A RAPID DEPLOYMENT BIG DATA COMPUTING PLATFORM FOR CLOUD ROBOTICS

Leigh Duggan[1], James Dowzard[2], Jayantha Katupitiya[3], and Ka C. Chan[4]

[1,2,3,4]Department of Mechatronic Engineering, University of New South Wales, Australia
[4]Department of Computer Science and Information Technology, La Trobe University, Australia

ABSTRACT

The primary contribution of this research is the production of a general cloud robotics architecture that leverages the established and evolving big data technologies. Prior research in this area has not released all details of their deployed architectures, which prevents experimental results from being replicated and verified. By providing a general-purpose architecture, it is hoped that this framework will allow future research to build upon and begin to create a standardised platform, where research can be easily repeated, validated and compared.The secondary contribution is the critical evaluation of the design of cloud robotic architectures. Whilst prior research has demonstrated that cloud-based robotic processing is achievable via big data technologies, such research has not discussed the choice in design. With the ecosystem of big data technologies expanding in recent years, a review of the most relevant technologies for cloud robotics is appropriate to demonstrate and validate the proposed architectural design.

KEYWORDS

Cloud robotics, big data, OpenStack, Apache, ROS.

1. INTRODUCTION

Cloud robotics possesses enormous potential for furthering the capabilities and applications of robots in existing and new challenging environments. The designed architecture detailed in this paper has the potential to dramatically change the affordability of features and capabilities currently available to high performance and expensive robots. Cloud robotics and cloud-based deployments of big data systems share similar objectives and features. The more established and mature big data frameworks have the potential to be leveraged by cloud robotics, which prior research has demonstrated through the implementation of cloud-based processing with such technologies.

Since the original Cloud Robotic framework of DAvinCi [1], few cloud robotic architectures have validated their choice of design, or explored the growing ecosystem of big data software and frameworks.

The objective of this research is to:

- Create a general cloud robotic architecture that leverages appropriate and mature big data technologies which can be used as a blueprint for cloud robotic deployments;
- For this general architecture to possess components which are flexible, relevant, affordable, well supported, and implementable on a variety of infrastructures.

The approach to achieve this objective is to:

- Examine the architecture of prior research and deployments of cloud robotics to understand and identify the main components that constitute cloud robotics;
- Examine and assess the current technologies of each component;
- Make an appropriate recommendation for technology which is most appropriate for cloud robotics.

DOI: 10.5121/ijcnc.2017.9606

2. RELATED WORK

The primary cloud robotic projects that have utilized aspects of big data include:

- DAvinCi (Distributed Agents with Collective Intelligence) project [1];
- RoboEarth Platform (including Raputya) [2] [3];
- Robot Grasping project [4].

2.1 DAVINCI

DAvinCi was a privately funded project in 2010, that utilised a cloud-based infrastructure to demonstrate the advantages of moving heavy computational processes such as SLAM and mapping to the cloud. The project utilised Robot Operating System (ROS) software for data collection from sensors and forwarding to the cloud architecture, and a Hadoop cluster to store and process this data with localisation and mapping algorithms. The project demonstrated the scalability of Hadoop, and its impact on the time to perform robotic algorithms [1].

2.2 ROBOEARTH PLATFORM

The RoboEarth Platform is a collection of software components that aims to create and facilitate the 'World Wide Web for Robots' [2]. The platform's aim was for robots to exchange their acquired data to enable robots to collectively learn from one another. Big data techniques are used to learn and extract useful information such as acquired data, and make this available to robots in RoboEarth's online database. This database houses a variety of robot-centric information including software components, maps, and task knowledge and object recognition models. Such information can be used by a robot (and its hardware) to extend a robots sensing, reasoning and functionality. Rapyuta [2] [3] is an open-source cloud computing component that creates a Platform-as-a-Service (PaaS) framework to move computational intensive activities from robots into the cloud. Each robot running Rapyuta creates a virtual container in the cloud that can be used to store data, to run computational intensive tasks, exchange information with other Rapyuta containers, and connect to the RoboEarth knowledge database [5].

2.3 CLOUD BASED ROBOT GRASPING

The Cloud Based Robot Grasping project was developed to facilitate the sharing and learning of objects and grasping functions [4]. The project allowed a robot to send a picture of an object to the cloud for processing, which would attempt to identify it within its database, returning the information on the item if found. This information allowed a robot to understand and perform the appropriate grasping manoeuvre, the results of which were stored in the cloud for future reference.

