2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

DAvinCi: A Cloud Computing Framework for Service Robots

Rajesh Arumugam†, Vikas Reddy Enti, Liu Bingbing, Wu Xiaojun, Krishnamoorthy Baskaran
Foong Foo Kong, A.Senthil Kumar, Kang Dee Meng, and Goh Wai Kit

Abstract— **We propose DAvinCi, a software framework that provides the scalability and parallelism advantages of cloud computing for service robots in large environments. We have implemented such a system around the Hadoop cluster with ROS (Robotic Operating system) as the messaging framework for our robotic ecosystem. We explore the possibilities of parallelizing some of the robotics algorithms as Map/Reduce tasks in Hadoop. We implemented the FastSLAM algorithm in Map/Reduce and show how significant performance gains in execution times to build a map of a large area can be achieved with even a very small eight-node Hadoop cluster. The global map can later be shared with other robots introduced in the environment via a Software as a Service (SaaS) Model. This reduces the burden of exploration and map building for the new robot and minimizes it's need for additional sensors. Our primary goal is to develop a cloud computing environment which provides a compute cluster built with commodity hardware exposing a suite of robotic algorithms as a SaaS and share data co-operatively across the robotic ecosystem.**

I. INTRODUCTION

Service robotics is forecasted to become a US$12b industry by the year 2015, [1]. There has also been an expressed need by the governments of Japan, Korea, and the EU [2] to develop robots for the home environment. Consequently, the amount of research being done in this area has increased substantially and has taken a few distinct design directions. One design approach has been the use of a single, human-like robot with abilities to manipulate the environment and perform multiple tasks. The second approach involves the use of multiple, simple task-specific robots to perform multiple tasks in the same environment. Our design approach fuses the two approaches to create a hybrid team of distributed, networked and heterogenous agents with each agent having a unique ability or sensory perception. This is proposed as a solution for servicing large environments such as office buildings, airports and shopping malls.

A. Robots in Large Environments

A typical robot executes several primary tasks such as obstacle avoidance, vision processing, localization, path planning and environment mapping. Some of these tasks such as vision processing and mapping are computationally intensive but given the increasing speeds of current processors these can be done on the onboard computers. However, these onboard computers require dedicated power supplies, good shock protection if a Hard disk drive is used and they are responsible for a large amount of the robot's power consumption. And in a heterogenous robotic team serving a large environment, the presence of powerful onboard computers on every single robot is both cost prohibitive and unnecessary. Traditionally, in large environments, each robot would have to explore and build its own map. Without means to create a live, global map of the large environment, there is duplication of exploration effort and sensor information on the robots. Moreover, when a new robot is introduced to the same environment, it will again duplicate all the efforts of its predecessors in exploring the environment, making the system very inefficient.

The major aspect of our research involves the development of a software architecture that will enable the heterogeneous agents to share sensor data and also upload data to processing nodes for computationally intense algorithms. We have evaluated various frameworks such as Player/Stage, MS Robotics Studio and ROS and found each of them to be particularly capable in small environments. However, for large environments, these architectures need to be augmented and we propose the DAvinCi (Distributed Agents with Collective Intelligence) framework to enable teams of heterogenous robots to handle large environments.

The primary method of surpassing the challenges in large environments is to network the robots. Networked robots have certain challenges in the field of data sharing, co-operative perception and collective intelligence [3] which have been addressed by the DAvinCi framework.

The DAvinCi framework combines the distributed ROS architecture, the open source Hadoop Distributed File System (HDFS [4]) and the Hadoop Map/Reduce Framework. The DAvinCi framework is still in its early stage of development and should be largely considered as a work-in-progress. To our knowledge, it is the first system of its kind and we consider it ideal for large scale environments with infrastructure. A brief introduction to cloud computing and its benefits are described in Section 2. The DAvinCi architecture is described in Section 3 along with descriptions of Hadoop and ROS in Sections 4 and 5, respectively. We then discuss a proof-of concept implementation of Grid-based FastSLAM as a Hadoop Map/Reduce task in Section 6. We have also presented our preliminary results in Section 7. Finally, we conclude with a discussion of our future work and research direction towards completing DAvinCi.

