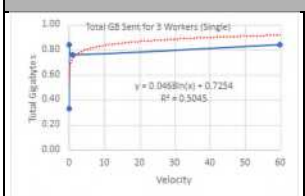
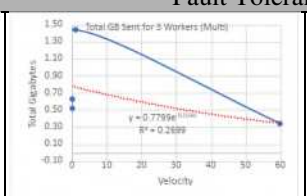
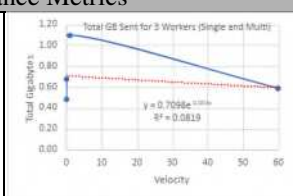
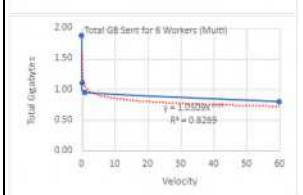
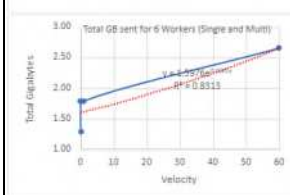
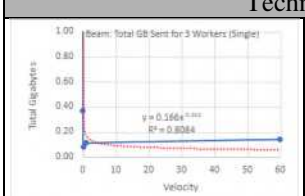
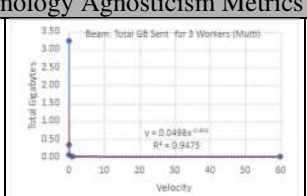
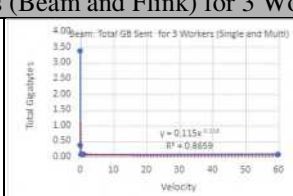
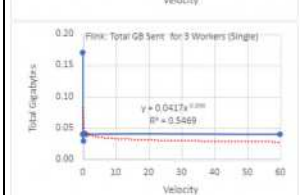
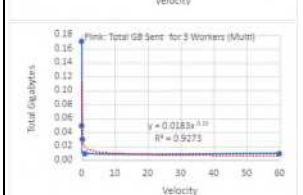
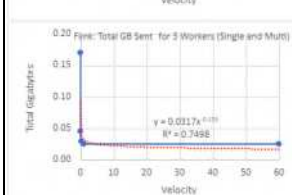
Table 9
Network Utilisation – Total Number of Gigabytes Sent over the Network

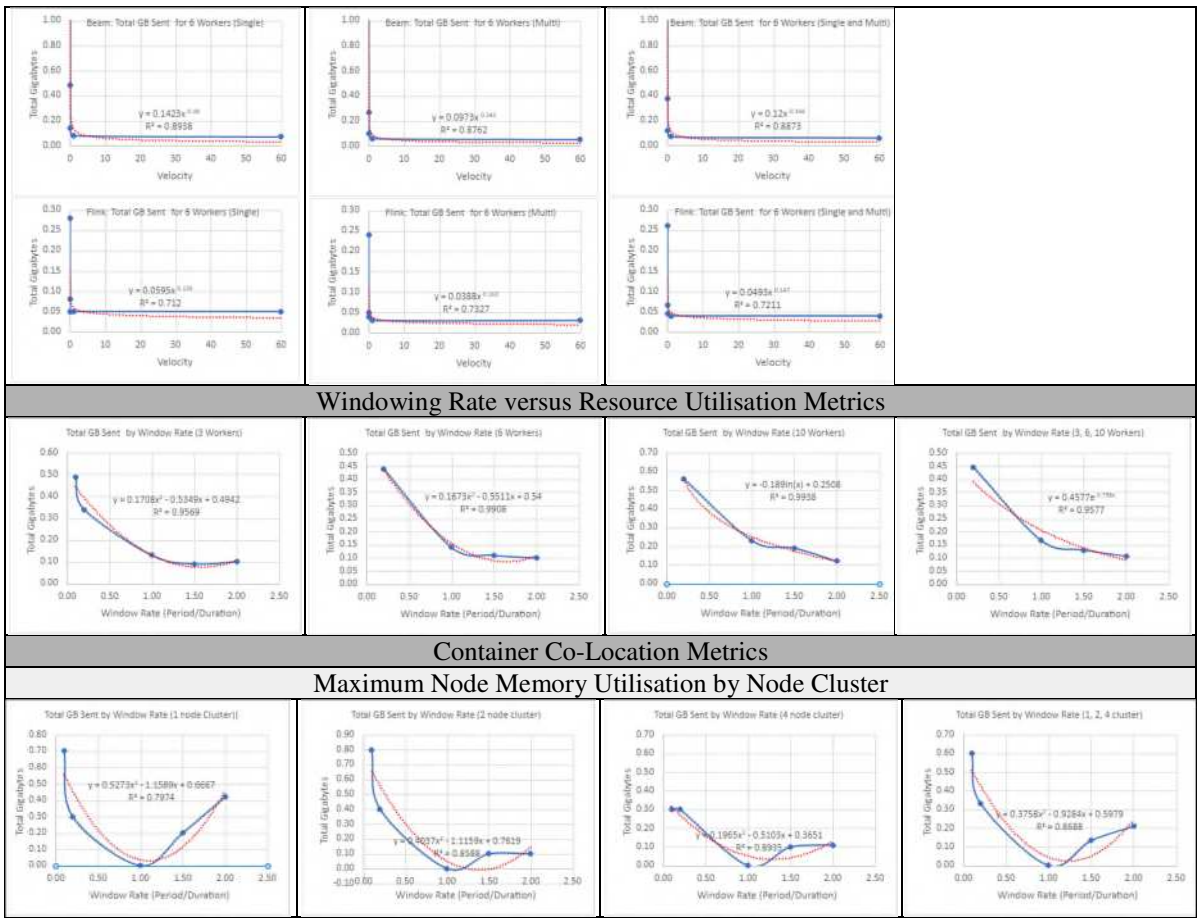
Total Number of GB Sent over the Network by Velocity	Scalability Metrics					
	Velocity	Three Workers (n=3)		Six Workers (n=6)		Mann Whitney U Test Results
		Single	Multi	Single	Multi	3 Workers and 3 Workers (Single and Multi)
	1 min	0.04	0.03	0.06	0.05	p-value = 0.7702, accept Ho
	1 sec	0.06	0.04	0.09	0.06	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	0.06	0.10	0.12	0.10	p-value = 0.3832, accept Ho (S,S)
	10 ms	0.48	0.47	0.21	0.49	p-value = 0.3832 , accept Ho (S,M) p-value = 0.5556, accept Ho (M,S) p-value = 0.3532 , accept Ho (M,M)
	1 ms	NA	NA	0.36	3.30	6 Workers and 6 Workers (Single and Multi)
	Average	0.16	0.16	0.12	0.12	p-value = 0.9166, accept Ho
	Fault Tolerance Metrics					
	Velocity	Three Workers (n=3)		Six Workers (n=6)		Mann Whitney U Test Results
		Single	Multi	Single	Multi	3 Workers and 3 Workers (Single and Multi)

	1 min	0.84		0.34		4.50		0.80	p-value = 0.8845, accept Ho
	1 sec	0.76		1.44		2.62		0.95	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	0.84		0.52		1.48		1.10	p-value = 0.0294 , accept Ho (S,S)
	10 ms	0.33		0.63		1.70		1.88	p-value = 0.1102, accept Ho (S,M) p-value = 0.02857, accept Ho (M,S) p-value = 0.2000, accept Ho (M,M)
	1 ms	NA		NA		NA		NA	6 Workers and 6 Workers (Single and Multi)
	Average	0.69		0.73		2.58		1.18	p-value = 0.1143 , accept Ho
	Technology Agnosticism Metrics (for Beam and Flink SDK)								
	Velocity	Three Workers (n=3)				Six Workers (n=6)			Mann Whitney U Test Results
		Beam		Flink		Beam		Flink	Beam / Flink
		S	M	S	M	S	M	S	M	3 Workers and 3 Workers (Single and Multi)
	1 min	0.14	0.02	0.04	0.01	0.07	0.05	0.05	0.03	p-value = 0.2948, accept Ho / p-value = 0.5219, accept Ho
	1 sec	0.11	0.02	0.04	0.01	0.08	0.06	0.05	0.03	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	0.08	0.06	0.03	0.03	0.14	0.10	0.05	0.04	p-value = 1.0000, accept Ho (S,S) / p-value = 0.8668, accept Ho (S,S)
	10 ms	0.37	0.35	0.04	0.05	0.48	0.27	0.08	0.05	p-value = 0.4206, accept Ho(S,M) / p-value = 0.9131, accept Ho (S,M) p-value = 0.2948, accept Ho(M,S) / p-value = 0.1599, accept Ho (M,S) p-value = 0.7526, accept Ho(M,M) / p-value = 0.4564, accept Ho (M,M)
	1 ms	3.52	3.25	0.17	0.17	6.26	3.02	0.28	0.24	6 Workers and 6 Workers (Single and Multi)
	Average	0.84	0.74	0.06	0.05	1.41	0.70	0.10	0.08	p-value = 0.5476, accept Ho / p-value = 0.1599, accept Ho
Total Number of GB Sent over the Network by Windowing Rate (Period/ Duration)	Windowing Rate versus Resource Utilisation Metrics								
	Windowing Rate (Period/Duration)	Three Workers (n=3)				Six Workers (n=6)		Ten Workers (n=10)	Mann Whitney U Test Results
	0.1	0.49				0.63		1.02	3 Workers and 6 Workers
	0.2	0.34				0.44		0.56	p-value = 0.6004, accept Ho
	1.0	0.13				0.14		0.23	3 Workers and 10 Workers
	1.5	0.09				0.11		0.19	p-value = 0.3095, accept Ho
	2.0	0.10				0.10		0.12	6 Workers and 10 Workers
	Average	0.23				0.28		0.42	p-value = 0.4206, accept Ho
Total Number of GB Sent over the Network by Number of Containers per Node (by Windowing Rate and Node Cluster)	Container Colocation Metrics								
	Windowing Rate (Period/Duration)	Cluster (n = 1) (GB)				Cluster (n = 2) (GB)		Cluster (n = 4) (GB)	Mann Whitney U Test Results
	0.1	0.70				0.80		0.30	3 Workers and 6 Workers
	0.2	0.30				0.40		0.30	p-value = 0.7526 , accept Ho
	1.0	0.00				0.00		0.00	3 Workers and 10 Workers
	1.5	0.20				0.10		0.10	p-value = 0.3398, accept Ho
	2.0	0.42				0.10		0.11	6 Workers and 10 Workers
	Average	0.32				0.28		0.16	p-value = 0.9153, accept Ho

Results (in the form graphs and regression models) for the number of gigabytes sent over the network by velocity, windowing rate and cluster are tabulated in Table 10.

Table 10
Network Utilisation (Gigabytes Sent over the Network) – Graphs and Regression Models

Network Utilisation: Total Number of Gigabytes Sent over the Network
Scalability Metrics
     
Fault Tolerance Metrics
     
Technology Agnosticism Metrics (Beam and Flink) for 3 Workers
     
Technology Agnosticism Metrics (Beam and Flink) for 6 Workers



4.2.6. Container Network Utilisation (GB Received by Workers)

Based on the summary results presented in Table 11 for total number of GB received over the network, the following conclusions were drawn: (i) total number of GB received over the network by velocity for scalability – means of three or six workers (single and multi) are significantly different from each other for all combinations except for three workers (single, multi). The same conclusion is drawn for almost all possible combinations of three workers (single and multi) x six workers (single and multi), except for three workers (single) x six workers (multi); (ii) total number of GB received over the network by velocity for fault tolerance –there is no significant difference in the means for all combinations, except for six workers (single x multi); (iii) total number of GB received over the network by velocity for technology agnosticism – all means for three or six workers (Beam) x (single, multi) are not significantly different from each other. There is a significant difference for three workers (Flink) (single x multi), (Flink) [three workers (single) x 6 workers (single, multi)], while the rest of the combinations show no significant difference. The means of different numbers of workers (3, 6, 10) are significantly different from each other, while the number of containers per

node (1, 2, 4) are not significantly different from each other for total number of GB received over the network by windowing rate.

Table 11
Network Utilisation (Gigabytes Received over the Network) – Graphs and Regression Models