3. DESIGN JUSTIFICATION

Whilst the introduction of big data conjures the characteristics of scalability, reliability, durability and fault tolerance, the overall characteristics of an end-to-end architecture must be appropriated as not all components will require such functionality. The characteristics of a general framework come from the need to make an architecture open and accessible to the widest audience possible, and to limit barriers to its use. A general architecture should also be modular and allow for components to be interchanged with alternative or more relevant technologies as they evolve and mature. Finally, a general architecture must have relevance to the area of study, but also be trending up within their uptake and use; this brings longevity to a general architecture. The characteristics of the proposed cloud robotic architecture to adhere to is summarised in Table 1.

Table 1.Characteristics for a generic cloud robotic architecture.

Characteristics	Description
Repeatability and Accessibility	The architecture is not restricted and readily available for a free or nominal amount. Such features reduce the access barrier for future research to build upon. Accessibility also allows results to be repeatable and verified.
Modular/Flexible	Framework selection should not restrict the architecture or influence other areas of the cloud based architecture. This provides flexibility for future research to build upon.
Relevance	For components to be actively developed and supported, and will remain relevant for the foreseeable future. Existing implementations is desirable to show robustness and proven capabilities.

3.1 ROBOT COMPONENT

Utilising the robot middleware ROS (Robot Operating System) is ubiquitous when it comes to the research field of robotics. This platform is well established, actively supported, implemented upon basic robots right through to industrial implementations, and offers the greatest form of flexibility and features than any other robot middleware. Therefore, the robot component will not be exposed to an analysis.

3.2 INTEGRATION AND MESSAGE SYSTEM COMPONENT

The messaging system must satisfy a range of requirements to ensure it is suitable for a cloud robotics architecture and compatible with its interfacing components as seen in Fig. 1.

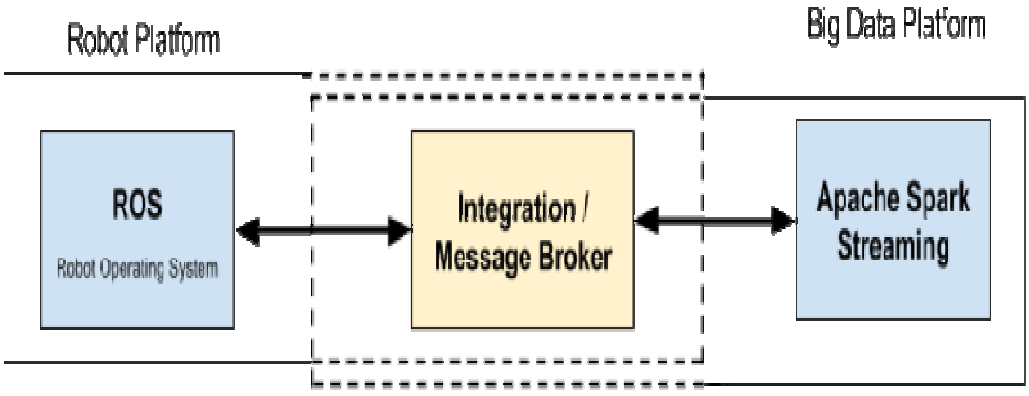

Figure 1.Messaging system framework between the ROS robot platform and Apache Spark big data platform.

The three primary solutions as compared relative to these requirements as detailed in Table 2 below.

Table 2. Requirements for the messaging system and comparison between network implementations.

#	Requirement	Standard Network Protocol	Big Data	Sensor Networks	Description
1	Event Driven	-	Y	-	For streaming systems, data must be the event which triggers data to be sent, and not utilise polling which introduces latency (i.e.publish/subscribe).
2	Durability	-	Y	-	The recovery from faults, including writing messages to memory, and/or replicating across a distributed system.
3	Versatility	Y	Y	Y	The message size and types supported.
4	Scalability	-	Y	-	The resilience to performance degradation (i.e. throughput, latency) as the volume and velocity of data increases (i.e. distributed messaging systems).
5	Availability	-	Y	-	Providing responsive recovery in the event of failure and/or system redundancy.
6	Cross-platform support	-	Y	Y	For the message system to work across multiple services, systems, platforms and enterprises. Multi-language support for clients.
7	Decoupling	-	Y	-	The ability to sustain performance in the event of slow messaging sources/destinations, or the sudden surge in velocity of messages.
8	Efficiency	Y	Y	Y	Efficiency includes the establishment of sessions, handshakes and acknowledgments (if required), and can extend to the use of compression.
9	Security/Privacy	Y	Y	-	Use of authorisation, encryption, handshakes to ensure data is protected and sent to the intended destination.