II. CLOUD COMPUTING

Cloud computing is a paradigm shift in the way computing resources are used and applications delivered. These resources include servers, storage and the network insfrastructure along with the software applications. Cloud computing

All authors are with Data Storage Institute, A*STAR, Singapore. †Rajesh_VA@dsi.a-star.edu.sg

refers to providing these resources as a service over the Internet to the public or an organization [5]. There are three types of cloud services and they differ in the approach on how the resources are made available. The first approach is to make the hardware infrastructure available as a service and is called Infrastructure as a Service (IaaS). Amazon's EC3/S3 is an example of such a service [6]. The second approach is to provide a platform (the OS along with the necessary software) over the hardware infrastructure. This is called Platform as a Service (PaaS). An example of this kind is the Google Application Engine [7]. The third approach is to provide the application as a service along with the hardware infrastructure and is called Software as a Service (SaaS) and examples include Google Docs, ZOHO and Salesforce.com .

The cloud environment has the following advantages:

1) Make efficient use of available computational resources and storage in a typical data center.
2) Exploit the parallelism inherent in using a large set of computational nodes

Their relevance to robotics is described below. The papers [8], [9], [10] describe algorithms, techniques and approaches for a network of robots for coordinated exploration and map building. Some of these approaches can be parallelized and refined by doing parts of the map building offline in a backend multiprocessor system which will also have information from the other robots. The decision to explore a particular area can also be coordinated among the robots by a central system. In [8], the segmenting of the environment and the corresponding combining of the maps can be offloaded to a backend system. The fusion of visual processing of camera images with the laser ranger described in [11] is a computationally intensive task that can be offloaded to a backend multiprocessor system.

Many of SLAM algorithms [e.g. FastSLAM] which use particle filters for state estimation (feature sets in maps) have conditional independence among the different particle paths [12], [13], [14] and the map features. These algorithms are candidates for parallel processing in a computational cloud where the map for each particle can be estimated in separate processors thus speeding up the whole procedure. We describe such an implementation in a later section of this paper. Other than this, most of the robotic algorithms are inherently parallel computing tasks working on relatively independent data sets. Our platform therefore provides an ideal environment for executing such tasks.

The DAvinCi system is a PaaS which is designed to perform crucial secondary tasks such as global map building in a cloud computing environment.

III. DAvinCi Architecture

For large environments, we propose a team structure where the sensors are distributed amongst the members such that some have very precise localization sensors, a few others have LIDARs, a few have image acquisition sensors and all have the basic proprioceptive sensors i.e. a low-cost, single-axis gyro and wheel encoders [15]. The robots are assumed to have at least an embedded controller with Wi-Fi connectivity and the environment is expected to have a Wi-Fi infrastructure with a gateway linking the cloud service to the robots. By linking these robots and uploading their sensor information to a central controller we can build a live global map of the environment and later provide sections of the map to robots on demand as a service. A similar approach can be used for other secondary tasks such as multi-modal map building, object recognition in the environment and segmentation of maps.

Currently our DAvinCi environment consists of Pioneer robots, Roombas, Rovios and SRV-1. The ROS platform was used for sensor data collection and communication among the robot agents and clients. We make use of the Hadoop Distributed File System (HDFS) for data storage and Hadoop Map/Reduce framework for doing the batch processing of sensor data and visual information.

Figure 1 shows the high level overview of our system and how it can be accessed over the cloud. The DAvinCi server is the access point to external entities (Robots/Human interface) accessing the cloud service. It also binds the robotic ecosystem to the backend Hadoop computational cluster. The ROS framework provides a standard form of communication and messaging across the robots, between the DAvinCi server and the robots. A standard set of algorithms (SLAM, global path planning, sensor fusion) are exposed as a cloud service which can either be accessed over the intranet as in a private cloud or over the Internet with ROS messages wrapped in HTTP requests/responses.