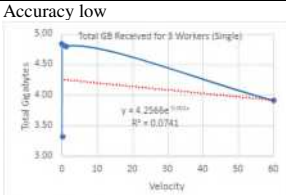
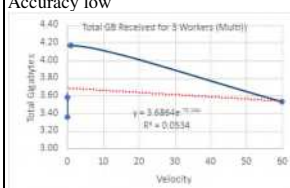
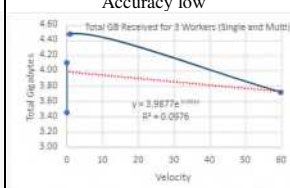
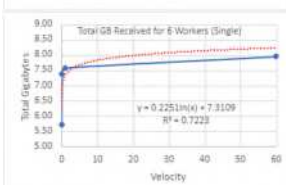
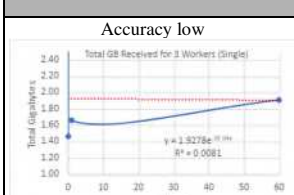
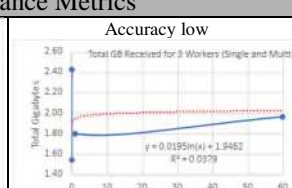
Total Number of GB Received over the Network by Velocity	Scalability Metrics									
	Velocity	Three Workers (n=3)		Six Workers (n=6)		Mann Whitney U Test Results				
		Single	Multi	Single	Multi	3 Workers and 3 Workers (Single and Multi)				
	1 min	3.91	3.53	7.94	5.17	p-value = 0.4857 , accept Ho				
	1 sec	4.80	4.17	7.57	5.18	3 Workers and 6 Workers ([S,S] x [M,M])				
	100 ms	3.32	3.58	7.38	4.84	p-value = 0.01587, reject Ho (S,S)				
	10 ms	4.84	3.36	5.72	5.73	p-value = 0.02684, accept Ho (S,M) p-value = 0.01587, reject Ho (M,S) p-value = 0.01587, reject Ho (M,M)				
	1 ms	NA	NA	9.64	12.49	6 Workers and 6 Workers (Single and Multi)				
	Average	4.22	3.66	7.15	5.98	p-value = 0.2222, reject Ho				
	Fault Tolerance Metrics									
	Velocity	Three Workers (n=3)		Six Workers (n=6)		Mann Whitney U Test Results				
		Single	Multi	Single	Multi	3 Workers and 3 Workers (Single and Multi)				
	1 min	1.91	2.02	5.52	5.58	p-value = 0.8857, accept Ho				
	1 sec	1.66	1.94	6.34	4.71	3 Workers and 6 Workers ([S,S] x [M,M])				
	100 ms	2.95	1.90	4.64	4.75	p-value = 0.02857, accept Ho (S,S)				
	10 ms	1.46	1.62	5.27	5.64	p-value = 0.02857, accept Ho (S,M) p-value = 0.02857, accept Ho (M,S) p-value = 0.02857, accept Ho (M,M)				
	1 ms	NA	NA	NA	NA	6 Workers and 6 Workers (Single and Multi)				
	Average					p-value = 0.2222, reject Ho				
	Technology Agnosticism Metrics (for Beam and Flink SDK)									
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results
		Beam		Flink		Beam		Flink		Beam / Flink
		S	M	S	M	S	M	S	M	3 Workers and 3 Workers (Single and Multi)
	1 min	4.84	2.28	1.83	1.03	8.75	7.10	4.19	3.45	p-value = 0.09269, accept Ho / p-value = 0.007937, reject Ho
	1 sec	4.84	2.28	2.25	1.09	9.16	7.23	4.73	3.27	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	4.61	2.31	2.32	1.16	9.47	6.51	4.57	2.92	p-value = 0.05933, accept Ho (S,S) / p-value = 0.007937, reject Ho (S,S)
	10 ms	5.14	4.19	2.04	1.92	10.06	7.96	4.66	3.79	p-value = 0.09369, accept Ho (S,M) / p-value = 0.01587, reject Ho (S,M) p-value = 0.03615, accept Ho (M,S) / p-value = 0.007937 , accept Ho (M,S) p-value = 0.5296, accept Ho (M,M) / p-value = 0.01587, accept Ho (M,M)
	1 ms	10.03	9.26	3.14	3.07	20.79	16.21	7.76	5.57	6 Workers and 6 Workers (Single and Multi)
	Average	5.89	4.06	2.32	1.65	11.65	9.00	5.18	3.80	p-value = 0.09524, accept Ho / p-value = 0.09524 , accept Ho
Total Number of GB Received over the Network by Windowing Rate (Period/ Duration)	Windowing Rate versus Resource Utilisation Metrics									
	Windowing Rate (Period/Duration)	Three Workers (n=3)		Six Workers (n=6)		Ten Workers (n=10)		Mann Whitney U Test Results		
	0.1	4.46		7.40		13.49		3 Workers and 6 Workers		
	0.2	4.45		7.81		13.70		p-value = 0.007937, reject Ho		
	1.0	4.39		7.57		13.69		3 Workers and 10 Workers		
	1.5	4.23		7.56		12.91		p-value = 0.007937, reject Ho		
	2.0	4.12		7.42		12.87		6 Workers and 10 Workers		
	Average	4.33		7.55		13.33		p-value = 0.007937 , reject Ho		
Total Number of GB Received over the Network by Number of Containers per Node	Container Co-Location Metrics									
	Windowing Rate (Period/Duration)	Cluster (n = 1) (GB)		Cluster (n = 2) (GB)		Cluster (n = 4) (GB)		Mann Whitney U Test Results		
	0.1	9.88		9.96		8.64		3 Workers and 6 Workers		
	0.2	8.51		8.90		8.42		p-value = 0.4206, accept Ho		

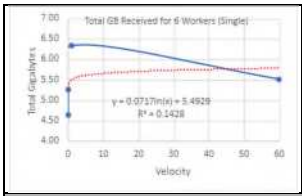
(by Windowing Rate and Node Cluster)	1.0	9.11	8.97	7.68	3 Workers and 10 Workers
	1.5	9.10	8.70	9.21	p-value = 0.1508, accept Ho
	2.0	9.15	7.91	8.18	6 Workers and 10 Workers
	Average	9.15	8.89	8.43	p-value = 0.3095, accept Ho

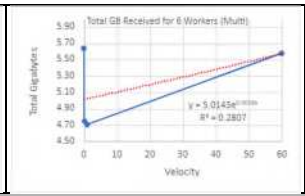
Increased network utilisation as clusters increased in size was an expected effect, since more workers mean greater parallelisation of work, which optimises performance, but increases container-to-container communication [48]. This is particularly relevant in the context of cloud computing where data transfers are charged on a metered basis. As expected, co-location did not significantly affect network utilisation, since containers were configured to use the Internet to communicate with each other. It is worth noting, however, that there are other orchestration technologies, such as Kubernetes, which allow a greater degree of resource sharing between containers deployed to the same node. Kubernetes uses the concept of a pod to group related containers running on the same machine, so containers which are part of the same pod can access each other's ports and transfer data internally without leaving the host [71]. It would be interesting to implement an alternative prototype in the future using a different orchestrator such as Kubernetes in order to gain an understanding of other ways in which the co-location of containers into the same machine could translate into reduced cloud network utilisation.

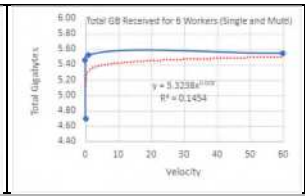
Results (in the form graphs and regression models) for the number of gigabytes received over the network by velocity, windowing rate and cluster are tabulated in Table 12.

Table 12
Network Utilisation (Gigabytes Received over the Network) – Graphs and Regression Models

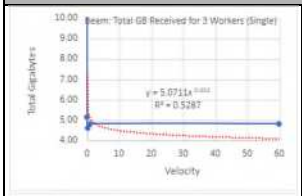
Network Utilisation: Total Number of Gigabytes Received Over the Network			
Scalability Metrics			
Accuracy low	Accuracy low	Accuracy low	
			
	Accuracy low 		
Fault Tolerance Metrics			
Accuracy low		Accuracy low	
			

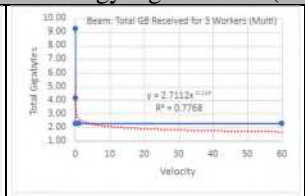


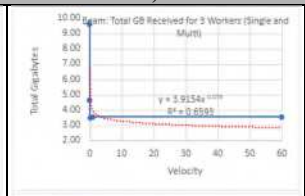


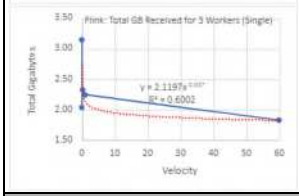


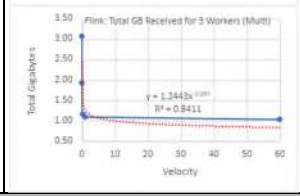
Technology Agnosticism (Beam and Flink) for 3 Workers

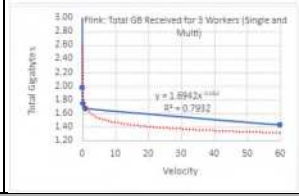




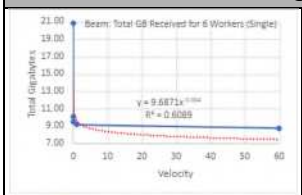


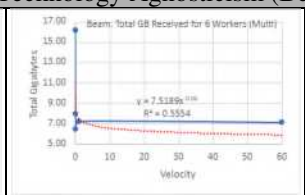


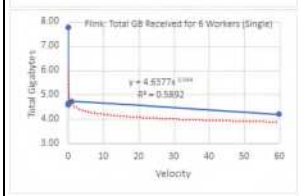


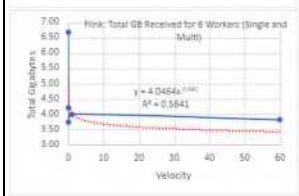


Technology Agnosticism (Beam and Flink) for 6 Workers

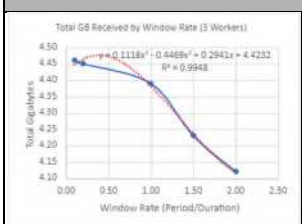


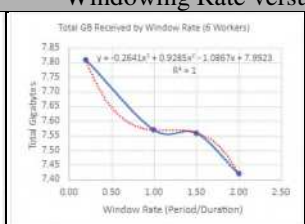


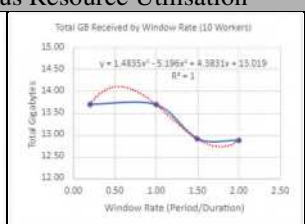


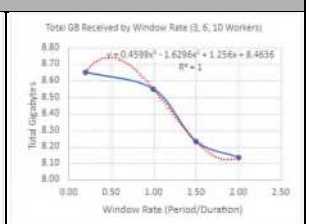


Windowing Rate versus Resource Utilisation



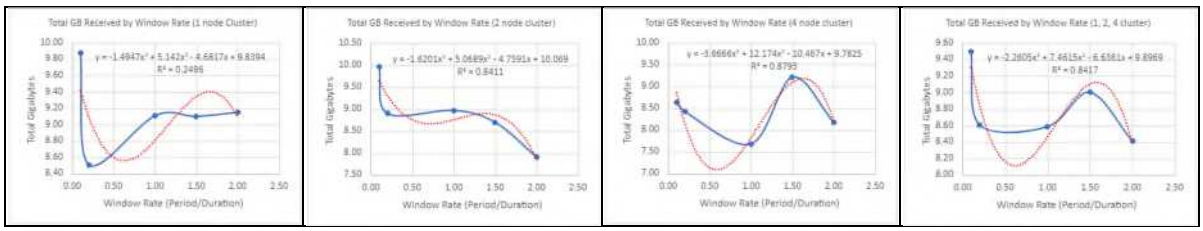






Container Co-Location

Maximum Node Network Utilisation by Node Cluster



4.2.7. Container Network Utilisation (GB Sent and Received by Jobmanager)

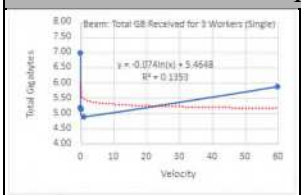
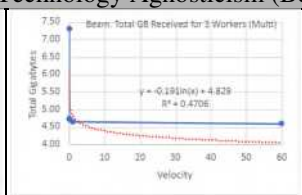
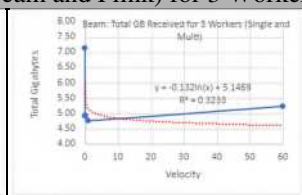
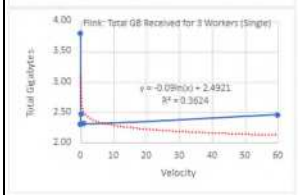
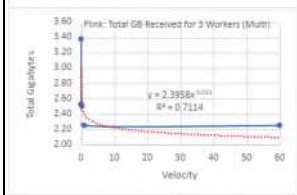
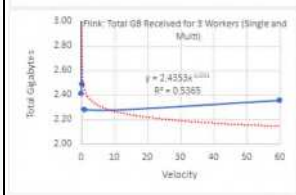
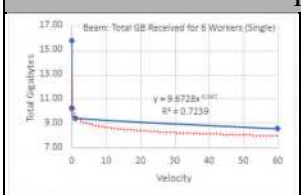
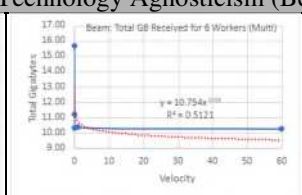
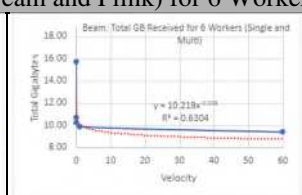
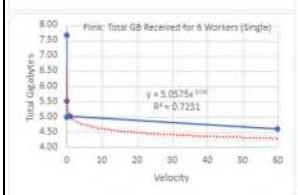
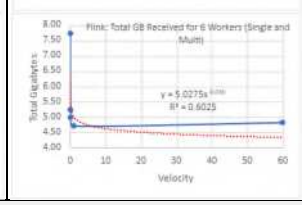
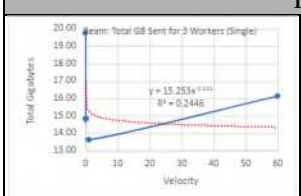
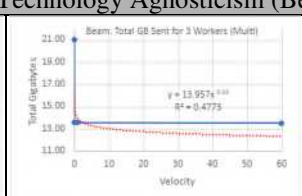
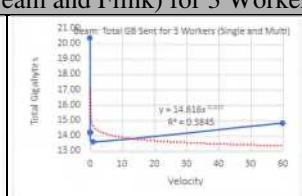
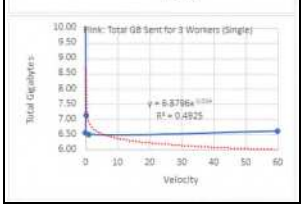
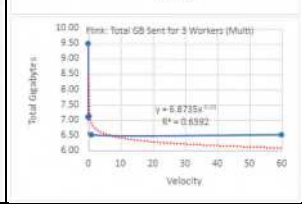
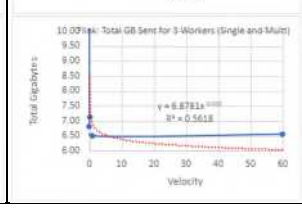
Based on the summary results presented in Table 13 for total number of GB received and sent over the network by the Jobmanager during the Technology Agnosticism experiments, the following conclusions were drawn:(i) total number of GB received by Jobmanager over the network by velocity for technology agnosticism – all means for three or six workers (Beam, Flink) x (single, multi) are not significantly different from each other except for 6 workers (multi) x 6 workers (multi). However, means are significantly different for all possible combinations [Beam, Flink][(3 workers (Single, Multi) x 6 workers (Single, Multi)], except for [Beam][(3 workers (multi) x 6 workers (multi)]; (ii) total number of GB sent by Jobmanager over the network by velocity for technology agnosticism – all means for three or six workers (Beam, Flink) x (single, multi) are not significantly different from each other. However, means are significantly different for all possible combinations [Beam, Flink][(3 workers (Single, Multi) x 6 workers (Single, Multi)]. This corroborates the observations described in Section 4.2.6, namely, that the greater the number of workers in the cluster, the greater the parallelisation of work, which leads to increased communication between containers across the network.