3.2.1 GATEWAY

The IoT Cloud system [6] was first developed in 2012 as an open source cloud-compatible messaging system that utilises a combination of open source software that enables movement between sensor-centric devices and cloud-based system for data processing and storage. The system allows for both data and control messages to be sent between systems, and the handling of both block and streaming data in near real-time. The architecture includes Apache Zoo Keeper for

device coordination, and a choice of supported message brokers (i.e. Active MQ, Rabbit MQ, Apache Kafka) to manage the movement of data into the cloud.

3.2.2 MESSAGE BROKER

In terms of middleware communications, there are four main options which are compared in Table 3 below.

Table 3.Comparison of middleware communication options and their functionalities.

Name	**Description**
Apache Flume	Distributed and reliable system for the aggregation of large volume and velocity data (primarily log) from many sources to big data storage.
Apache Kafka	General purpose message system that supports high throughput, reliability and horizontal scalability. Able to ingest from and publish to multiple sources.
Flakfa	Combines both Apache Kafka and Storm, utilising the versatile architecture of Kafka, and taking advantage of Flume's pre-built consumers/producers.
Qpid	Implements the Advanced Messaging Queuing Protocol (AMQP) standard, on top of a reliable, fault-tolerant and scalable platform.

From these options, Apache Flume and Apache Kafka were exposed to a deeper analysis as they are the most viable options for the messaging component of the architecture. This is detailed in Table 4 and the analysis below.

Table 4. Deeper comparison between Apache Flume and Apache Kafka.

	Apache Flume	**Apache Kafka**
Throughput	XXXX	XXXX
Integration	XXXX Offers many built-in sources and sinks	XX Does possess many multi-language clients, however more coding required
Versatility	XX	XXXX Allows multiple producers and consumers to share topics. Advantageous when data is to be consumed by multiple applications
Processing Capabilities	Simple filtering/transformation	None
Scalability	XXX Downtime required when adding new consumers	XXXX New consumers without affecting performance or downtime
Reliability & Durability	No Data is not persisted, and lost upon failure	Yes Data is persisted on Kafka cluster, and replicated across the cluster. Supports synchronous & asynchronous replication

Spike Handling	No Cannot vary speed at which data is consumed from different sources	Yes Pulls data from the topic and supports varied consumption by consumers
Community Development and Support	XXXX	XX
Maturity	XXX	XX
Embedded Hadoop Distributions	Hortonworks, Cloudera	Hortonworks, Cloudera
Applications	Application logs, sensor and machine data, geo-location data and social media	Generic publish-subscribe messaging

Apache Flume is a more mature system with greater development, support, built-in integration to sources and sinks, and basic in-stream processing, however the system is geared towards one-way ingestion to a Highly Distributed File System (HDFS) datastore. This results in a lack of reliability, durability and spike handling capabilities compared to Apache Kafka, which creates a robust and versatile messaging system. The variable speed of Apache Kafka's streams and the ease at which data can be consumed by multiple applications creates a flexible architecture in which sensor data of varying sizes and velocities can be saved or processed in different ways.
One limitation of Kafka as a general purpose messaging system is its support of only byte messages without any headers. Metadata about messages cannot be added as a header, and therefore when such metadata is required to be sent, a wrapper must beutilised such as Apache Thrift.

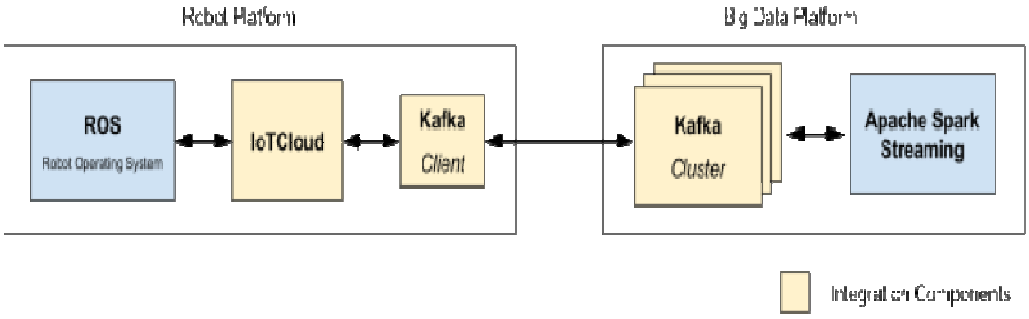

Figure 2.Updated message broker diagram demonstrating the use of IoTCloud and Apache Kafka in both client and cloud forms.