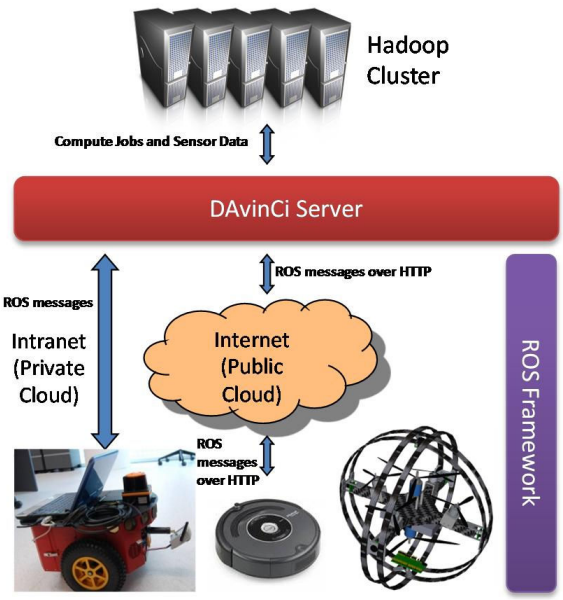

Fig. 1. *High level overview of DAvinCi.*

Figure 2 shows a high level architecture of our system. At the bottom is our robotic ecosystem which as explained before consists of Pioneer robots, SRV-1, Roomba and the Rovio. Some of these are equipped with onboard CPUs on which we run the ROS nodes with some of the existing

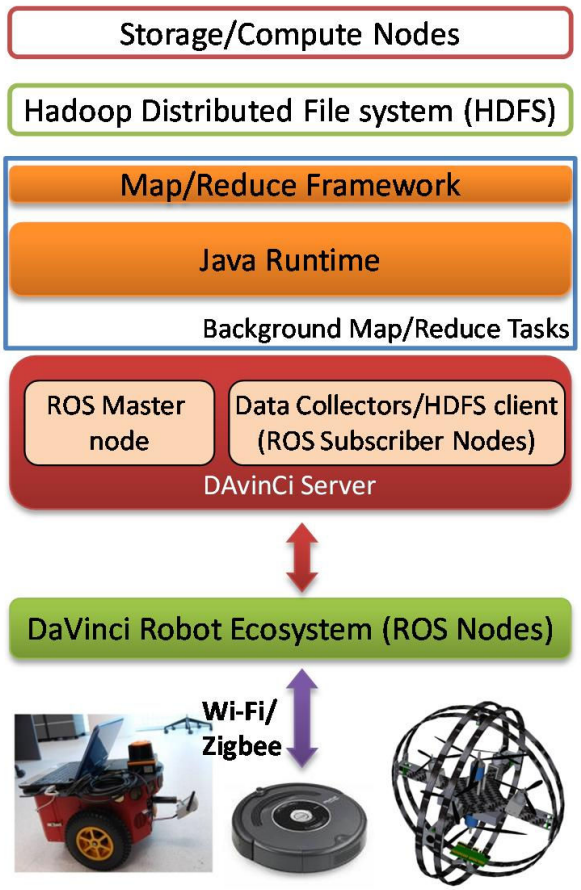

Fig. 2. *Architecture of the DAvinCi Cloud computing Platform.*

drivers. Systems like the Roomba and the Rovio does not have the provision to run as ROS nodes. For these types of robots the DAvinCi server runs proxy ROS nodes which collect sensor data or control the robot via interfaces like Zigbee, Bluetooth or through normal Wi-Fi. Above the DAvinCi server is the HDFS cluster with the Map/Reduce framework for execution of various robotic algorithms. The DAvinCi server acts as the central point of communication for the robots as well as the external agents to access the cloud services provided by the platform. The following section describes the architecture components in detail.

A. DAvinCi Server

As shown in Figure the DAvinCi server acts as a proxy and a service provider to the robots. It binds the robot ecosystem to the backend computation and storage cluster through ROS and HDFS. The DAvinCi server acts as the master node which runs the ROS name service and maintains the list of publishers. ROS nodes on the robots query the server to subscribe and receive messages/data either from the HDFS backend or from other robots. Data from the HDFS is served using the ROS service model on the server. The server collects data from the robot ecosystem through the Data collectors running as ROS subscribers or ROS recorders. This data is then pushed to the backend HDFS file system. The server triggers some of the backend Map/Reduce tasks to process the data either through explicit requests from the robots or periodically for some of the tasks. For example a robot surveying a new area can send a new batch of sensor data and trigger the map building task to refine parts of the already surveyed global map. The communication mode between the server and the robots will be standard Wi-Fi. A typical setup will have a Wi-Fi mesh network to make the connection available throughout a given area. For external entities the services are exposed by the DAvinCi server via the Internet over HTTP. The ROS messages or message bags in this case are wrapped in HTTP requests.

