Quantifying the Impact of Standards When Hosting Robotic Simulations in the Cloud

Sekou L. Remy

School of Computing,
Clemson University
Clemson SC
`sremy@clemson.edu`

Abstract. Cloud computing has the ability to transform simulation by providing access to computation remotely. The transformations are not without cost however. The physics-based simulations required in robotics are sensitive to timing, and given the complexity of the operating environments, there are many reasons for a roboticist to be concerned.

In this work we explore the impact of the cloud, web, and networking standards on the control of a simulated robot. Our results show that, on average, there is a noticeable impact on performance, but this impact is not statistically significant in five of the six considered scenarios. These results provide support for efforts that seek to use the cloud to support meaningful simulations. Our results are not globally applicable to robotics simulation. When using cloud-hosted simulations, roboticists yield fine tuned control of the environment, and as such there are some simulations are simply not viable candidates for this treatment.

1 Introduction

Realistic computer based simulation has been an effective tool – in both academia [1,2] and industry [3,4] – to explore content that is difficult to recreate in the physical world (safety, distance), challenging to understand because of scale, or even impossible to occur under easily reproducible conditions. Simulation technology benefits from advances in computing and information technology, but only if these advances can be incorporated in effective and timely ways.

Cloud computing is a "model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources" [5]. It is one example of technology that has the potential to impact simulation, especially in scenarios where the simulations are computationally intensive or data intensive. It also bears promise for simulations that are difficult to manage, or administer. Cloud computing is just one (set) of a host of standards or pseudo-standards that impact the way that simulation is developed and deployed. These standards are often developed in different communities and as such, it is not uncommon that multiple efforts to accomplish the same aim exist. It is critical that developers try the tools (and standards) that are developed in other communities, and comprehensive understanding of these tools is the doorway to such cross-community participation.

Cloud computing, while an example of exactly such a standard, also provides a means for this participation to occur. It can do so by providing shared access and management of tools, algorithms and data (more broadly, resources). In this work we consider an application of cloud computing to share open-source simulation environments. These simulation environments are built upon several standards (and pseudo-standards) themselves. Our aim is to identify these standards, so as to promote their use, and to also study how two sets of standards, (cloud computing and web standards) can impact the use of the overall simulation that they facilitate.

In robotic simulation, time has a higher priority than in typical computing scenarios. Sharing much in common with controls [6], timing variability and latency both impact the operation of simulations (and physical robots too), but are often invisible properties. Since many of the standards that we consider in this work add a layer of abstraction and time overhead, the impact of time must have some effect, and it is our aim to begin the process of quantifying how significant this impact is for the many combinations of possible configurations.

2 Background

The Defense Advanced Research Projects Agency (DARPA) Virtual Robotics Challenge (VRC) is an exciting competition slated to occur June 10 – 24, 2013. In support of this challenge, the agency has funded the creation of a "cloud-based, real-time, operator-interactive virtual test bed that uses physics-based models of inertia, actuation, contact and environment dynamics". More simply stated, DARPA has funded the development of cloud-hosted robotic simulation environment. This type of simulation is an avenue to increased access to meaningful interaction with robotics, and has the promise to stimulate the robotics industry worldwide.

This cloud-hosted simulation leverages Infrastructure as a Service (IaaS) to provision a virtual machine on which well vetted open source simulation can be hosted. IaaS in essence provides virtualization of operating systems and the infrastructure to run them on, including both networking and storage. Currently there are scores of IaaS providers, and in many ways this cloud model is one of the most readily understood "cloud standards". While there are differences in the configuration processes, and in payment options, all of the provided offerings enable the same basic premise: remote access to an operating system. The value proposition of IaaS is interesting beyond defense related research. The prospects of scale, on-demand flexibility, managed hosting, are only but a few of the reasons that this approach is also quite relevant to education and training.

Cloud computing is just one of the developments in Internet and Computing Technologies that are exciting for robotic simulation. The falling costs of memory, increasing clock cycles, the advent of manycore machines, and the mash-up of all of these in modern graphics processing units (GPUs) have all been good things for simulation. They have helped to make high quality, feature rich and photo realistic, real-time physics-based simulations possible.

The process of hosting and accessing remote simulations in cloud infrastructures is not without challenges. For instance, cloud technologies are inherently networked and as such, their use is subject to the overhead and uncertainty associated with several formal and informal standards. Examples of these standards include TCP [7] and HTTP [8] which are networking protocols, HTML [9] and ECMAscript [10] which enable content to be rendered and processed on remote clients, and URI [11] and XMLHttpRequest [12] which facilitate the transfer of data over the network. In addition to these formal standards, there is a powerful set of informal standards (or pseudo-standards) that exist in the web ecosystem. These resources, like web servers, web browsers and URL libraries like CURL, support defined standards and serve as powerful cross platform pathways to enable modern software development.

In our context, these are the core standards with which we are concerned. These standards provide tremendous flexibility in development and delivery of these simulations, but many in the controls community are wary about the impact that they have on the dynamics of the overall simulation and control of the robot.

For well over a decade, the networked controls community has developed approaches to control systems in the presence of delay and jitter associated with network use in the control systems [6]. The remote simulation of physical systems, especially when the simulation is performed on a cloud-hosted virtual machine has the potential to amplify the challenges faced thus far, hence the reticence displayed by the community is not without just cause. These challenges can be detrimental to the usability of the cloud-hosted simulation, and can result in inconsistencies in applications like the DARPA VRC.