3.3 BIG DATA PROCESSING COMPONENT

3.3.1 PROCESSING ENGINE

For the last few years, debate in stream processing has been greatly contested between Apache Storm and Apache Spark Streaming. These two processing engines within the Apache ecosystem are also offered as services in the cloud, and contained within big data distributions, with both possessing their strengths and limitations. Storm and Spark among other competitors for the processing engine are compared in Table 4 below.

Table 4. Suitability of Apache products to serve as a cloud robotics engine.

#	Requirement	Cloud Robotics	Apache Spark Streaming	Apache Storm Core	Apache Storm Trident	Apache Samza
1	Data Mobility	Medium	XX	XXXX	XXXX	XXXX
2	Data Querying	Medium	XXXX	XX	XX	X
3	Handling Stream Imperfections	Low	XX	-	-	-
4	Predictable Outcomes	High	XX	XXXX	XX	XX
5	State and Stored Data	High	XXX	-	XX	XXXX
6	Data Safety and Availability	High	XXXX	XXXX	XXXX	XXXX
7	Partition and Scale (automatic)	Medium	XXX	XXX	XXX	XXX
8	Responsiveness	High	XXX	XXXX	XXX	XXX

Storm possesses superior processing speed, with sub-second latency. Spark Streaming can only offer latency measured in seconds, however, the inherent latency with network communications reduces the importance of latency for cloud robotics.

Stateful operations are the ability for an engine to possess persistent memory and utilise this within stream processing for dynamic decision making. Spark Streaming possesses stateful operations by default, however Storm requires the use of third-party components or the use of Storm Trident in order to provide statefulness. Since statefulness is not built into the Storm architecture, such customisations may not be covered by Storm's reliability and durability measures.Both Apache projects possess some of the most active Apache development communities, indicating both their popularity and relevance. Both projects are also offered in common big data distributions and are available as services by cloud service providers.

Lastly, the requirement for a consistent code-base for both batch and stream processing is becoming an important factor when selecting big data frameworks. The individual benefits that batch and stream processing provide have led to an acknowledgement that both forms of processing must exist. A single framework for query implementation and maintenance reduces the time and cost involved with development and support. Whilst this research has a focus on stream processing for initial cloud robotic processing, it is acknowledged that other forms of processing (i.e. batch, machine learning) will have their place. Storm requires development quite different from batch processing, whereas Spark provides a consistent development platform for multiple processing paradigms.

Due to Apache Spark Streaming's stateful operation, single code-base and reduced importance placed on latency, this distributed stream processing engine will be chosen as the engine for use by the cloud robotics general architecture.

3.4 CLOUD INFRASTRUCTURE

The chosen big data platform must address the shortcomings of prior cloud robotic architectures utilising big data software, which include:

- The ability of their implementation to be repeated and verified;
- Support for a single processing paradigm;
- Evolving with recent developments and improvements in the big dataecosystem.

The use of a big data distribution over individual pieces of software has many benefits for the field of cloud robotics:

- Reduces the expertise required to use big data platforms;
- Allows for rapid deployment by reducing the time required to deploy andconfigure a cluster;
- Provides enormous flexibility in terms of supporting multiple processingparadigms, and the many plugins and frameworks which aid in writing queriesand algorithms;
- Allows for deployments to be easily re-created and results verified;
- The enterprise grade stability, quality and support provides legitimacy to the architecture.

Big Data distributions also possess advantages over cloud providers, as distributions offer greater flexibility in where they can be deployed (i.e. onsite hardware, or using software such as Cloud Break to deploy on cloud providers if additional resources are required). Cloud providers also incur a cost, which would increase the barriers to a general cloud robotics architecture.