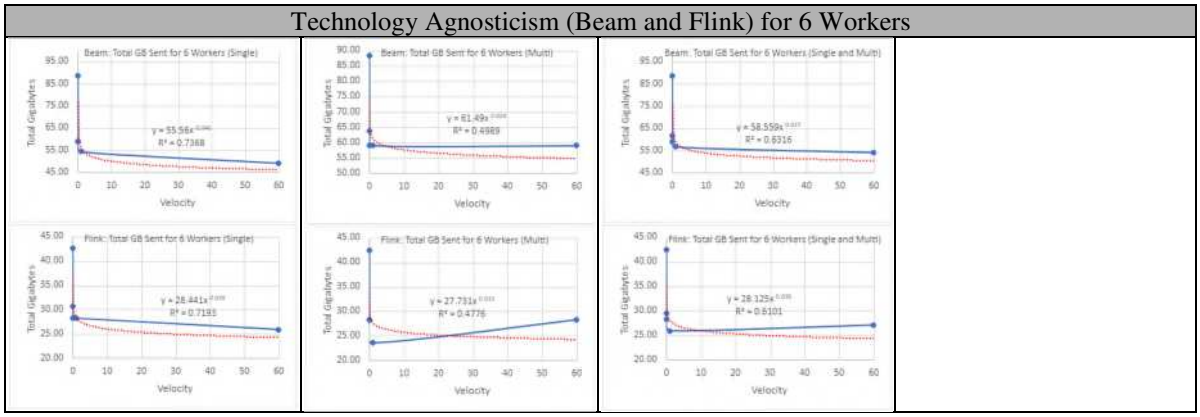
Table 13
Network Utilisation for Technology Agnosticism– Total Number of Gigabytes Received and Sent by Jobmanager

Total Number of GB Received by Jobmanager over the Network	Technology Agnosticism Metrics (for Beam and Flink SDK)										
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results	
		Beam		Flink		Beam		Flink		Beam	Flink
		S	M	S	M	S	M	S	M	3 Workers and 3 Workers (Single and Multi)	
	1 min	5.87	4.59	2.46	2.25	8.55	10.25	4.60	5.04	p-value = 0.1425, accept Ho	p-value = 1.0000, accept Ho
	1 sec	4.88	4.65	2.31	2.25	9.42	10.35	5.02	4.38	3 Workers and 6 Workers ([S,S] x [M,M])	
	100 ms	5.14	4.74	2.47	2.51	10.27	11.18	4.99	4.98	p-value = 0.01193, reject Ho (S,S)	p-value = 0.007937, reject Ho (S,S)
	10 ms	5.17	4.71	2.30	2.53	10.27	10.33	5.52	4.98	p-value = 0.007937, reject Ho (S,M) p-value = 0.001193, reject Ho (M,S) p-value = 0.4020, accept Ho (M,M)	p-value = 0.01193, reject Ho (S,M) p-value = 0.01193, reject Ho (M,S) p-value = 0.01167, reject Ho (M,M)
	1 ms	6.98	7.31	3.80	3.38	15.78	15.66	7.68	7.76	6 Workers and 6 Workers (Single and Multi)	
	Average	5.61	5.20	2.67	2.58	10.86	11.55	5.56	5.43	p-value = 0.01167, reject Ho	p-value = 0.6572, accept Ho
Total Number of GB Sent by Jobmanager over the Network	Technology Agnosticism Metrics (for Beam and Flink SDK)										
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results	
		Beam		Flink		Beam		Flink		Beam	Flink
		S	M	S	M	S	M	S	M	3 Workers and 3 Workers (Single and Multi)	
	1 min	16.13	13.53	6.60	6.52	49.13	58.96	25.92	28.27	p-value = 0.1425 , accept Ho	p-value = 1.0000, accept Ho
	1 sec	13.62	13.56	6.50	6.52	54.26	58.98	28.24	23.59	3 Workers and 6 Workers ([S,S] x [M,M])	
	100 ms	14.83	13.59	7.12	7.13	59.12	63.85	28.29	28.23	p-value = 0.007937, reject Ho (S,S)	p-value = 0.007937, reject Ho (S,S)
	10 ms	14.84	13.59	6.54	7.09	59.11	58.94	30.70	28.24	p-value = 0.007937, reject Ho (S,M) p-value = 0.01193 , reject Ho (M,S) p-value = 0.01193, reject Ho (M,M)	p-value = 0.007937, reject Ho (S,M) p-value = 0.01193 , reject Ho (M,S) p-value = 0.01193, reject Ho (M,M)
	1 ms	19.77	21.02	10.72	9.52	88.56	88.42	42.57	42.43	6 Workers and 6 Workers (Single and Multi)	
	Average	15.84	15.06	7.50	7.36	62.04	65.83	31.14	30.15	p-value = 0.8413, accept Ho	p-value = 0.4633, accept Ho

Results (in the form graphs and regression models) for the number of gigabytes received and sent by the Jobmanager during the Technology Agnosticism experiments by velocity are tabulated in Table 14.

Table 14
Network Utilisation for Technology Agnosticism (Gigabytes Received and Sent by Jobmanager) – Graphs and Regression Models

Total Gigabytes Received via the Network by Jobmanager			
Technology Agnosticism (Beam and Flink) for 3 Workers			
			
			
Technology Agnosticism (Beam and Flink) for 6 Workers			
			
			
Total Gigabytes Sent via the Network by Jobmanager			
Technology Agnosticism (Beam and Flink) for 3 Workers			
			
			



Based on the summary results presented in Table 15 for the total number of GB received and sent by the Jobmanager over the network during the Windowing Rate versus Resource Utilisation experiments, the following conclusions were drawn: (i) total number of GB received by Jobmanager over the network by velocity for windowing rate – all means for (3, 6, 10) workers are significantly different from each, other except for 6 workers x 10 workers; (ii) total number of GB sent by Jobmanager over the network by velocity for windowing rate – all means for (3, 6, 10) workers are significantly different from each other. This is consistent with the results observed in Sections 4.2.5 and 4.2.6, and explained by the fact that parallelisation of work is higher in larger clusters, leading to increased communication between containers across the network. While this model is a good fit for distributed systems hosted entirely on-premises, it could result in additional costs when commercial clouds are used.

Table 15
Network Utilisation for Windowing Rate versus Resource Utilisation – Total Number of Gigabytes Received and Sent by Jobmanager over the Network

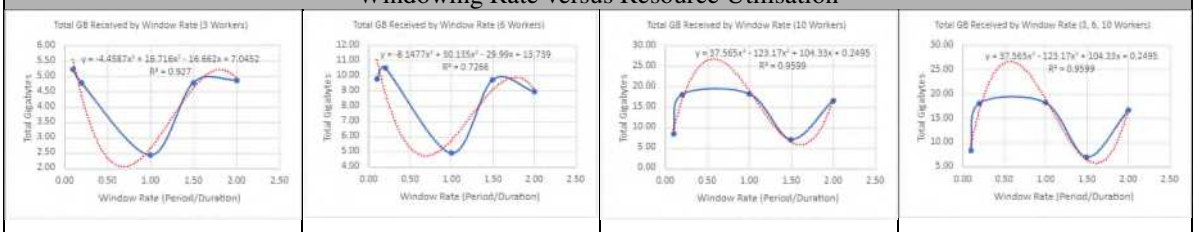
Total Number of GB Received by Jobmanager over the Network	Windowing Rate versus Resource Utilisation Metrics				
	Windowing Rate (Period/Duration)	Three Workers (n=3)	Six Workers (n=6)	Ten Workers (n=10)	Mann Whitney U Test Results
	0.1	5.21	9.76	8.29	3 Workers and 6 Workers
	0.2	4.77	10.51	18.02	p-value = 0.01587, reject Ho
	1.0	2.42	4.89	18.17	3 Workers and 10 Workers
	1.5	4.79	9.73	7.01	p-value = 0.007937, reject Ho
	2.0	4.87	8.94	16.56	6 Workers and 10 Workers
	Average	4.41	8.77	13.61	p-value = 0.4206, accept Ho
Total Number of GB Sent by Jobmanager over the Network	Windowing Rate versus Resource Utilisation Metrics				
	Windowing Rate (Period/Duration)	Three Workers (n=3)	Six Workers (n=6)	Ten Workers (n=10))	Mann Whitney U Test Results
	0.1	14.90	54.93	76.20	3 Workers and 6 Workers
	0.2	13.75	59.49	165.26	p-value = 0.007937, reject Ho
	1.0	6.91	27.45	165.13	3 Workers and 10 Workers
	1.5	13.76	54.94	63.68	p-value = 0.007937, reject Ho
	2.0	13.78	50.35	139.77	6 Workers and 10 Workers
	Average	12.62	49.43	122.01	p-value = 0.07937, reject Ho

Results (in the form graphs and regression models) for the number of gigabytes received and sent by the Jobmanager during the Windowing Rate versus Resource Utilisation experiments by windowing rate are tabulated in Table 16.

Table 16
Network Utilisation for Windowing Rate Versus Resource Utilisation (Gigabytes Received and Sent by Jobmanager) – Graphs and Regression Models

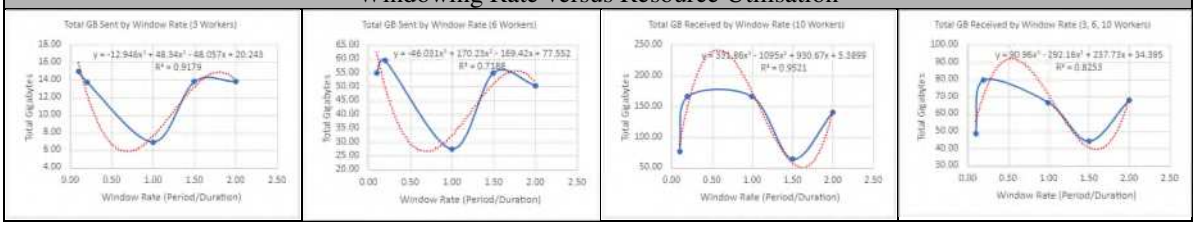
Total Gigabytes Received via the Network by Jobmanager

Windowing Rate versus Resource Utilisation



Total Gigabytes Sent via the Network by Jobmanager

Windowing Rate versus Resource Utilisation



Based on the summary results presented in Table 17 for the total number of GB received and sent by the Jobmanager over the network during the Container Co-Location experiments, the following conclusions were drawn: (i) total number of GB received by Jobmanager over the network by number of containers per node – all means for (1, 2, 4) clusters are not significantly different from each other; (ii) total number of GB received by Jobmanager over the network by number of containers per node – all means for (1, 2, 4) workers are not significantly different from each other, except for (3 workers x 6 workers). This is an important finding when considered alongside the network utilisation readings for workers. It establishes that, generally, when a cluster increases in size, the effect on manager-to-worker communication is minimal. Worker-to-worker communication, on the other hand, increases significantly since there is greater parallelisation.

Table 17
Network Utilisation for Container Co-Location – Total Number of Gigabytes Received and Sent by Jobmanager over the Network

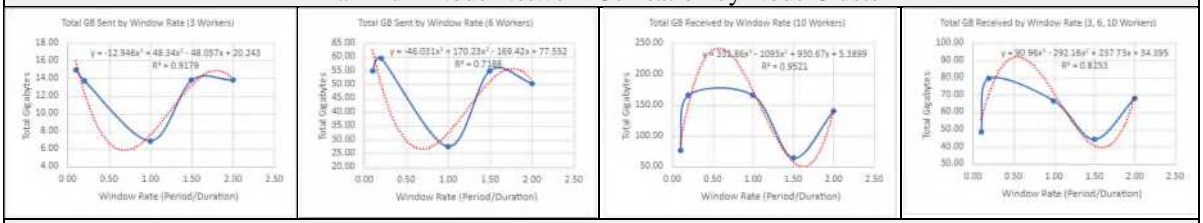
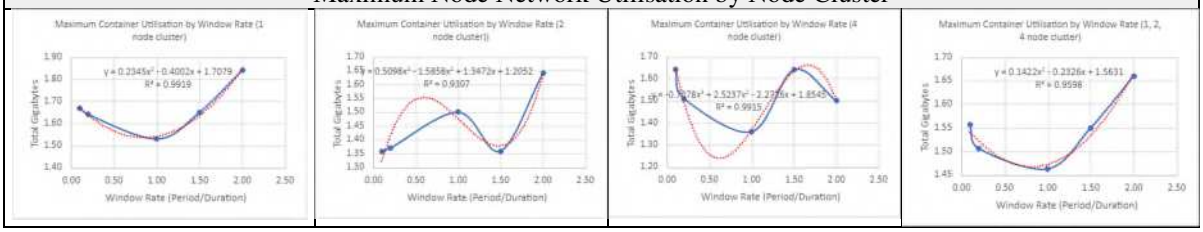
Total Number of GB Received by Jobmanager over the Network by Number of Containers per Node	Container Co-Location Metrics				
	Windowing Rate (Period/Duration)	Cluster (n=1)	Cluster (n=2)	Cluster (n=4)	Mann Whitney U Test Results
	0.1	1.67	1.36	1.64	3 Workers and 6 Workers
	0.2	1.64	1.37	1.51	p-value = 0.02733, accept Ho
	1.0	1.53	1.50	1.36	3 Workers and 10 Workers
	1.5	1.65	1.36	1.64	p-value = 0.05701, accept Ho
	2.0	1.84	1.64	1.50	6 Workers and 10 Workers
	Average	12.62	49.43	122.01	p-value = 0.3337, accept Ho
Total Number of GB	Container Co-Location Metrics				

Sent by Jobmanager over the Network by Number of Containers per Node	Windowing Rate (Period/Duration)	Cluster (n=1)	Cluster (n=2)	Cluster (n=4)	Mann Whitney U Test Results
	0.1	12.20	10.17	12.21	3 Workers and 6 Workers
	0.2	12.22	10.16	11.18	p-value = 0.01167, reject Ho
	1.0	11.20	11.18	10.17	3 Workers and 10 Workers
	1.5	12.20	10.17	12.19	p-value = 0.09269, accept Ho
	2.0	12.28	10.18	10.17	6 Workers and 10 Workers
	Average	1.67	1.45	1.53	p-value = 0.2343, accept Ho

Note: NA means data is not available

Results (in the form graphs and regression models) for the number of gigabytes received and sent by the Jobmanager during the Container Co-Location experiments by windowing rate are tabulated in Table 18.

Table 18
Network Utilisation for Container Co-Location (Gigabytes Received and Sent by Jobmanager) – Graphs and Regression Models

Total Number of GB Received by Jobmanager over the Network by Number of Containers per Node
Container Co-Location
Maximum Node Network Utilisation by Node Cluster

Total Number of GB Sent by Jobmanager over the Network by Number of Containers per Node
Container Co-Location
Maximum Node Network Utilisation by Node Cluster


4.2.8. Total Records Processed and Data Loss

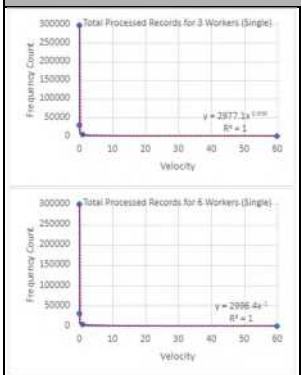
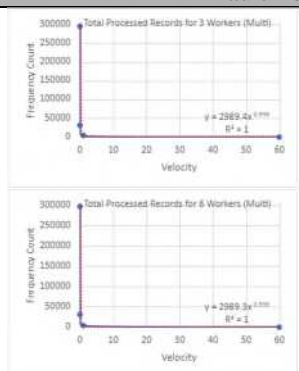
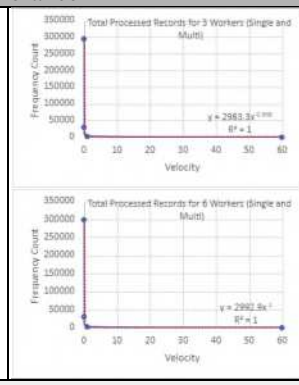
Based on the summary results presented in Table 19 for the total number of records processed during the fault tolerance experiments, the following conclusions were drawn: (i) total records processed by fault tolerance – all means for (3, 6) workers x (single, multi) are not significantly different from each other; (ii) data loss by fault tolerance – all means for (3, 6) workers x (single, multi) are not significantly different from each other.