B. HDFS cluster

The HDFS cluster contains the computation nodes and the storage. In our setup we have a small 8 node cluster setup. Each node is an Intel Quad core server with 4GB of RAM. Currently we use only the internal storage of the servers. The HDFS file system runs on these nodes and the Map/Reduce framework facilitates the execution of the various robotic algorithm tasks. These tasks are run in parallel across the cluster as Map/Reduce tasks, thereby reducing the execution times by several orders of magnitude.

Sensor data from the robots that was pushed by the DAvinCi server are available to the Map/Reduce tasks across the cluster through the HDFS file system.

IV. Hadoop and the Map/Reduce Framework

Our platform uses the Hadoop distributed file system (HDFS) [4] for storing data from the different robots. The data can be from the sensors like laser scanners, odometer data or images/video streams from cameras. Hadoop is a open source software similar to Google's Map/Reduce framework [16]. It also provides a reliable, scalable and distributed computing platform. Hadoop is a Java based framework that supports data intensive distributed applications running on large clusters of computers. One of the main features of Hadoop is that it parallelizes data processing across many nodes (computers) in the cluster, speeding up large computations. Most of these processing occurs near the data or storage so that I/O latencies over the network are reduced. The HDFS file system of Hadoop also takes care of splitting file data into manageable chunks or blocks and distributes them across multiple nodes for parallel processing. Figure 3 shows the components of HDFS consisting of data nodes where the file chunks are stored. The name node provides information to the clients about how the file data is distributed across the data nodes to the clients.

One of the main functionality in Hadoop is the Map/Reduce framework which is described in [17]. While Hadoop's HDFS filesystem is for storing data, the Map/Reduce framework facilitates execution of tasks for processing the data. The Map/Reduce framework provides a mechanism for executing several computational tasks in parallel on multiple nodes on a huge data set. This reduces the processing or execution time of computationally intensive

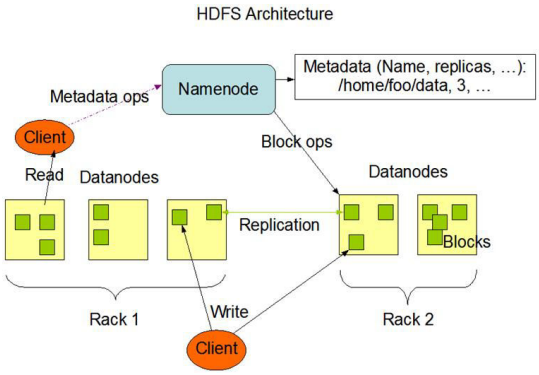

Fig. 3. *Hadoop High level Architecture.*

tasks by several orders of magnitude compared to running the same task on a single server. Even though Hadoop has been primarily used in search and indexing of large volumes of text files, nowadays it has even been used in other areas also like in machine learning, analytics, natural language search and image processing. We have now found its potential application in robotics.

Figure 4 shows an overview of how Map/Reduce tasks gets executed in Hadoop. The map tasks processes an input list of key/value pairs. The reduce tasks takes care of merging the results of the map tasks. These tasks can run in parallel in a cluster. The framework takes care of scheduling these tasks. Normal Hadoop systems uses a single pass of Map/Reduce. Here, it is shown that tasks can be broken down into multiple passes of Map/Reduce jobs. Each of the map and reduce tasks can run on individual nodes on the cluster working on different sets of the large data set.

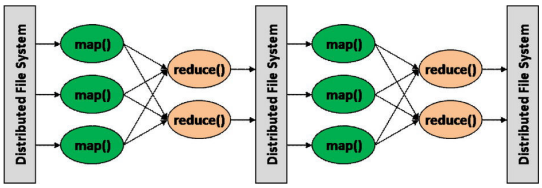

Fig. 4. *Hadoop Map/Reduce Framework.*

The Hadoop cluster in our system provides the storage and computation resources required for executing the near real time parallel execution of batch algorithms. The algorithms run as a set of single pass or multiple pass Map/Reduce tasks.