3.4.1 BIG DATA DISTRIBUTION

Of the big data distributions investigated, only Cloudera and Hortonworks offer their distributions for free. Cloudera's free distribution however is a cut-down version of the enterprise distribution, charged annually on a per node basis [7]. The free Cloudera distribution (Cloudera Express) also does not offer stream processing support, and is only supported by Amazon's AWS platform [8].

The Horton Distribution Platform (HDP) [9] is an Open Enterprise Hadoop platform, released entirely as a free and open distribution. As one of the largest contributors to the open source big data community, the company understands the core Apache big data ecosystem (i.e. Hadoop, Ambari, Storm, Spark etc), and combines these core projects into a stable platform with enterprise robustness and features. HDP is also the only distribution to currently support a non-Linux operating system in the form of the Windows platform. The distribution is available as a standalone sandbox distribution, which allows for a rapid implementation of a multi-node Hadoop cluster on a virtual machine, or as a standard distribution for custom deployments and configurations.

HDP contains both Apache Spark Streaming and Apache Kafka, and includes Apache Ambari for rapid deployment and management of the cluster. Partnering with Microsoft Azure and Rackspace to provision cloud based HDP clusters, their Cloud break software provides great flexibility when a HDP deployment is required to be hosted by a cloud provider. The complete HDP ecosystem supports core data storage and access, but also offers components for governance, integration, security and operations.

The landscape of big data has changed dramatically over the last four years, with the creation of an ecosystem of software, distributions, and cloud based providers offering big data as a service. The use of a big data distribution in a cloud robotic architecture is a novel approach, justified from the many benefits a stable, well supported and flexible platform brings to a big data deployment. HDP was selected as the most appropriate distribution for a general cloud robotics

architecture due to its distribution being free and open, created by active open source contributors, and possesses flexible and unique deployment options.

3.4.2 Cloud Deployment And Infrastructure

Horton works released Cloud Break (assisted through the acquisition of Sequence IQ) which can automatically provision and scale HDP clusters upon major cloud providers. Cloud break implements a policy-based auto-scaling system compatible with Azure, AWS, Google Cloud Platform, and providers using OpenStack, which allows HDP to elastically expand and contract on demand. Cloud break achieves this functionality through the creation of cloud provider templates which integrate the auto-scaling settings to the chosen provider, and also the creation of an Ambari blueprint which declares the cluster's architecture, which defines how the cluster will be scaled.

The advantages of Cloud Break include:

- Auto-scaling of a cluster;
- Flexibility of cloud provider to host a HDP cluster;
- Increased control of the settings of a cluster's architecture to a universalAmbari blueprint, and not embedded upon a cloud provider.

4. Complete Architecture

The developed architecture is visualised in Fig. 3 and broken down into its functional components in Table 5 below.

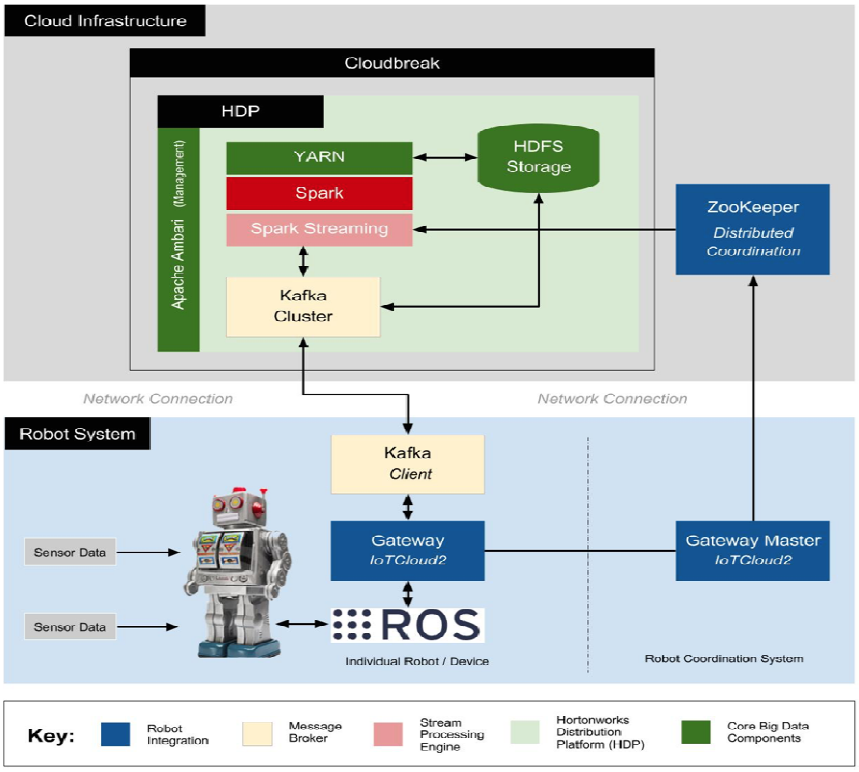

Figure 3.Physical layout of the designed architecture and the interfaces between the chosen components.