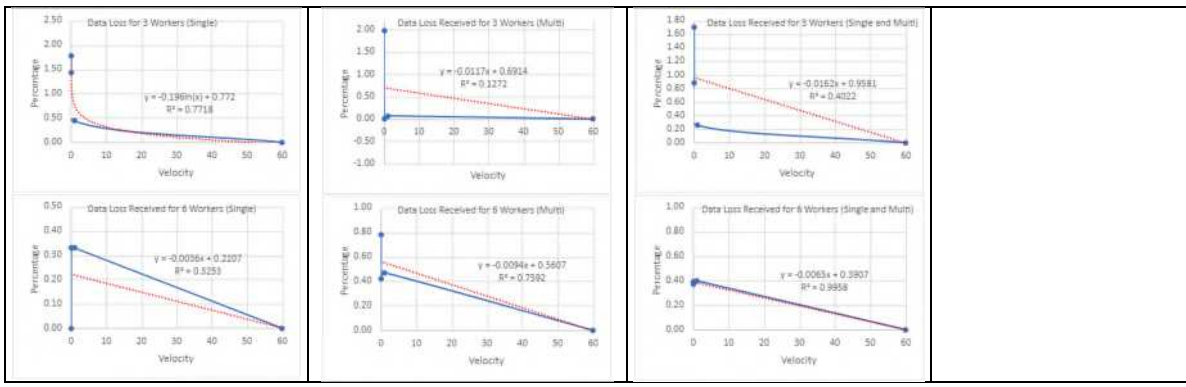
Table 19
Experimental Results for Records Processed and Data Loss During the Evaluation of MC-BDP Reference Architecture for Fault Tolerance

Total Records Processed	Fault Tolerance Metrics					
	Velocity	Three Workers (n=3)		Six Workers (n=6)		Mann Whitney U Test Results
		Single	Multi	Single	Multi	3 Workers and 3 Workers (Single and Multi)
	1 min	50	50	50	50	p-value = 1.0000, accept Ho
	1 sec	2987	2998	2990	2986	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	29478	30030	29902	29876	p-value = 0.7715, accept Ho (S,S)
	10 ms	295765	294179	300673	297689	p-value = 1.0000, accept Ho (S,M) p-value = 1.0000, accept Ho (M,S) p-value = 1.0000, accept Ho (M,M)
	1 ms	NA	NA	NA	NA	6 Workers and 6 Workers (Single and Multi)
	Average	82070	81814	83403	82650	p-value = 0.7715, accept Ho
Data Loss %	Fault Tolerance Metrics					
	Velocity	Three Workers (n=3)		Six Workers (n=6)		Mann Whitney U Test Results
		Single %	Multi %	Single %	Multi %	3 Workers and 3 Workers (Single and Multi)
	1 min	0.00	0.00	0.00	0.00	p-value = 0.6573, accept Ho
	1 sec	0.44	0.07	0.33	0.47	3 Workers and 6 Workers ([S,S] x [M,M])
	100 ms	1.77	0.00	0.33	0.42	p-value = 0.1804, accept Ho (S,S)
	10 ms	1.43	1.98	0.00	0.78	p-value = 0.5614, accept Ho (S,M) p-value = 0.877, accept Ho (M,S) p-value = 0.6573, accept Ho (M,M)
	1 ms	NA	NA	NA	NA	6 Workers and 6 Workers (Single and Multi)
	Average	0.91	0.51	0.17	0.42	p-value = 0.1804, accept Ho

Results (in the form graphs and regression models) for the total number of records processed and percentage of data loss during the Fault Tolerance experiments by velocity are tabulated in Table 20.

Table 20
Total Records Processed and Data Loss in a Fault Tolerance Scenario – Graphs and Regression Models

Total Processed Records
Fault Tolerance
  
Data Loss %
Fault Tolerance



4.2.9. Data Ingested

Based on the summary results presented in Table 21 for the total data ingested during the Technology Agnosticism experiments, the following conclusions were drawn: data ingestion by technology agnosticism – means are not significantly different for all combinations for Beam and Flink [(3 workers, 6 workers) x (single, multi), (3 workers (single, multi) x 6 workers (single, multi)].

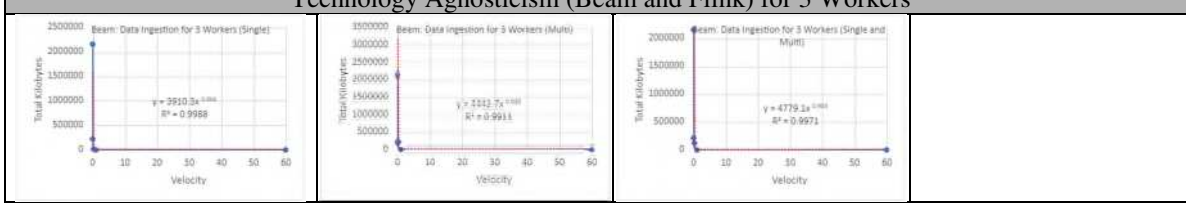
Table 21
Experimental Results for Data Ingestion during the Evaluation of MC-BDP Reference Architecture for Technology Agnosticism

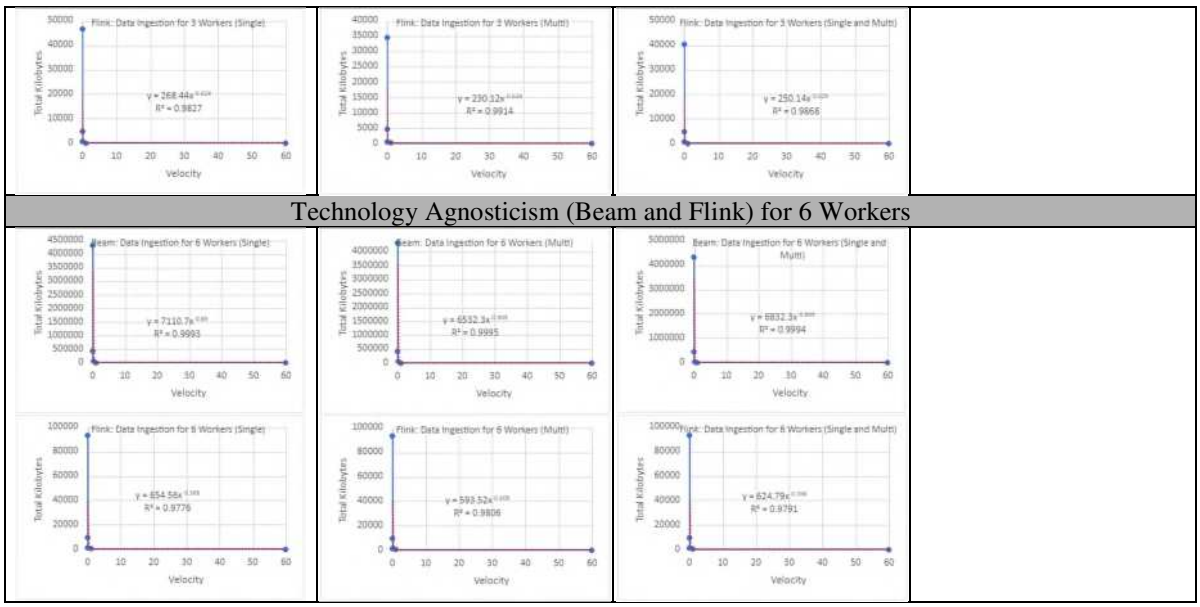
Data Ingestion (KB) by Velocity	Technology Agnosticism Metrics (for Beam and Flink SDK)									
	Velocity	Three Workers (n=3)				Six Workers (n=6)				Mann Whitney U Test Results
		Beam		Flink		Beam		Flink		3 Workers and 3 Workers (Single and Multi)
		S	M	S	M	S	M	S	M	p-value = 1.0000 , accept Ho / p-value = 0.9166, accept Ho
	1 min	172	74	55	39	259	205	158	126	3 Workers and 6 Workers ([S,S] x [M,M])
	1 sec	2373	2328	98	90	4824	4824	242	230	p-value = 0.6905, accept Ho (S,S) / p-value = 0.5476, accept Ho (S,S)
	100 ms	21900	219300	510	543	44580	44580	1086	1074	p-value = 0.6905, accept Ho (S,M) / p-value = 0.5476, accept Ho (S,M)
	10 ms	215700	215700	4680	4680	432600	432600	9480	9420	p-value = 0.8413, accept Ho (M,S) / p-value = 0.5476, accept Ho (M,S)
	1 ms	2154000	2154000	46500	34500	4308000	4308000	93000	93000	p-value=0.8413, accept Ho (M,M) / p-value = 0.5476, accept Ho (M,M)
	Average	478829	518280	10368	7970	958052	958041	20793	20770	6 Workers and 6 Workers (Single and Multi)
										p-value = 1.000, accept Ho / p-value = 0.583, accept Ho

Results (in the form graphs and regression models) for data ingested during the Technology Agnosticism experiments by velocity are tabulated in Table 22.

Table 22
Data Ingestion during the Evaluation of MC-BDP Reference Architecture for Technology Agnosticism – Graphs and Regression Models

Data Ingestion (Kilobytes) by Velocity
Technology Agnosticism (Beam and Flink) for 3 Workers





Based on the summary results presented in Table 23 for the data ingested during the Windowing Rate versus Resource Utilisation and Container Co-Location experiments, the following conclusions were drawn: (i) total KB ingested by windowing rate – means for all (3, 6, 10) workers are not significantly different from each other; (ii) total number of records ingested by windowing rate – means for all (3, 6, 10) workers are not significantly different from each other; (iii) total KB ingested by windowing rate and number of containers per node – means for all (1, 2, 4) clusters are not significantly different from each other; (iv) total number or records ingested by windowing rate and number of containers per node – means for all (1, 2, 4) clusters are not significantly different from each other.

Table 23
Experimental Results for Data Ingestion during the Evaluation of MC-BDP Reference Architecture for Windowing Rate versus Resource Utilisation and Container Co-Location

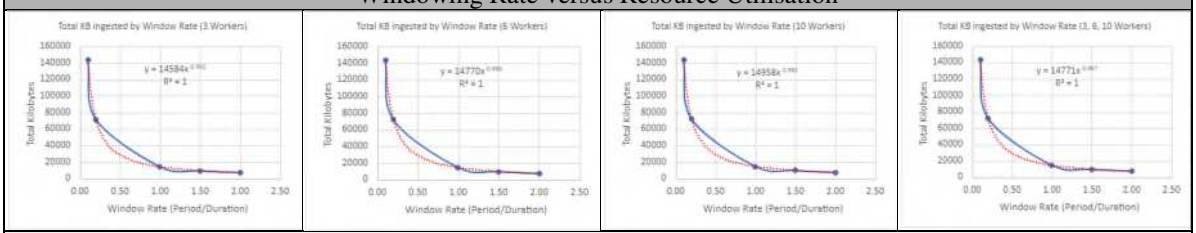
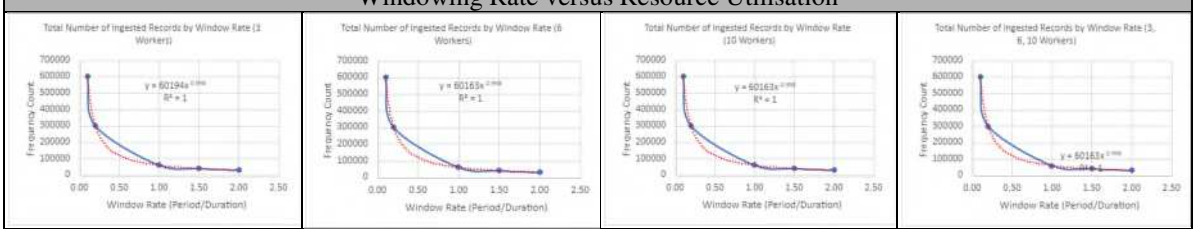
Total KB Ingested by Windowing Rate (Period/Duration)	Windowing Rate versus Resource Utilisation Metrics				
	Windowing Rate (Period/Duration)	Three Workers (n=3)	Six Workers (n=6)	Ten Workers (n=10)	Mann Whitney U Test Results
	0.1	143700	143800	144000	3 Workers and 6 Workers
	0.2	71900	71990	72280	p-value = 0.6905, accept Ho
	1.0	14500	14650	14814	3 Workers and 10 Workers
	1.5	9690	9892	10052	p-value = 0.6905, accept Ho
	2.0	7410	7515	7638	6 Workers and 10 Workers
	Average	49440	49569	49756	p-value = 0.6905, accept Ho
Total Number of Records Ingested by Windowing Rate (Period/Duration)	Windowing Rate versus Resource Utilisation Metrics				
	Windowing Rate (Period/Duration)	Three Workers (n=3)	Six Workers (n=6)	Ten Workers (n=10))	Mann Whitney U Test Results
	0.1	600000	600000	600000	3 Workers and 6 Workers
	0.2	300000	300000	300000	p-value = 0.9161, accept Ho
	1.0	60000	60000	60000	3 Workers and 10 Workers
	1.5	39945	40106	39920	p-value = 0.9161, accept Ho
	2.0	30361	30197	29637	6 Workers and 10 Workers