V. ROS

The ROS platform [18] is used as the framework for our robotic environment. One of the attractive features of ROS is that it is a loosely coupled distributed platform. ROS provides a flexible modular communication mechanism for exchanging messages between nodes. Nodes are processes running on a robot. There can be different nodes running on a robot serving different purposes such as collecting sensor data, controlling motors and running localization algorithms. The messaging system is based on either loosely coupled topic publish subscribe model or service based model. This is shown in Figure 5. In publish subscribe model, nodes publish their messages on a named topic. Nodes which want to receive these messages subscribe to the topic. A master node exposes the publishers to the subscribers through a name service. An example of the ROS communication mechanism is illustrated in Figure 6, showing a node publishing range scans to a topic called 'scan' from a Hokuyo laser. The viewer node subscribes to this topic to receive and display the messages. The master node acts as the arbitrator of the nodes for the publishing and subscribing for various topics. Note that actual data does not pass through the master node.

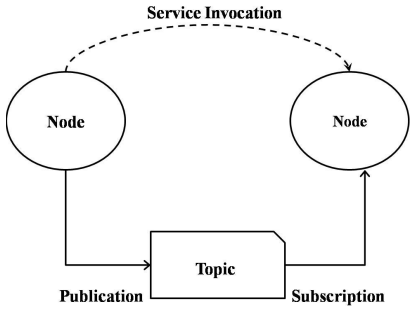

Fig. 5. *ROS messaging mechanism (From [18]).*

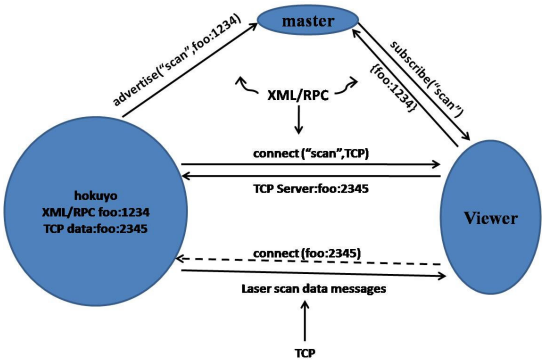

Fig. 6. *Example ROS messaging scenario (From [18]).*

We make use of the ROS messaging mechanism to send data from the robots to collectors which in turn push them to the backend HDFS file system. The collectors run ROS recorders to record laser scans, odometer readings and camera data in ROS message bags. These are later processed by Map/Reduce tasks in the Hadoop cluster.

VI. Implementation of Grid Based FastSLAM in Hadoop

We adapted the grid based FastSLAM algorithm described in [14] which is shown in Figure VI.1. We parallelized the algorithm as Map/Reduce tasks for each particle trajectories and the map estimates.

Algorithm VI.1: FASTSLAM(*From*[14])

$\{\bar{X}_t = X_t = \emptyset$
for $k \leftarrow 1$ **to** N
do $\begin{cases} x_t^{[k]} = odometry\_model(u_t, x_{t-1}^{[k]}) \\ w_t^{[k]} = measurement\_model\_correction(z_t, x_t^{[k]}, m_{t-1}^{[k]}) \\ m_t^{[k]} = update\_occupancy\_grid(z_t, x_t^{[k]}, m_{t-1}^{[k]}) \\ \bar{X}_t = \bar{X}_t + < x_t^{[k]}, w_t^{[k]}, m_t^{[k]} > \end{cases}$
for $k \leftarrow 1$ **to** N
do $\begin{cases} \text{draw } i \text{ with probability} \propto w_t^{[k]} \\ \text{add} < x_t^{[i]}, m_t^{[i]} > \text{to } X_t \end{cases}$
return $(< x_t^{[i]}, m_t^{[i]} >$ with maximum $i)$

Each Hadoop map task corresponds to a particle (k) in the algorithm. The variables $x_t^{[k]}$ and $m_t^{[k]}$ are the state variables corresponding to the robot path (pose) and the global map at time t respectively for particle k. The variable $w_t^{[k]}$ corresponds to the weight of a particular estimation of the robot path and map for particle k. This is obtained through the measurement model which calculates the correlation between the range scan and the global map estimated in the previous time step. The algorithm returns the path and map $< x_t^{[i]}, m_t^{[i]} >$ having the maximum probability $[i]$ proportional to the accumulated weight $w_t^{[k]}$. We exploit the conditional independence of the mapping task for each of the particle paths $x_t^{[k]}$ and the map features $m_t^{[k]}$. All the particle paths (1 to k) and global features $m_t^{[k]}$ are estimated in parallel by several of map tasks. A single reduce task for all the particles selects the particle path and map $< x_t^{[i]}, m_t^{[i]} >$ having the highest accumulated weight or the probability $[i]$. This is depicted in Algorithm VI.1.