Table 5.Complete architecture breakdown with choices of each component and a description of the software's functionality.

#	Component	Software	Description
1	Robot Middleware	ROS	An established, flexible, and heavily utilised robot platform used widely from research to complex and industrial purposes.
2	Gateway	IoTCloud2 • IoTrobots • Sensorstream • ZooKeeper	Message translation that serves the robot messages to/from the message broker. Is composed of (3) components due to the complexities associated with passing data across a ROS boundary
3	Message Broker	Apache Kafka	A reliable, scalable, and fast message broker passes messages between a ROS enabled robot, and the big data stream processor and cloud storage.
4	Big Data Distribution	HDP (Hortonworks Distribution Platform)	The open enterprise big Data distribution created and supported by Hortonworks. Assembles a full big data stack into a stable and flexible platform for rapid deployment with enterprise grade features.
4.1	Data Storage	HDFS	HDP's default big data storage system
4.2	Cluster Management	Apache Ambari	HDP's cluster deployment and management tool which provides a simple web-based interface to deploy and manage a HDP cluster and all its features and services.
5	Processing Engine	Apache Spark Streaming	A big data stream processing engine that is packaged within HDP
6	Cloud Deployment/Scaling	Cloud break	Allows the provisioning, managing and monitoring of cloud-based Big Data clusters.
7	Cloud Infrastructure/Supplier	Flexible: • Own hardware • Cloud Provider	The architecture is independent of a cloud provider and can be deployed on major cloud providers (i.e. Azure, AWS, Google), or on bare metal internet connected infrastructure.

5. CONCLUSION

The use of mature big data technologies in the emerging field of cloud robotics is an area of research in its infancy. The field of cloud computing offers enormous potential to robotics by leveraging the prodigious processing resources, information and services available in the cloud, making more complex sensing and decision making accessible to low cost robots. Big data is a more mature technology that has already developed the ability to leverage the power of the cloud, and the abilities that the field of cloud robotics can benefit from.

Whilst prior research has implemented cloud robotic and architectures which utilise big data components, the architectures of instances which yielded positive results were found to be:

- Out of date;
- Non-replicable;
- Did not justify the choice of component or architecture; or
- Have not taken full advantages of changes in the big data ecosystem.

This research has examined the major components of cloud robotics and conducted a thorough analysis of available technologies, and determined the most suitable for the purposes of cloud robotics. These components were then assembled to produce general cloud robotics architecture.

The contributions made by this research include:

- Critical analysis and justification of components within a cloud robotic architecture;
- Introduction and justification of the use and benefits of an enterprise grade big data distribution for the purposes of cloud robotics;
- Composing a generic cloud robotic architecture from open source and freely distributed software, capable of running on standalone or cloud provider infrastructure;

By critically evaluating and justifying the composition and developing an easily replicable cloud robotic architecture, it is hoped that this research will modernise the existing approaches. Cloud robotics has taken with regards to big data technologies, and facilitate collaboration and verification to see this area of research mature.

6. FUTURE WORK

The development of generic cloud robotics architecture creates many avenues in which future work can be taken, in addition to emphasise the existing areas of cloud robotics which could be explored.