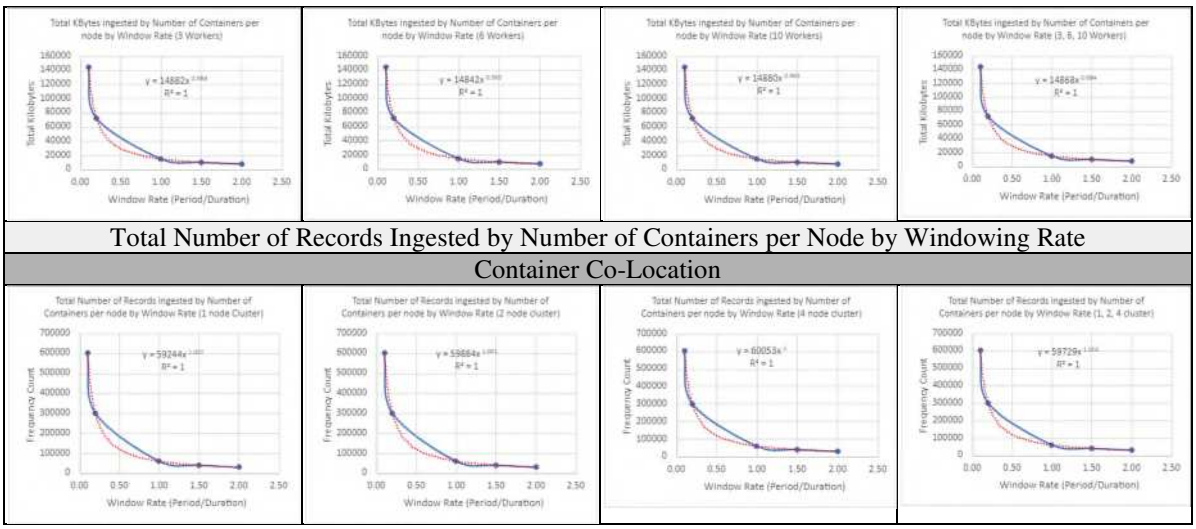
	Average	206061	206060	205911	p-value = 0.9161, accept Ho
Total KB ingested by Number of Containers per Node Windowing Rate (Period/Duration)	Container Co-Location Metrics				
	Windowing Rate (Period/Duration)	Cluster (n=1)	Cluster (n=2)	Cluster (n=4)	Mann Whitney U Test Results
	0.1	144100	144000	143900	3 Workers and 6 Workers
	0.2	72150	72090	72140	p-value = 0.8413, accept Ho
	1.0	14746	14724	14711	3 Workers and 10 Workers
	1.5	10044	9935	9998	p-value = 0.8413 , accept Ho
	2.0	7548	7569	7593	6 Workers and 10 Workers
	Average	49717	49663	49668	p-value = 1.0000 , accept Ho
Total Number of Records Ingested by Number of Containers per Node by Windowing Rate (Period/Duration)	Container Co-Location Metrics				
	Windowing Rate (Period/Duration)	Cluster (n=1)	Cluster (n=2)	Cluster (n=4)	Mann Whitney U Test Results
	0.1	600000	600000	600000	3 Workers and 6 Workers
	0.2	300000	300000	300000	p-value = 0.9161 , accept Ho
	1.0	60000	60000	60000	3 Workers and 10 Workers
	1.5	39827	39869	40073	p-value = 0.9161 , accept Ho
	2.0	28881	29898	30037	6 Workers and 10 Workers
	Average	205741	205953	206022	p-value = 0.9161, accept Ho

Note: NA means data is not available

Results (in the form graphs and regression models) for data ingested during the Windowing Rate versus Resource Utilisation and Container Co-Location experiments by velocity are tabulated in Table 24.

Table 24
Data Ingestion during the Evaluation of MC-BDP Reference Architecture for Windowing Rate versus Resource Utilisation and Container Co-Location – Graphs and Regression Models

Total Kilobytes Ingested by Windowing Rate
Windowing Rate versus Resource Utilisation

Total Number of Records Ingested by Windowing Rate
Windowing Rate versus Resource Utilisation

Total Kilobytes Ingested by Number of Containers per Node by Windowing Rate
Container Co-Location



This section presented the experimental results obtained in the technical evaluation of the MC-BDP reference architecture. The next section concludes this paper by proposing recommendations for future work and discussing the limitations of our research.

5. Conclusion and Future Work

This research set out to investigate a unified vendor-agnostic solution to big data stream processing in a multi-cloud environment. As key beneficiaries of commercial cloud computing, small and medium enterprises, organisations or departments characterised by devolved administration, tight budgetary constraints and lack of specialised technical skills in-house (SMEODs) were selected as the target domain for MC-BDP, a new reference architecture for big data stream processing aimed at maximising the cloud's economies of scale whilst at the same reducing the risk of vendor lock-in.

The majority of developments targeted at facilitating the adoption of big data analytics through cloud computing have focused on reducing the complexity of implementing and managing large distributed systems. Consequently, solutions are predominantly SaaS-based and associated with another major cause of apprehension: the risk of vendor lock-in. A number of authoritative studies have been conducted on possible ways of mitigating the risk of vendor lock-in [30], [31], [56]. However, no solutions were found which minimised the risk of vendor lock-in whilst simultaneously allowing implementers to maximise the cloud's economies of scale. Although the combination of containers, cloud computing and stream big data is not new, the field lacked a domain-specific approach developed from a cloud consumer's perspective to enable implementers to ingress into stream big data analytics using a cloud model less restrictive than SaaS. As a response to the breadth, extensiveness and complexity of big data research, recent developments have tended to focus on specific domains such as the biomedical sciences [10], the IoT [55], edge computing [57], national security [11] and scientific simulations [9]. The current research is one such effort, developed to fill the gap for a systematic approach to stream big data processing targeted at a domain whose main preoccupation is with minimising the risk of vendor lock-in whilst at the same time maximising the cloud's economies of scale.

Motivated by a desire to facilitate the adoption of big data stream analytics, this research proposed a new systematic and unified approach to stream big data. The MC-BDP reference architecture was created to:

1) leverage the cloud's economies of scale by promoting an infrastructure hosted on commercial clouds;
2) move away from the traditional SaaS approach towards a standardised form of PaaS;
3) mitigate the risk of vendor lock-in by prescribing the use of portable, interoperable and vendor-agnostic components deployed to multiple clouds; and
4) alleviate concerns around complexity and skills shortage in the domain by providing a domain-specific reference architecture to guide implementers.

This research challenges the accepted belief according to which the most appropriate cloud consumption model for SMEs is SaaS [12], [13], [72]–[74]. Whilst acknowledging the concerns around technical complexity and skills shortage which have led previous authors to recommend the SaaS cloud model to SMEs, particularly within the realm of big data, this research puts equal importance on the perceived risk of vendor lock-in explored extensively in recent research [30], [31], [75]–[77]. The MC-BDP reference architecture, designed specifically for the SMEOD domain, provides a good level of scaffolding to enable implementers to navigate the complexities of big data technology and cloud computing without having to employ a dedicated and highly specialised technical team.

The simplification and systematisation provided by the MC-BDP reference architecture allowed this research to break with tradition [12], [13], [72]–[74] and recommend a cloud consumption model more desirable than SaaS from a vendor lock-in perspective. This is a significant contribution to the field of stream big data which, up until now, lacked a systematic and thoroughly researched approach to big data based on a consumption model other than SaaS. It is not expected that research on specialised SaaS provisions for SMEs will cease given that concerns around the risk of vendor lock-in are not universally or evenly shared amongst companies – some could find that this is a risk worth taking for the benefit of simplicity. Therefore, work such as Heilig & Voß's (2017) reference architecture for big data designed from a cost perspective or Sang et al.'s (2016) reference architecture based on real-world big data implementations, discussed in Chapter 2, Sections 2.2.1.5. and 2.2.1.7. respectively, remain valid and indeed useful within the field. What this research has demonstrated however is that rather than searching for a universal architecture for big data to suit all SMEODs, research should be directed towards finding approaches which are flexible and abstract enough to account for the immense variety in requirements and priorities that characterise the domain.

This section looks to revisit the aim and objectives of this research, stated in Section 1.2, and reflect on the work carried out to realise them. As an investigation into unified vendor-agnostic stream processing for big data in a multi-cloud environment, this research started with a comprehensive literature review of the fields of big data, cloud computing and virtualisation. Having found a gap in the literature which called for a systematic, domain-specific approach to stream big data purposely targeted at SMEODs, this research proceeded to investigate requirements specific to this domain such as low initial investment, simplicity and ease of installation and maintenance, as well as a low risk of vendor lock-in. It was found that a reference architecture, combined with a multi-cloud containerised infrastructure commissioned on a PaaS model provided the necessary guidance to implementers whilst considerably reducing the risk of vendor lock-in by relying on portable and interoperable open-source components. As part of requirements gathering, ten non-functional requirements for big data were inferred from a selection of academic papers and other official literature published by three real-world companies: Facebook, Twitter and Netflix. They were published as a conference paper and presented at the 13th International Conference on Software Technologies [29]. An

extended version of the paper was subsequently selected for publication as a book chapter on requirements engineering for large-scale big data applications [1]. RO1 was thus fulfilled following peer review of the aforementioned publications.

Informed by the above study, MC-BDP, a new reference architecture based on industry's best practices was proposed, focused on maximising the cloud's economies of scale whilst minimising the risk of vendor lock-in. The new reference architecture provided an answer to RQ2, and its publication as a conference paper [78] demonstrated the fulfilment of RO2. A methodology combining post-positivist and interpretivist methods was selected for the evaluation of the proposed reference architecture following a gap in the literature and recent calls for more mixed-methods research in the field of information systems. It was decided that a prototype implementation of MC-BDP would be constructed and evaluated within the context of a case study. The Estates and Sustainability departments at Leeds Beckett University were selected for the case study for having the particular conditions to allow the testing of the research's hypotheses. The data they possessed was characterised by the three Vs of big data: volume, velocity and variety. Moreover, a previous study provided evidence that the PUE monitoring for one of the university's data centres had been severely limited by the technology available [61] and would most likely benefit from a stream approach to data processing. While the former study relied on half-hourly emissions of the total energy consumed by the data centre and half-hourly emissions of the energy consumed by the IT equipment installed in the data centre, the current study used a simulator to emit data at much higher velocities. Additionally, instead of waiting to process historical data in batches, this prototype calculated the PUE in real time, or close to real-time.

The literature review established that the most appropriate big data infrastructure for SMEODs is based on commercial clouds, given that no substantial initial investment is required thanks to their pay-as-you-go model. Furthermore, in order to mitigate the risk of vendor lock-in, the implementation had to demonstrate that an infrastructure distributed across different clouds was viable and did not incur significant performance overheads. It was therefore decided that the prototype was to be developed using commercial clouds, so applications for grants were submitted to major providers. Having secured generous sponsorship credits from Microsoft Azure, Google Cloud and the Open Science Data Cloud, an initial prototype was developed for evaluation using predominantly open-source technology, given that they are cost-free and therefore more attractive to the target domain. The following technologies were used for the MC-BDP prototype implementation: Ubuntu 16.4 operating system, Docker Community Edition (CE), Docker Swarm container orchestrator, Docker Hub image library, Weave Net Plugin for container networking, Apache Flink 1.3.2 framework, Apache Beam 2.1.0 framework and Apache Kafka 0.10.2.0 distributed streaming platform. The successful development of the prototype using open-source technologies provided an answer to RQ3. Additionally, a simulator for the EGX300 gateway server located at one of Leeds Beckett University's data centres was developed using a combination of plain Java for its main components and Grails 3.3.1 for its user interface. The rationale for selecting each technology has been carefully explained in Section 3.5. The completion of the prototype development signalled the fulfilment of RO3.

In order to address RO4, a set of metrics was selected to monitor the performance of the prototype at container and virtual machine levels. The following container metrics were obtained using the Container Monitoring Service from Microsoft Azure: CPU Utilisation Average, CPU Utilisation Max, Memory Utilisation Average, Memory Utilisation Max, Total Bytes Sent and Total Bytes Received (over the network), thus answering RQ4. These metrics are detailed in Table 4. Various configurations using different speeds of incoming data, single and multi-cloud container clusters, different cluster sizes and different co-locations were compared based on these metrics, as summarised in Table 3. The 110 experiments performed under

controlled conditions demonstrated that MC-BDP's prototype was satisfactorily scalable and fault-tolerant across clouds, with no considerable difference observed between single and multi-cloud equivalent setups. Container co-location was also found not to significantly affect performance. In terms of technology agnosticism, the overhead introduced by the Beam framework in terms of CPU, memory and data transfer was considerable. The conclusion reached was that there is indeed a performance cost to technology agnosticism as implemented in MC-BDP's prototype by using a super-framework to provide portability of the code across various stream and batch technologies. Nevertheless, this may be a price some implementers would be willing to pay, so studies such as this where the overhead introduced by a super-framework was rigorously identified and measured are of utmost value to implementers who are able to take more informed decisions based on case-specific requirements for portability and interoperability of the data processing code. The relationship between the windowing function used for stream data processing and resource utilisation was also more thoroughly understood following MC-BDP's experimental evaluation. A simple formula representing the effects of the windowing function selected on CPU and network utilisation was proposed, and an approach to issue cluster size recommendations based on the windowing function desired was recommended for future work. It is important to note that, although exiting work can be found in the literature which links the windowing function to resource utilisation [79], the approach taken is one which assumes the infrastructure to be static and adjusts the windowing function to keep resource utilisation within a desirable range. Taking advantage of one of the cloud's most advantageous characteristics, its elasticity, the current research recommends a model where the windowing rate remains constant while the infrastructure commissioned expands and shrinks to keep performance metrics within a desirable range.

The quantitative findings of this research demonstrated that the proposed reference architecture is adequately scalable across different clouds. Given the cloud's pay-as-you-go model whereby consumers are only charged for the resources they utilise, being able to scale up or down is particularly important for SMEODs since budgetary constraints demand that resources be allocated sensibly and waste minimised. This conclusion is corroborated by theory T3, derived from the qualitative part of this research and outside the scope of the current paper, according to which economic factors such as cost, resource optimisation and being able to generate revenue are strong driving factors when assessing the suitability of information and data management strategies. The empirical findings therefore demonstrated that MC-BDP is scalable across clouds and that this requirement is relevant and valuable within the target domain. Fault tolerance across clouds was also demonstrated empirically, with no significant overhead observed in multi-cloud setups when compared to single-cloud ones. The relevance of this requirement to the domain of SMEs has been identified in the literature as the need for high availability of their production systems [13]. Since the aim of this research was to investigate stream big data processing from a vendor-agnostic perspective, being able to use any stream (or even batch) framework to run the processing code was identified as a desirable quality and integrated into the reference architecture proposed. The results of the technology agnosticism experiments revealed that the use of a super-framework such as Beam to provide portability and interoperability of the data processing code did have a cost in terms of performance, measured as, on average, 30% more CPU, 3% more RAM and 105% more network usage. The significance of these results to researchers and implementers is in being able to take more informed architectural decisions, particularly in the cloud where the estimated overhead can be translated into projected costs.

Of similar relevance are the findings of a direct relationship between the windowing rate and resource utilisation in a distributed big data stream system. Since it was demonstrated that the CPU and network utilisation can be predicted using a simple formula based on the windowing rate and constants derived from empirical observation, implementers now have a more accurate way of projecting the cluster size and the cost

of data transfers. It is believed that the aforementioned relationship is of wider application within the field of big data stream processing using commercial clouds. The container co-location findings confirm that co-location does not significantly affect the performance of CPU-intensive stream big data systems. In the context of cloud computing, this corroboration enables implementers to experiment with a greater number of deployment configurations, from multiple virtual machines with a few containers running in them to a smaller number of more powerful machines hosting a greater number of containers. Factors such as the desired level of fault tolerance (container, node, region or provider) or projected data transfer costs may play an important part in the decision-making, particularly considering that container-to-container data transfer costs can be reduced or eliminated through co-location. The cost-centric approach to task distribution offered by Li et al. (2020), aimed at minimising data transfer costs across clouds, is an important development in this direction. Performance monitoring at container and node level, combined with a weighted list of desired SLAs and the use of a flexible learning algorithm to make predictions is this research's answer to the same question, elaborated as a proposal for future work.