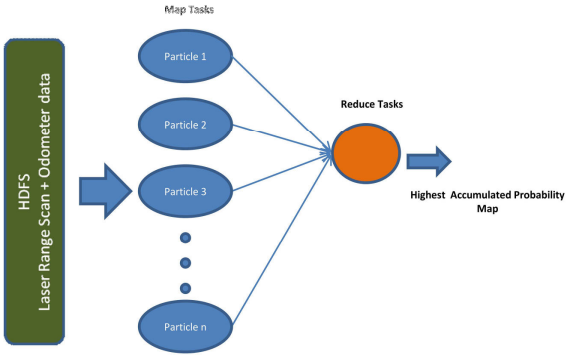

Fig. 7. *Implementation of FastSLAM in Map/Reduce Framework.*

VII. MAP/REDUCE IMPLEMENTATION RESULTS OF FASTSLAM

Figure 8 shows the graph of the time taken for execution of the algorithm for different number of particles. A dataset published in [19] was used for the map estimation. The grid map dimensions used in the algorithm was $300x300$ with a resolution of $10cm$ $(900,000$ cells). The algorithm was executed by using a single node, two-node and finally a eight-node Hadoop cluster, respectively. The execution times were calculated for each of the cases for 1, 50 and 100 particles. It can be seen that the running time decreases by several orders of magnitude as we go from a single node to an eight-node system. We believe that the execution times for mapping a large region will reduce to the order of few seconds if the number of nodes is increased further (say 16 or more). This is acceptable for a service robot surveying a large area and the surveying time is in the order of tens of minutes or more. The idea here is that even the batch mode of execution can be done in short acceptable times for mapping a large area in orders of a few seconds when a large number of nodes are used. In our case the Hadoop jobs can be triggered by map update requests from the robot itself while it is streaming the sensor data. It has been shown in [13] that the accuracy of the pose estimation reduces as the number of particles is increased. It is also show in [13] that increasing the number of particles results in increased execution time of the algorithm for a given dataset. In our case this is handled by spreading the execution across several machines in the compute cluster. It is also clear that whereas it is easier to scale the cluster it is not feasible to increase the computational capacity of the onboard system of a robot.

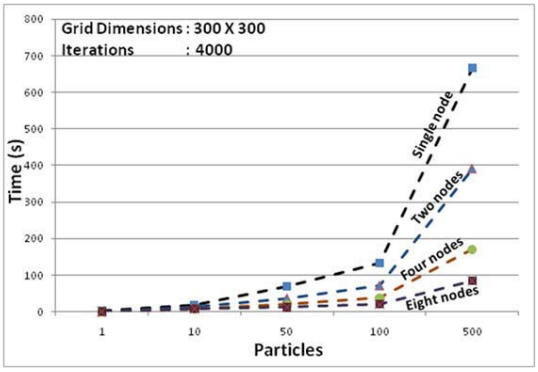

Fig. 8. *Execution time of FastSLAM in Hadoop vs. number of nodes.*

Figure 9 shows the map obtained from the data. It also shows that the pose noise reduces as the number of particles is increased to 100. The results show that a typical robotic algorithm can be implemented in a distributed system like Hadoop using commodity hardware and achieve acceptable execution times close to real time. Running the same on the onboard system of the robot might be time consuming as the single node result shows. Once we have accurate maps of such a large region, it can be shared across several of the other robots in the environment. Any new robot introduced into the environment can make use of the computed map. This is even more advantageous in some cases where the robot itself might not have an on board processor (e.g. a Roomba vacuum cleaner robot) and the DAvinCi server acting as a proxy can use the map for control and planning. Finally, as in any other cloud computing environment the computational and storage resources are now shared across a network of robots. Thus we make efficient use of the

computational and storage resources by exposing these as a cloud service to the robotic environment.