- Validation of the proposed cloud robotics architecture through the deployment onto hardware, implementation of a robotic processing algorithm in Apache Spark Streaming (i.e. SLAM), and comparing the performance and results achieved from SLAM implemented within ROS.
- Implementing multiple message systems and comparing performance to validate a claim by previous research that Apache Kafka may possess latency in comparison to other messaging systems (i.e. Rabbit MQ). Validating this claim may reduce the suitability of using big data technologies for cloud robotic purposes.
- Implementing different ingestion tools (Apache Flume, Apache Kafka, and Apache NiFi) for real-time and high volume data collection and streaming [10]. Evaluating limitations, identifying use cases, and comparing ease-of-use and performances of these technologies.
- Utilising the Hortonworks Sandbox distribution virtual machine image and creating a custom image is pre-configured for the purposes of cloud robotics. Much in the way that Hortonworks have created this Sandbox environment to instantly deploy a small cluster for testing purposes, A custom image for cloud robotics would instantly place a powerful big data stream processing platform at the fingertips of ROS enabled robots by just running the image in a virtual machine.
- Extending the architecture to support both Apache Spark Streaming and Apache Storm, and compare their performance for completing cloud robotic based processing. Despite the competition between these two streams processing engines, very few valid experiments exist which directly compares their performance in any field.
- A review of the key robotic based algorithms is suitable for implementation in the cloud, and comparing the suitability to existing and emerging big data processing paradigms, (i.e. Kappa and Lambda). Such research could indicate the most suitable distributed processing engine which the field of cloud robotics should adopt.

- Utilising the existing integration of IoT Cloud and its robotic libraries for Turtle bots and Parrot drones, to create a general integration package that facilitates the real-time movement of data across the boundaries of the ROS platform. A review of ROS integration found a lack of packages with this capability, with a general integration package benefitting many areas of robotics.
- Reviewing the enterprise features gained through the use of a big data distribution (i.e. authorisation, security, data life-cycle management, and assessing the relevance and possible use in the field of cloud robotics.

As the big data ecosystem continues to expand and grow with additional capabilities and processing power, the future of cloud robotics is going to be very exciting. The merging of artificial intelligence, big data, and cloud computing provides us a foundation with intelligence and computing power and resources in demand to tackle virtually any creative and unforeseen applications.

REFERENCES

[1] R.Arumugam et al., "DAvinCi: A Cloud Computing Framework for Service Robots" IEEE International Conference on Robotics and Automation, Anchorage, A.K., 2010, pp. 3084–3089, IEEE, doi [10.1109/ROBOT.2010.5509469]

[2] M. Waibel et al., "RoboEarth – a World Wide Web for Robots" IEEE Robotics and Automation Magazine, vol.18, issue: 2, June 2011. [10.1109/MRA.2011.941632]

[3] D. Hunziker et al., "Rapyuta: The RoboEarth Cloud Engine", IEEE International Conference on Robotics and Automation Karlsruhe, pp. 438–444, 2013, IEEE, doi [10.1109/ICRA.2013.6630612]

[4] B. Kehoe et al. "Cloud Based Robot Grasping with the Google Object Recognition Engine", IEEE International Conference on Robotics and Automation,Karlsruhe, pp. 21–28, 2013, IEEE

[5] G. Mohanarajah, D. Hunziker, M. Waibel, and R. D'Andrea, "Rapyuta: A Cloud Robotics Platform," IEEE Trans. Autom. Sci. Eng., pp. 1–13, Jul. 2014. doi [10.1109/TASE.2014.2329556]

[6] A.J. Fiannaca and J. Huang. "Benchmarking of Relational and NoSQL Databases to Determine Constraints for Querying Robot Execution Logs." Report, 2015, University of Washington [online] available http://courses.cs.washington.edu/courses/cse544/15wi/projects/Fiannaca_Huang.pdf

[7] Cloudera Enterprise (2015) Cloudera Data Sheet [online] available http://www.cloudera.com/content/dam/cloudera/Resources/PDF/Datasheet/datasheet-cloudera-enterprise.pdf

[8] Cloudera and Amazon Web Services, Cloudera, [online] available http://www.cloudera.com/content/www/en-us/partners/solutions/amazon-web- services.html

[9] Hortonworks homepage (2011–) [online] available http://hortonworks.com/

[10] T. Siciliani (2017) Dig Data Ingestion: Flume, Kafka, and NiFi [online] available https://dzone.com/articles/big-data-ingestion-flume-kafka-and-nifi

[11] K. Lim, J. Katupitiya, K. C. Chan, and M. Martin, "Design of an IT Capstone Subject–Cloud Robotics," Information Technology in Industry, vol. 2, no. 3, pp. 104–110, 2014.

[12] K. Lim, J. R. Carthew, L. R. Savy, J. Katupitiya, K. C. Chan, and M. Martin, "Development of a Cloud Robotic Testbed for IT Capstone Projects," Proceedings of the 9th International Conference on Information Technology and Applications (ICITA), Sydney, NSW, Australia, July 2014.