5.1. Recommendations for Future Research

The success of this research project has left some interesting avenues open for exploration. This section aims to examine these by providing an account of related projects which are planned or in progress and suggesting future work. The case study conducted with the Estates and Sustainability departments at Leeds Beckett University concluded with a strong desire by the participants to extend the investigation by considering an important aspect of big data which had previously been left outside of scope: its variety. The apprehension with regards to the variety of smart buildings data was directly captured in the qualitative evaluation of MC-BDP, which reflects concerns about the lack of standardisation of ingested data. The complexity and multidimensionality of buildings data has in fact been identified in recent research as one of the biggest challenges to big data technology adoption within the sector [81]. A new research project aimed at investigating the appropriateness of the MC-BDP reference architecture for use-cases where the variety of the data is paramount was created as the case study concluded. The project was scheduled to start in the second trimester of 2020, but had to be postponed due to the Covid-19 pandemic which led to the closure of the university campus. It is expected that the project will restart as soon as the university activities normalise following the re-opening of its facilities.

Other aspects of MC-BDP which would benefit from future work are:

- Investigate how MC-BDP handles batch process. Compare Akidau et al.'s (2015) approach of using a stream engine to process both batch and stream with an alternative approach based on the lambda architecture.
- Following a trend observed in Jha et al.'s (2018) research on container co-location, observe how MC-BDP performs with jobs of different characteristics. Memory-intensive, network-intensive and disk I/O-intensive jobs could be created and compared to the CPU-intensive job utilised in the experiments.
- Integrating recent research on scheduling algorithms such as RTSATD, which uses task duplication to optimise the performance of big data stream processing across different cloud regions [59], or Zhao et al.'s (2020) algorithm developed to minimise data transfers over the network into MC-BDP's orchestration layer.
- Add a cost perspective to MC-BDP's evaluation by integrating fine-grained billing information obtained from cloud providers. This is in line with Heilig & Voß's (2017) and Li et al.'s (2020) research. It is believed that understanding non-functional requirements from a cost as well as

performance perspective would be advantageous to budget-constrained organisations.
- Extend MC-BDP's evaluation to include other domain-specific industry case studies.
- Conduct additional case studies with other types of SMEODs to validate the proposition that MC-BDP is beneficial to them.
- Strengthen the statistical significance of the quantitative results obtained by widening the scale of the experiments: use more than two commercial cloud providers, a greater number of virtual machines and containers, extending the experiments in duration and volume of incoming data.

As other relevant research is added to the fields of big data, cloud computing and virtualisation, new hypotheses and unexplored questions shall become apparent, impelling further investigation and solidifying this research's position as a mature contribution to the field.

5.2. Limitations of the Study

As is the case with all empirical investigations, there were limitations to the experiments conducted to evaluate the MC-BDP reference architecture, in light of which the results obtained must be understood. This section prevents these limitations and provides suggestions for further studies. Table 25 summarises the sixteen main limitations to the experimental evaluation, classified by type into one of the following four categories: financial, design, time/scope, and technical.

Table 25
Limitations to the Experimental Evaluation of MC-BDP Reference Architecture

Limitation	Type	Effect	Suggestion
1. Provider Variety	Financial	Two providers were used.	Repeat the multi-cloud experiments with a greater number of providers and verify if the results obtained are equivalent. Compare the performance of machines from different providers to verify if Google Cloud machines do indeed use more memory.
2. Cluster Size	Financial	Clusters of three, six and ten machines were used.	Repeat the experiments with larger clusters of up to hundreds of nodes.
3. Experiment Duration	Financial	Each experiment ran for five minutes.	Increase the duration of each run to several minutes, or even hours, and observe if there is a relationship between experiment duration and the performance metrics selected.
4. Container Co-Location	Design	Three container co-location configurations were used.	Repeat the experiments with a greater number of container co-locations, up to hundreds of containers per node, and observe the direct effect of co-location on the metrics selected.
5. Checkpointing Frequency	Design	Checkpointing was configured to take a snapshot every half second.	Experiment with different checkpointing configurations and observe the effects on general system performance versus the percentage of data loss.
6. Containers Monitored	Design	The jobmanager logs were not monitored for the scalability and fault tolerance experiments.	Repeat the scalability and fault tolerance experiment observing the number of gigabytes sent and received over the network.
7. Windowing Function Type	Design	Only windows which are time-based and uniformly distributed	Consider evaluating more complex time-based windowing functions such as sessions.

		were evaluated.	Consider evaluating count or punctuation-based windowing functions.
8. Data Transfer Configuration	Design	All data transfer between containers incurred networking charges, since there was no resource sharing.	Consider using a different container orchestrator such as Kubernetes, which allows resource sharing between containers deployed to the same node.
9. Simulator Scalability	Time/Scope	The simulator was deployed to a single high-spec physical machine and delays were observed in the emission of data at higher volumes/velocities.	Deploy the simulator to a dedicated distributed platform capable of scaling up to accommodate the more demanding experiments.
10. State Backend	Time/Scope	Checkpointing data was stored in-memory on the machine running the jobmanager service.	Configure a persistent state backend for checkpointing, so experiments using higher volumes/velocities of data can complete successfully.
11. Choice of Technology	Time/Scope	Only one prototype was used to evaluate the MC-BDP reference architecture.	Repeat the experiments using different big data frameworks such as Spark or Dataflow and observe how they compare. Repeat the experiments using a different orchestrator such as Kubernetes and see how the results compare. Repeat the experiments using a different container technology such as Linux Containers and see how the results compare. Repeat the experiments using a different platform/operating system and see how the results compare.
12. Pricing Model Complexity	Time/Scope	Complexities of different providers' pricing models were not considered in proposed formulas.	Consider the application of the formulas suggested in conjunction with the complexities involved in different providers' pricing models, such as fixed quotas free of charge, and regional discounts.
13. Measurement Frequency	Technical	The Azure Log Analytics agent for Linux computers was pre-configured with a ten second collection sample interval.	Experiment with monitoring solutions which allow greater flexibility in defining the data collection sample interval and observe how different data collection strategies affect the metrics observed.
14. Dynamic Scaling	Technical	Dynamic scaling was not offered by the Flink framework at the time the experiments were designed, so machines used to provide fault tolerance were added to the cluster from the start.	Experiment with dynamic scaling when available and observe how commissioning machines on an ad-hoc basis compares to commissioning them in advance. Compare the performance of different providers when machines are commissioned ad-hoc.
15. Virtual Machine Specification	Technical	The virtual machines commissioned for the four containers per node experiments had 16 GB of RAM each, when they should have had eight.	Consider using different providers which allow more flexibility when commissioning virtual machines.
16. Monitoring Metrics	Technical	Only container-level metrics were configured, so data had to be aggregated to perform node and cluster-level comparisons.	Consider using different monitoring technologies to observe the behaviour of virtual machines and of the whole cluster. Consider limiting the resources available to each container to a fixed quota to mimic what

			was available in the setting with no co-location.

The methodology used in this part of MC-BDP's evaluation ensured the experiments were focused, had a clear aim, and were conducted under controlled conditions. As a result, generalisation of their findings to wider contexts must consider the limitations shown in Table 5.1.

5.3. Conclusion

As a greater volume, velocity and variety of data is produced by an increasing number of internet-enabled systems and devices, existing data processing architectures are continually challenged. The perpetual quest for more efficient algorithms, more capable infrastructure and more powerful analytics manifested in academic research is followed with excitement and anticipation, but also apprehension and concern by businesses. Real-time intelligence from big data analytics has the power to transform companies and organisations by giving them an advantage over competitors. Furthermore, cloud computing's pay-as-you-go model has revolutionised access to technical infrastructure and enabled small and medium companies to partake of computing-intensive ventures previously inaccessible to them such as big data analytics. Nevertheless, harnessing the power of big data though cloud computing is not a straightforward enterprise, and solutions directly aimed at simplifying this process have traditionally carried considerable risks such as that of vendor lock-in [12], [13], [72]–[74].

Motivated to find a viable approach to big data stream processing in the cloud with a reduced risk of vendor lock-in but, at the same time, simple enough to be useful to SMEODs, this research followed a recent trend in the literature towards developing reference architectures for big data aimed at specific domains [9]–[11], [55], [57]. It worked from the premise that traditional SaaS-based approaches were not the ultimate solution for the domain, since they were associated with a high risk of vendor lock-in, and believed that the scientific community could benefit from contributions which enabled commercial cloud resources to be consumed by these companies on different models. MC-BDP was thus proposed as a reference architecture offering a systematic approach to big data stream processing using commercial clouds. Aimed at the more generic SMEOD domain, it simplified access to big data analytics by acting as an implementation blueprint, thus filling a gap in the literature for cloud computing information and training directly aimed at these companies [35], whilst at the same time mitigating the risk of vendor lock-in by promoting a containerised infrastructure demonstrated to be portable and interoperable across multiple commercial clouds.

MC-BDP was evaluated as part of a case study involving the Estates and Sustainability departments at Leeds Beckett university using a mixed-methods approach which combined post-positivist and interpretivist elements. In its post-positivist evaluation, presented in this paper, MC-BDP was demonstrated to be scalable and fault-tolerant across clouds. Moreover, different container co-locations were shown not to significantly affect performance. MC-BDP's provision for technology agnosticism through the use of a super-framework to allow code portability to other platforms was shown to incur some processing, memory and networking overhead. This overhead was measured and is believed to be of use to future researchers and implementers who may have code portability as a requirement. Finally, the relationship between windowing rate and resource utilisation was observed empirically, and a simple formula was derived to represent the relationship between the windowing rate and CPU or network utilisation.

Declarations

1. Ethics approval and consent to participate

This research project received ethics approval from the Local Research Ethics Committee of the School of Built Environment, Engineering and Computing at Leeds Beckett University. Informed consent was obtained from all individual participants included in the study.

2. Consent for publication

Participants of the energy efficiency case study signed informed consent regarding publishing their data.

3. Availability of data and materials

The data used to support the findings of this study is available from the corresponding author upon request.

4. Competing interests

The authors declare that they have no competing interests.

5. Funding

This work made use of the Open Science Data Cloud (OSDC) which is an Open Commons Consortium (OCC)-sponsored project. Cloud computing resources were provided by Google Cloud and Microsoft Azure for Research awards. Container and cloud native technologies were provided by Weaveworks.

6. Authors' contributions

TV built the prototype, performed the experiments, collected the data, and wrote the manuscript. AK produced the evaluation graphs, tables, and quantitative analysis. DM reviewed the draft manuscript and made editing suggestions. All authors read and approved the final manuscript.

7. Acknowledgements

We would like to thank the OSDC, Google Cloud, Microsoft Azure for Research and Weaveworks for their support of this research.

8. Authors' information

Thalita Vergilio is a Senior Lecturer at Leeds Beckett University and a researcher with the Cybercrime and Security Innovation Centre. With a strong industry background of over 10 years in software engineering, her research interests include stream big data frameworks, containers and container orchestration technology, web development best practices, systems architecture, and DevOps practices.

Ah-Lian Kor is a Reader in Data Science and Artificial Intelligence at Leeds Beckett University. She has been involved in several EU projects for Green Computing, Innovative Training Model for Social Enterprises Professional Qualifications, and Integrated System for Learning and Education Services. She has published work on Ontology, Semantics Web, Web Services, Portal and semantics for GIS.

Duncan Mullier is a Senior Lecturer in Computer Science at Leeds Beckett University. He gained a PhD in

1999 for work in Artificial Intelligence and neural networks for pattern recognition, and has contributed to several projects, most recently the DSCENT project for identifying terrorist behaviour using neural networks.

References

[1] T. Vergilio, M. Ramachandran, and D. Mullier, 'Requirements Engineering for Large Scale Big Data Applications', in *Software Engineering in the Era of Cloud Computing*, M. Ramachandran and Z. Mahmood, Eds. Springer International Publishing, 2019.

[2] J. Dean and S. Ghemawat, 'MapReduce: simplified data processing on large clusters', *Commun. ACM*, vol. 51, no. 1, p. 107, Jan. 2008, doi: 10.1145/1327452.1327492.

[3] K. Patel, Y. Sakaria, and C. Bhadane, 'Real Time Data Processing Frameworks', *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 5, pp. 49–63, Sep. 2015, doi: 10.5121/ijdkp.2015.5504.

[4] J. Li, D. Maier, K. Tufte, V. Papadimos, and P. A. Tucker, 'Semantics and Evaluation Techniques for Window Aggregates in Data Streams', in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2005, pp. 311–322. doi: 10.1145/1066157.1066193.

[5] T. Akidau *et al.*, 'MillWheel: fault-tolerant stream processing at internet scale', *Proc. VLDB Endow.*, vol. 6, no. 11, pp. 1033–1044, Aug. 2013, doi: 10.14778/2536222.2536229.

[6] J. Kreps, 'Questioning the Lambda Architecture - O'Reilly Media', Jul. 02, 2014. https://www.oreilly.com/ideas/questioning-the-lambda-architecture (accessed Oct. 28, 2016).

[7] G. J. Chen *et al.*, 'Realtime Data Processing at Facebook', in *Proceedings of the 2016 International Conference on Management of Data*, New York, NY, USA, 2016, pp. 1087–1098. doi: 10.1145/2882903.2904441.

[8] S. Krishnan, 'Discovery and Consumption of Analytics Data at Twitter', *Twitter Engineering Blog*, Jun. 29, 2016. https://blog.twitter.com/engineering/en_us/topics/insights/2016/discovery-and-consumption-of-analytics-data-at-twitter.html (accessed Feb. 09, 2018).

[9] A. Kashlev, S. Lu, and A. Mohan, 'Big Data Workflows: A Reference Architecture and The Dataview System', *Serv. Trans. Big Data*, vol. 4, no. 1, p. 19, 2017.

[10] V.-D. Ta, C.-M. Liu, and G. W. Nkabinde, 'Big data stream computing in healthcare real-time analytics', in *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, Chengdu, China, Jul. 2016, pp. 37–42. doi: 10.1109/ICCCBDA.2016.7529531.

[11] J. Klein, R. Buglak, D. Blockow, T. Wuttke, and B. Cooper, 'A Reference Architecture for Big Data Systems in the National Security Domain', in *2016 IEEE/ACM 2nd International Workshop on Big Data Software Engineering (BIGDSE)*, May 2016, pp. 51–57. doi: 10.1109/BIGDSE.2016.017.

[12] C. A. Ardagna, P. Ceravolo, and E. Damiani, 'Big data analytics as-a-service: Issues and challenges', in *2016 IEEE International Conference on Big Data (Big Data)*, Washington, DC, USA, Dec. 2016, pp. 3638–3644. doi: 10.1109/BigData.2016.7841029.