(a) 1 particle

(b) 100 particles

Fig. 9. *Estimated map (black points) with the robot path (green curves) using 1 and 100 particles (raw sensor data), respectively.*

VIII. CONCLUSIONS AND FUTURE WORK

In the paper a cloud computing architecture was proposed for service robots. The goal of this architecture is to offload data intensive and computationally intensive workloads from the onboard resources on the robots to a backend cluster system. Moreover the backend system will be shared by multiple clients (robots) for performing computationally intensive tasks as well as for exchanging useful data (like estimated maps) that is already processed. The open source Hadoop Map/Reduce framework was adopted for providing a platform which can perform computation in a cluster built on commodity hardware. A proof of concept adaptation of the grid based FastSLAM algorithm was implemented as a Hadoop Map/Reduce task. This task was on a small eight-node Hadoop cluster and promising results were achieved. Even though the FastSLAM algorithm was adapted for the Map/Reduce task, one of our primary goals is to adapt sensor fusion algorithms such as to fuse visual data with laser range scans, which is much more intensive with regard to computation and data amount.

A limitation of our design could be that we do not consider the network latencies or delays inherent in any cloud environment. We might face difficulties in transferring ROS messages involving large data (like maps and images) between the DAvinCi server and the robots. This also requires that the communication channel is reliable most of the time during such transfers. We are also working on improving the reliability and providing fail safe mechanisms for the communication between the DAvinCi server and the robots.

Our final goal is to expose a suite of robotic algorithms for SLAM, path planning and sensor fusion over the cloud. With the higher computational capacity of the backend cluster we can handle these tasks in an acceptable time period for service robots. Moreover exposing these resources as a cloud service to the robots make efficient sharing of the available computational and storage resources. As a proof of concept, this architecture will be deployed as a private cloud test bed in our office building for testing in near future.

REFERENCES

[1] ABI Research. Personal robots hit the consumer mainstream [online]. `http://www.roboticstrends.com/personal_robotics/article/personal_robots_hit_the_consumer_mainstream`, 2008.

[2] EARTO. Eu to double its r&d investment in robotics [online]. `http://www.earto.eu/nc/service/news/details/article/eu_to_double_its_rd_investment_in_robotics/`, 2008.

[3] Alessandro Saffiotti and Pedro Lima. Two "hot issues" in cooperative robotics: Network robot systems and formal models and methods of cooperation, 2008.

[4] Hadoop Distributed File System [Online] http://hadoop.apache.org/hdfs/, 2009.

[5] R. Griffith A. D. Joseph R. Katz A. Konwinski G. Lee D. Patterson A. Rabkin I. Stoica M. Armbrust, A. Fox and M. Zaharia. Above the clouds: A berkeley view of cloud computing [white paper]. `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf`, 2009.

[6] Amazon ec2. amazon elastic compute cloud [online]. `http://aws.amazon.com/ec2/`, 2009.

[7] Google app engine [online]. `http://code.google.com/appengine/`, 2009.

[8] K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. *Proc. Of Int. Conf. Intelligent Robots and Systems*, 2008.

[9] Dieter Fox. Distributed multi-robot exploration and mapping. In *CRV '05: Proceedings of the 2nd Canadian conference on Computer and Robot Vision*, pages .15–xv, Washington, DC, USA, 2005. IEEE Computer Society.

[10] W. Burgard, M. Moors, C. Stachniss, and F.E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[11] J.A. Castellanos, J.Neira, and J.D.Tardos. Multisensor fusion for simultaneous localization and map buildingg. *IEEE Transactions on Robotics and Automation*, 17(6):908–914, 2001.

[12] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[13] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[14] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[15] Vikas Reddy Enti, Rajesh Arumugam, Krishnamoorthy Baskaran, Bingbing Liu, Foo Kong Foong, Appadorai Senthil Kumar, Dee Meng Kang, Xiaojun Wu, and Wai Kit Goh. Tea table, come closer to me. In *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 325–326, New York, USA, 2009.

[16] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. pages 137–150.

[17] Hadoop map reduce framework [online]. `http://hadoop.apache.org/hdfs/`, 2009.

[18] The robotic operating system [online]. `http://www.willowgarage.com/pages/software/ros-platform`, 2009.

[19] Radish: The robotics data set repository [online]. `http://radish.sourceforge.net/index.php`, 2009.