[13] R. S. Kalan and M. O. Ünalir, 'Leveraging big data technology for small and medium-sized enterprises (SMEs)', in *2016 6th International Conference on Computer and Knowledge Engineering (ICCKE)*, Mashhad, Iran, Oct. 2016, pp. 1–6. doi: 10.1109/ICCKE.2016.7802106.

[14] Y. Liu, A. Soroka, L. Han, J. Jian, and M. Tang, 'Cloud-based big data analytics for customer insight-driven design innovation in SMEs', *Int. J. Inf. Manag.*, vol. 51, p. 102034, Apr. 2020, doi: 10.1016/j.ijinfomgt.2019.11.002.

[15] D. Sen, M. Ozturk, and O. Vayvay, 'An Overview of Big Data for Growth in SMEs', *Procedia - Soc. Behav. Sci.*, vol. 235, pp. 159–167, Nov. 2016, doi: 10.1016/j.sbspro.2016.11.011.

[16] P. D. Vecchio, A. D. Minin, A. M. Petruzzelli, U. Panniello, and S. Pirri, 'Big data for open innovation in SMEs and large corporations: Trends, opportunities, and challenges', *Creat. Innov. Manag.*, vol. 27, no. 1, pp. 6–22, 2018, doi: 10.1111/caim.12224.

[17] C. Bange, T. Grosser, and N. Janoschek, 'Big Data Use Cases 2015 - Getting real on data monetization', Jul. 2015. Accessed: Feb. 15, 2019. [Online]. Available: http://barc-research.com/research/big-data-use-cases-2015/

[18] M. R. M. Assis and L. F. Bittencourt, 'A survey on cloud federation architectures: Identifying functional and non-functional properties', *J. Netw. Comput. Appl.*, vol. 72, pp. 51–71, Sep. 2016, doi: 10.1016/j.jnca.2016.06.014.

[19] N. Naik, 'Docker container-based big data processing system in multiple clouds for everyone', in *2017 IEEE International Systems Engineering Symposium (ISSE)*, Vienna, Austria, Oct. 2017, pp. 1–7. doi: 10.1109/SysEng.2017.8088294.

[20] B. Satzger, W. Hummer, C. Inzinger, P. Leitner, and S. Dustdar, 'Winds of Change: From Vendor Lock-In to the Meta Cloud', *IEEE Internet Comput.*, vol. 17, no. 1, pp. 69–73, Jan. 2013, doi: 10.1109/MIC.2013.19.

[21] G. C. Silva, L. M. Rose, and R. Calinescu, 'Towards a Model-Driven Solution to the Vendor Lock-In Problem in Cloud Computing', in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, Bristol, UK, Dec. 2013, vol. 1, pp. 711–716.

doi: 10.1109/CloudCom.2013.131.

[22] A. N. Toosi, R. N. Calheiros, and R. Buyya, 'Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey', *ACM Comput Surv*, vol. 47, no. 1, p. 7:1-7:47, May 2014, doi: 10.1145/2593512.

[23] D. Bernstein, 'Cloud Foundry Aims to Become the OpenStack of PaaS', *IEEE Cloud Comput.*, vol. 1, no. 2, pp. 57–60, Jul. 2014, doi: 10.1109/MCC.2014.32.

[24] A. Leung, A. Spyker, and T. Bozarth, 'Titus: Introducing Containers to the Netflix Cloud', *Queue*, vol. 15, no. 5, p. 30:53-30:77, Oct. 2017, doi: 10.1145/3155112.3158370.

[25] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, 'Elasticity in Cloud Computing: State of the Art and Research Challenges', *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 430–447, Mar. 2018, doi: 10.1109/TSC.2017.2711009.

[26] M. A. Rodriguez and R. Buyya, 'Container-based cluster orchestration systems: A taxonomy and future directions', *Softw. Pract. Exp.*, vol. 49, no. 5, pp. 698–719, 2019, doi: 10.1002/spe.2660.

[27] C. Pahl, 'Containerization and the PaaS Cloud', *IEEE Cloud Comput.*, vol. 2, no. 3, pp. 24–31, May 2015, doi: 10.1109/MCC.2015.51.

[28] C. Pahl and B. Lee, 'Containers and Clusters for Edge Cloud Architectures – A Technology Review', in *2015 3rd International Conference on Future Internet of Things and Cloud*, Rome, Italy, Aug. 2015, pp. 379–386. doi: 10.1109/FiCloud.2015.35.

[29] T. Vergilio and M. Ramachandran, 'Non-functional Requirements for Real World Big Data Systems - An Investigation of Big Data Architectures at Facebook, Twitter and Netflix':, in *Proceedings of the 13th International Conference on Software Technologies*, Porto, Portugal, 2018, pp. 833–840. doi: 10.5220/0006825408330840.

[30] G. C. Silva, L. M. Rose, and R. Calinescu, 'A Systematic Review of Cloud Lock-In Solutions', in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, Bristol, UK, Dec. 2013, vol. 2, pp. 363–368. doi: 10.1109/CloudCom.2013.130.

[31] J. Opara-Martins, R. Sahandi, and F. Tian, 'Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective', *J. Cloud Comput.*, vol. 5, no. 1, p. 4, Dec. 2016, doi: 10.1186/s13677-016-0054-z.

[32] O. Jokonya, 'Investigating Open Source Software Benefits in Public Sector', in *2015 48th Hawaii International Conference on System Sciences*, Kauai, HI, USA, Jan. 2015, pp. 2242–2251. doi: 10.1109/HICSS.2015.268.

[33] M. Palyart, G. C. Murphy, and V. Masrani, 'A Study of Social Interactions in Open Source Component Use', *IEEE Trans. Softw. Eng.*, vol. 44, no. 12, pp. 1132–1145, Dec. 2018, doi: 10.1109/TSE.2017.2756043.

[34] Y. Al-Hazmi, K. Campowsky, and T. Magedanz, 'A monitoring system for federated clouds', in *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, Paris, France, Nov. 2012, pp. 68–74. doi: 10.1109/CloudNet.2012.6483657.

[35] P. R. Palos-Sanchez, 'Drivers and Barriers of the Cloud Computing in SMEs: the Position of the European Union', *Harv. Deusto Bus. Res.*, vol. 6, no. 2, pp. 116–132, Oct. 2017, doi: 10.3926/hdbr.125.

[36] D. Zhao, M. Mohamed, and H. Ludwig, 'Locality-Aware Scheduling for Containers in Cloud Computing', *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 635–646, Apr. 2020, doi: 10.1109/TCC.2018.2794344.

[37] K. Hui, 'AWS 101: Regions and Availability Zones', *Rackspace Blog*, Feb. 16, 2017. https://blog.rackspace.com/aws-101-regions-availability-zones (accessed Feb. 22, 2019).

[38] R. Scott, 'Mitigating an AWS Instance Failure with the Magic of Kubernetes', *Medium*, Mar. 01, 2017. https://medium.com/spire-labs/mitigating-an-aws-instance-failure-with-the-magic-of-kubernetes-128a44d44c14 (accessed Jan. 24, 2018).

[39] J. Brodkin, 'Amazon EC2 outage calls "availability zones" into question', *Network World*, Apr. 21, 2011. https://www.networkworld.com/article/2202805/cloud-computing/amazon-ec2-outage-calls--availability-zones--into-question.html (accessed Feb. 22, 2019).

[40] M. Singhal *et al.*, 'Collaboration in multicloud computing environments: Framework and security issues', *Computer*, vol. 46, no. 2, pp. 76–84, Feb. 2013, doi: 10.1109/MC.2013.46.

[41] N. Grozev and R. Buyya, 'Inter-Cloud architectures and application brokering: taxonomy and survey', *Software—Practice Exp.*, vol. 44, no. 3, pp. 369–390, Mar. 2014, doi: 10.1002/spe.2168.

[42] G. Rattihalli, 'Exploring Potential for Resource Request Right-Sizing via Estimation and Container Migration in Apache Mesos', in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Zurich, Switzerland, Dec. 2018, pp. 59–64. doi: 10.1109/UCC-Companion.2018.00035.

[43] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, 'Towards Model-Driven Provisioning, Deployment, Monitoring, and Adaptation of Multi-cloud Systems', in *2013 IEEE Sixth International Conference on Cloud Computing*, Santa Clara, CA, USA, Jun. 2013, pp. 887–894. doi: 10.1109/CLOUD.2013.133.

[44] R. Bruno, F. Costa, and P. Ferreira, 'freeCycles - Efficient Multi-Cloud Computing Platform', *J. Grid Comput.*, vol. 15, no. 4, pp. 501–526, Dec. 2017, doi: 10.1007/s10723-017-9414-2.

[45] D. Weerasiri, M. C. Barukh, B. Benatallah, Q. Z. Sheng, and R. Ranjan, 'A Taxonomy and Survey of Cloud Resource Orchestration Techniques', *ACM Comput Surv*, vol. 50, no. 2, p. 26:1-26:41, May 2017,

doi: 10.1145/3054177.

[46] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3 edition. Upper Saddle River, NJ: Addison-Wesley Professional, 2012.

[47] NIST Big Data Public Working Group, 'NIST Big Data Interoperability Framework', National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 1500-6r2, Oct. 2019. doi: 10.6028/NIST.SP.1500-6r2.

[48] I. Verbitskiy, L. Thamsen, and O. Kao, 'When to Use a Distributed Dataflow Engine: Evaluating the Performance of Apache Flink', in *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, Toulouse, France, Jul. 2016, pp. 698–705. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0114.

[49] A. Verma, A. H. Mansuri, and N. Jain, 'Big data management processing with Hadoop MapReduce and spark technology: A comparison', in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, Indore, India, Mar. 2016, pp. 1–4. doi: 10.1109/CDAN.2016.7570891.

[50] M. Maier, 'Towards a Big Data Reference Architecture', Masters, Eindhoven University of Technology, Eindhoven, The Netherlands, 2013. Accessed: Dec. 07, 2020. [Online]. Available: https://pure.tue.nl/ws/files/46951182/761622-1.pdf

[51] L. Heilig and S. Voß, 'Managing Cloud-Based Big Data Platforms: A Reference Architecture and Cost Perspective', in *Big Data Management*, F. P. García Márquez and B. Lev, Eds. Cham: Springer International Publishing, 2017, pp. 29–45. doi: 10.1007/978-3-319-45498-6_2.

[52] G. M. Sang, L. Xu, and P. de Vrieze, 'A reference architecture for big data systems', in *2016 10th International Conference on Software, Knowledge, Information Management Applications (SKIMA)*, Chengdu, China, Dec. 2016, pp. 370–375. doi: 10.1109/SKIMA.2016.7916249.

[53] P. Pääkkönen and D. Pakkala, 'Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems', *Big Data Res.*, vol. 2, no. 4, pp. 166–186, Dec. 2015, doi: 10.1016/j.bdr.2015.01.001.

[54] I. Kant, *Critique of Pure Reason*, Rev Ed edition. London: Penguin Classics, 1781.

[55] L. Belli, S. Cirani, L. Davoli, L. Melegari, M. Mónton, and M. Picone, 'An Open-Source Cloud Architecture for Big Stream IoT Applications', in *Interoperability and Open-Source Solutions for the Internet of Things*, Cham, 2015, pp. 73–88. doi: 10.1007/978-3-319-16546-2_7.

[56] R. Pellegrini, P. Rottmann, and G. Strieder, 'Preventing vendor lock-ins via an interoperable multi-cloud deployment approach', in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, Cambridge, UK, Dec. 2017, pp. 382–387. doi: 10.23919/ICITST.2017.8356428.

[57] R. Scolati, I. Fronza, N. El Ioini, A. Samir, and C. Pahl, 'A Containerized Big Data Streaming Architecture for Edge Cloud Computing on Clustered Single-board Devices':, in *Proceedings of the 9th International Conference on Cloud Computing and Services Science*, Heraklion, Crete, Greece, 2019, pp. 68–80. doi: 10.5220/0007695000680080.

[58] J. Moreno, M. A. Serrano, E. Fernández-Medina, and E. B. Fernández, 'Towards a Security Reference Architecture for Big Data', in *Proceedings of the 20th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data co-located with 10th EDBT/ICDT Joint Conference (EDBT/ICDT 2018), Vienna, Austria, March 26-29, 2018*, 2018, vol. 2062. Accessed: Aug. 06, 2020. [Online]. Available: http://ceur-ws.org/Vol-2062/paper04.pdf

[59] H. Chen, J. Wen, W. Pedrycz, and G. Wu, 'Big Data Processing Workflows Oriented Real-Time Scheduling Algorithm using Task-Duplication in Geo-Distributed Clouds', *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 131–144, Mar. 2020, doi: 10.1109/TBDATA.2018.2874469.

[60] L. Dubé and G. Paré, 'Rigor in information systems positivist case research: current practices, trends, and recommendations', *MIS Q.*, vol. 27, no. 4, pp. 597–635, Dec. 2003.

[61] C. Pattinson, A. L. Kor, and R. Cross, 'Measuring Data Centre Efficiency', Leeds Beckett University, Leeds, MCDE, Aug. 2012.

[62] T. Kolajo, O. Daramola, and A. Adebiyi, 'Big data stream analysis: a systematic literature review', *J. Big Data*, vol. 6, no. 1, p. 47, Jun. 2019, doi: 10.1186/s40537-019-0210-7.

[63] M. Ramachandran and V. Chang, 'Towards performance evaluation of cloud service providers for cloud data security', *Int. J. Inf. Manag.*, vol. 36, no. 4, pp. 618–625, Aug. 2016, doi: 10.1016/j.ijinfomgt.2016.03.005.

[64] T. Ylonen and C. Lonvick, 'The Secure Shell (SSH) Protocol Architecture', RFC Editor, RFC4251, Jan. 2006. doi: 10.17487/rfc4251.

[65] 'Weave Cloud: Kubernetes Automation for Developers', *Weave Cloud*, 2019. https://www.weave.works/product/cloud/ (accessed Apr. 10, 2019).

[66] T. Vergilio and M. Ramachandran, 'PaaS-BDP - A Multi-Cloud Architectural Pattern for Big Data Processing on a Platform-as-a-Service Model', in *Proceedings of the 3rd International Conference on Complexity, Future Information Systems and Risk*, Funchal, Madeira, Portugal, 2018, pp. 45–52. doi:

10.5220/0006632400450052.

[67] T. Vergilio, *data-interpolator*. 2018. Accessed: Jun. 28, 2020. [Online]. Available: https://bitbucket.org/vergil01/data-interpolator/src/master/

[68] T. Vergilio, *energy-consumption-producer*. 2018. Accessed: Jun. 28, 2020. [Online]. Available: https://bitbucket.org/vergil01/energy-consumption-producer/src/master/

[69] T. Vergilio, *energy-consumption-simulator*. 2018. Accessed: Jun. 28, 2020. [Online]. Available: https://bitbucket.org/vergil01/energy-consumption-simulator/src/master/

[70] K. D. Bridgmon and W. E. Martin, *Quantitative and Statistical Research Methods: From Hypothesis to Results: 42*, 1st edition. San Francisco: Jossey-Bass, 2012.

[71] 'Pods - Kubernetes', *Kubernetes*, May 12, 2019. https://kubernetes.io/docs/concepts/workloads/pods/pod/#motivation-for-pods (accessed Jun. 08, 2019).

[72] M. R. Karabek, J. Kleinert, and A. Pohl, 'Cloud Services for SMEs – Evolution or Revolution?', *Bus. Innov.*, vol. 2, no. 1, pp. 26–33, Jan. 2011, doi: 10.1365/s35789-011-0005-4.

[73] I. Hamburg and M. Marian, 'Learning as a Service – A Cloud-based Approach for SMEs', Jul. 2012, pp. 53–57. Accessed: Jul. 31, 2020. [Online]. Available: https://www.thinkmind.org/index.php?view=article&articleid=service_computation_2012_3_30_10065

[74] O. Oyekola and L. Xu, 'Selecting SaaS CRM Solution for SMEs', presented at the ICIST 2020: 10th International Conference on Information Systems and Technologies, Lecce, Italy, Jun. 2020. Accessed: Jul. 31, 2020. [Online]. Available: http://eprints.bournemouth.ac.uk/33047/

[75] B. D. Martino, 'Applications Portability and Services Interoperability among Multiple Clouds', *IEEE Cloud Comput.*, vol. 1, no. 1, pp. 74–77, May 2014, doi: 10.1109/MCC.2014.1.

[76] R. Yasrab and N. Gu, 'Multi-cloud PaaS Architecture (MCPA): A Solution to Cloud Lock-In', in *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, Beijing, China, Jul. 2016, pp. 473–477. doi: 10.1109/ICISCE.2016.108.

[77] G. Finta, 'Mitigating the effects of vendor lock-in in edge cloud environments with open-source technologies', Oct. 2019, Accessed: Jul. 31, 2020. [Online]. Available: https://aaltodoc.aalto.fi:443/handle/123456789/40884

[78] T. Vergilio and M. Ramachandran, 'PaaS-BDP - A Multi-Cloud Architectural Pattern for Big Data Processing on a Platform-as-a-Service Model', presented at the COMPLEXIS 2018, Madeira, Mar. 2018.

[79] M. Cammert, J. Kramer, B. Seeger, and S. Vaupel, 'A Cost-Based Approach to Adaptive Resource Management in Data Stream Systems', *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 2, pp. 230–245, Feb. 2008, doi: 10.1109/TKDE.2007.190686.

[80] P. Li, S. Guo, S. Yu, and W. Zhuang, 'Cross-Cloud MapReduce for Big Data', *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 375–386, Apr. 2020, doi: 10.1109/TCC.2015.2474385.

[81] N. Košeleva and G. Ropaitė, 'Big data in building energy efficiency: understanding of big data and main challenges', *Procedia Eng. Mod. Build. Mater. Struct. Tech. MBMST 2016*, vol. 172, pp. 544–549, 2017, doi: 10.1016/j.proeng.2017.02.064.

[82] T. Akidau *et al.*, 'The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing', *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1792–1803, Aug. 2015, doi: 10.14778/2824032.2824076.

[83] D. N. Jha, S. Garg, P. P. Jayaraman, R. Buyya, Z. Li, and R. Ranjan, 'A Holistic Evaluation of Docker Containers for Interfering Microservices', in *2018 IEEE International Conference on Services Computing (SCC)*, San Francisco, CA, USA, Jul. 2018, pp. 33–40. doi: 10.1109/SCC.2018.00012.

Appendices

Appendix A. Supplementary Tables

Table A1
MC-BDP Reference Architectural layers

Layer	Description
Horizontal Layers	
Persistence Layer	The persistence layer consists of file stores, disk space, as well as relational and non-relational databases. This layer is used differently by components in other layers, namely nodes, containers, services and messaging layers.
Node Layer	The node layer is composed of virtual machines to which containers are deployed. In order to maximise economies of scale, MC-BDP recommends an infrastructure based on commercial clouds and, where possible, it recommends that multiple clouds be utilised simultaneously. This is a known strategy for mitigating the risk of vendor lock-in, as identified in the literature [20], [31], [76].
Container Layer	The container layer consists of containers running services based on container images. Since images contain all the necessary configuration that a service needs to run, nothing needs to be installed on the platform other than the container runtime, thus allowing consumers to compare offerings by different cloud providers more easily and to migrate to a different provider if needed.
Networking Layer	The networking layer represents a network which allows containers deployed to different nodes at different locations to communicate seamlessly. MC-BDP departs from the traditional approach of networking machines to that of networking containers.
Orchestration Layer	MC-BDP's orchestration layer is responsible for launching, stopping and managing containers in a cluster. It is therefore responsible for managing services deployed to containers, registering additional nodes from different clouds (or removing them), scaling the number of containers that a service runs on, controlling which containers run on which nodes, and monitoring the overall state of the cluster.
Service Layer	The service layer comprises the applications deployed to the container cluster, which range from smaller-scale deployments such as front-end user interfaces to larger-scale frameworks for big data processing distributed across hundreds of containers.
Vertical Layers	
Security Layer	The security layer is orthogonal to all other layers, since it is implemented in multiple contexts. Encryption, for example, can be configured independently at framework, orchestration, networking, messaging, and persistence levels. Due to the complexity inherent to the security aspect of large-scale cloud-based systems, our research recommends addressing it through a systematic and comprehensive framework which is multi-layer and multi-purpose such as the Cloud Computing Adoption Framework (CCAF) [63].
Monitoring Layer	The monitoring layer consists of services aimed at providing metrics related to the performance of specific components. Since diverse aspects of a system can and usually are monitored, this layer is also orthogonal to the others, and would likewise benefit from a systematic approach such as a multi-layer and multi-purpose framework.
Messaging Layer	The messaging layer is used primarily to facilitate the transmission of data from one system to

	another. One example of such usage in the context of streaming architectures is as a sink or output for real-time data from IOT devices, and as a source or input for big data processing frameworks. Likewise, the messaging layer could be configured as a sink for the result of the data processing performed by the big data framework, and as a source for subsequent processing by the same or different framework.

Table A2
Experimental Design for the Evaluation of MC-BDP Reference Architecture

Experimental Design	Environment setup using multi-cloud clusters of Azure and Google Cloud virtual machines.			
Experimental Setup	Experiment	Parallelism	Azure VMs	Google VMs
	Scalability			
	Single-Cloud 3 Workers	3	3	0
	Multi-Cloud 3 Workers	3	2	1
	Single-Cloud 6 Workers	6	6	0
	Multi-Cloud 6 Workers	6	2	4
	Fault Tolerance			
	Single-Cloud 3 Workers	2	3	0
	Multi-Cloud 3Workers	2	2	1
	Single-Cloud 6 Workers	4	6	0
	Multi-Cloud 6 Workers	4	4	2
	Technology Agnosticism			
	Single-Cloud 3 Workers Beam	3	3	0
	Single-Cloud 3 Workers Flink	3	3	0
	Multi-Cloud 3 Workers Beam	3	2	1
	Multi-Cloud 3 Workers Flink	3	2	1
	Single-Cloud 6 Workers Beam	6	6	2
	Single-Cloud 6 Workers Flink	6	6	2
	Multi-Cloud 6 Workers Beam	6	4	0
	Multi-Cloud 6 Workers Flink	6	4	0
	Windowing Rate vs. Resource Utilisation			
	Multi-Cloud 3Workers	3	2	1
	Multi-Cloud 6 Workers	6	3	3
	Multi-Cloud 10 Workers	10	6	4
	Container Co-location			
	1 Container per node	8	4	4
	2 Containers per node	8	2	2
	4 Containers per node	8	1	1

Table A3
Experiments Conducted for the Evaluation of MC-BDP Reference Architecture

Experiments Conducted	Scalability					
	Single-Cloud 3 Workers	Exp. 1 2 records/min	Exp. 2 2 records/s	Exp. 3 2 records/100s	Exp. 4 2 records/10ms	Exp. 5 2 records/ms
	Multi-Cloud 3 Workers	Exp. 6 2 records/min	Exp. 7 2 records/s	Exp. 8 2 records/100s	Exp. 9 2 records/100s	Exp. 10 2 records/ms

	Single-Cloud 6 Workers	Exp. 11 2 records/min	Exp. 12 2 records/s	Exp. 13 2 records/100s	Exp. 14 2 records/100s	Exp. 15 2 records/ms
	Multi-Cloud 6 Workers	Exp. 16 2 records/min	Exp. 17 2 records/s	Exp. 18 2 records/100s	Exp. 19 2 records/100s	Exp. 20 2 records/ms
	Fault Tolerance					
	Single-Cloud 3 Workers	Exp. 21 2 records/min	Exp. 22 2 records/s	Exp. 23 2 records/100s	Exp. 24 2 records/10ms	Exp. 25 2 records/ms
	Multi-Cloud 3 Workers	Exp. 26 2 records/min	Exp. 27 2 records/s	Exp. 28 2 records/100s	Exp. 29 2 records/10ms	Exp. 30 2 records/ms
	Single-Cloud 6 Workers	Exp. 31 2 records/min	Exp. 32 2 records/s	Exp. 33 2 records/100s	Exp. 34 2 records/10ms	Exp. 35 2 records/ms
	Multi-Cloud 6 Workers	Exp. 36 2 records/min	Exp. 37 2 records/s	Exp. 38 2 records/100s	Exp. 39 2 records/10ms	Exp. 40 2 records/ms
	Technology Agnosticism					
	Single-Cloud 3 Workers Beam	Exp. 41 2 records/min	Exp. 42 2 records/s	Exp. 43 2 records/100s	Exp. 44 2 records/10ms	Exp. 45 2 records/ms
	Single-Cloud 3 Workers Flink	Exp. 46 2 records/min	Exp. 47 2 records/s	Exp. 48 2 records/100s	Exp. 49 2 records/10ms	Exp. 50 2 records/ms
	Multi-Cloud 3 Workers Beam	Exp. 51 2 records/min	Exp. 52 2 records/s	Exp. 53 records/100s	Exp. 54 2 records/10ms	Exp. 55 2 records/ms
	Multi-Cloud 3 Workers Flink	Exp. 56 2 records/min	Exp. 57 2 records/s	Exp. 58 records/100s	Exp. 59 records/10ms	Exp. 60 2 records/ms
	Single-Cloud 6 Workers Beam	Exp. 61 2 records/min	Exp. 62 2 records/s	Exp. 63 2 records/100s	Exp. 64 2 records/10ms	Exp. 65 2 records/ms
	Single-Cloud 6 Workers Flink	Exp. 66 2 records/min	Exp. 67 2 records/s	Exp. 68 2 records/100s	Exp. 69 2 records/10ms	Exp. 70 2 records/ms
	Multi-Cloud 6 Workers Beam	Exp. 71 2 records/min	Exp. 72 2 records/s	Exp. 73 2 records/100s	Exp. 74 2 records/10ms	Exp. 75 2 records/ms
	Multi-Cloud 6 Workers Flink	Exp. 76 2 records/min	Exp. 77 2 records/s	Exp. 78 2 records/100s	Exp. 79 2 records/10ms	Exp. 80 2 records/ms
	Windowing Rate versus Resource Utilisation					
	Multi-Cloud 3Workers	Exp. 81 R = 2.0 5s start every 10s	Exp. 82 R = 1.5 5s start every 7.5s	Exp. 83 R = 1.0 5s start every 5s	Exp. 84 R = 0.2 5s start every 1s	Exp. 85 R = 0.1 5s start every 0.5s
	Multi-Cloud 6 Workers	Exp. 86 R = 2.0 5s start every 10s	Exp. 87 R = 1.5 5s start every 7.5s	Exp. 88 R = 1.0 5s start every 5s	Exp. 89 R = 0.2 5s start every 1s	Exp. 90 R = 0.1 5s start every 0.5s
	Multi-Cloud 10 Workers	Exp. 91 R = 2.0 5s start every 10s	Exp. 92 R = 1.5 5s start every 7.5s	Exp. 93 R = 1.0 5s start every 5s	Exp. 94 R = 0.2 5s start every 1s	Exp. 95 R = 0.1 5s start every 0.5s
	Container Co-location					
	1 Container per node	Exp. 96 R = 2.0 5s start every 10s	Exp. 97 R = 1.5 5s start every 7.5s	Exp. 98 R = 1.0 5s start every 5s	Exp. 99 R = 0.2 5s start every 1s	Exp. 100 R = 0.1 5s start every 0.5s
	2 Containers per node	Exp. 101 R = 2.0 5s start every 10s	Exp. 102 R = 1.5 5s start every 7.5s	Exp. 103 R = 1.0 5s start every 5s	Exp. 104 R = 0.2 5s start every 1s	Exp. 105 R = 0.1 5s start every 0.5s
	4 Containers per node	Exp. 106 R = 2.0 5s start every 10s	Exp. 107 R = 1.5 5s start every 7.5s	Exp. 108 R = 1.0 5s start every 5s	Exp. 109 R = 0.2 5s start every 1s	Exp. 110 R = 0.1 5s start every 0.5s

Table A4
Performance Metrics Used in the Evaluation of MC-BDP Reference Architecture

Performance Metrics	Container CPU Utilisation	Container Memory Utilisation	Container Network Utilisation	Additional Metric
	Scalability Metrics			
	Average and maximum CPU utilisation by a container during experiment execution time.	Average and maximum memory utilisation by a container during experiment execution time (in MB).	Total number of bytes transmitted and received over a network by a container during experiment execution time.	
	Fault Tolerance Metrics			
	Average and maximum CPU utilisation by a container	Average and maximum memory utilisation by a container during	Total number of bytes transmitted and received over a network by a	Records Processed Percentage of total number

	during experiment execution time.	experiment execution time (in MB).	container during experiment execution time.	of records processed compared to total number of transmitted records
	Technology Agnosticism Metrics			
	Average and maximum CPU utilisation by a container during experiment execution time.	Average and maximum memory utilisation by a container during experiment execution time (in MB).	Total number of bytes transmitted and received over a network by a container during experiment execution time.	
	Windowing Rate versus Resource Utilisation Metrics			
	Average and maximum CPU utilisation by a container during experiment execution time.	Average and maximum memory utilisation by a container during experiment execution time (in MB).	Total number of bytes transmitted and received over a network by a container during experiment execution time.	Data Volume Processed Total number of kilobytes and total number of records received by each worker for processing after the windowing function is applied.
	Container Co-Location Metrics			
	Average and maximum CPU utilisation by data processing containers running a node during experiment execution time.	Average and maximum CPU utilisation by data processing containers running a node during experiment execution time (in MB).	Total number of bytes transmitted and received over a network by the data processing containers during experiment execution time.	Data Volume Processed Total number of kilobytes and total number of records received by each worker for processing after the windowing function is applied.