3 Methodology

The scope of our investigation is focused on the impact that the suite of relevant standards can have on remote simulation. We do not seek to address the concerns of the merits of simulating a robot; instead we are concerned with the potential benefits (or drawbacks) on performance when these simulations are moved to the cloud. To that end, we begin by exploring a configuration similar to that devised for the DARPA Challenge. It involves the use of the Robot Operating System (ROS) infrastructure [13], coupled with a third party simulation environment. ROS is a high-quality open source “meta operating system” that provides the core infrastructure for message/services and abstractions of other useful robotic resources.

We initially intended to study the use of Gazebo [14] with ROS (as used in the Challenge), but it was observed that the simulation environment crashed randomly in the cloud infrastructure (See Fig. 1). The simulation environment did not demonstrate this instability in hardware, and for this reason we chose to replace Gazebo with the Stage simulator [15] and limit the control to the two dimensions permitted by this third party tool.

It is our aim to quantify the impact of the use of standards on performance of a robotic simulation, and to uncover whether that impact has a significant effect

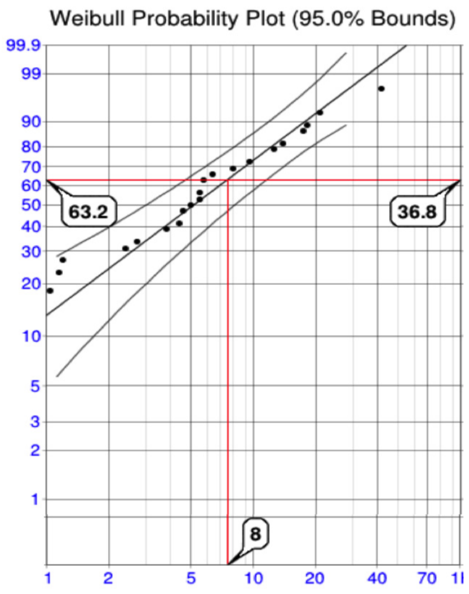


Fig. 1. Distribution of time to failure for this cloud-hosted Gazebo simulation

on the performance of a mobile robot during a navigation task. Time is at the root of the impact of these standards. The effect of time with web standards is associated with delay while with IaaS (cloud computing), it is directly linked to the approximation of time that comes from hardware virtualization. To explore these properties we use the following seven scenarios depicted in Fig. 2.

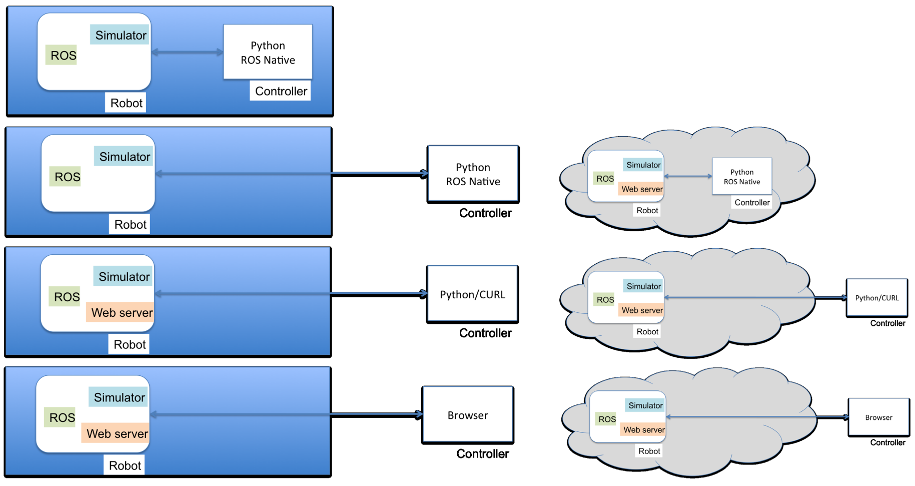
For each of the scenarios, the client issues velocity commands to the simulated robot over a duration of 200 minutes. The values for the X and Y components of the velocity are derived from (1) and (2). In these equations t is time in nanoseconds. The commands are issued every 1.05 seconds (as measured by the client process). The client is limited by the OS capabilities, in addition to the limitation of the clock that the OS is accessing.

$$v_x = .3 * cos(0.2 * t) \tag{1}$$

$$v_y = .2 * (1.5 * cos(0.3 * t) + .5 * cos(-0.1 * t)) \tag{2}$$

The simulation environment is updated at 10Hz. The simulation does not include any obstacles, so the motion of the robot is only influenced by the velocity commands issued by the client controller. There is a 0.2 second timeout applied to account for potential for disrupted communication between the controller and the simulation environment. All of these timing constraints are subject to the operating environment on the simulation computer.

The webserver used to provide access to the simulation makes use of webpy framework (webpy.org). This server is implemented as a ROS package, and provides an API to convert velocity commands in the form of a query-encoded string



(a) Simulation in hardware (b) Simulation in cloud

Fig. 2. The seven scenarios where the robotic simulation was run on a hardware platform as well as in a cloud-hosted virtual machine (provisioned through IaaS). The scenarios also feature different combinations of web standards that are relevant in the delivery of remote simulation to end-users.

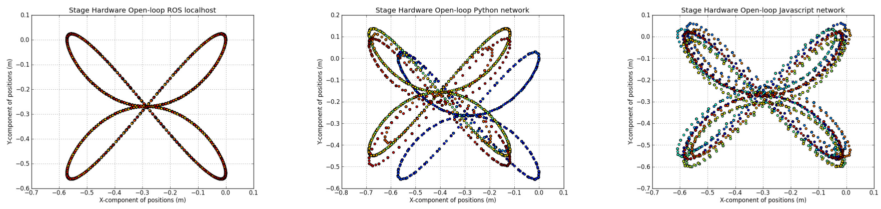
to Twist messages [16], which are then published to ROS. Twist messages are a ROS standard message type in ROS that defines the linear and angular component of velocity. This web server also hosts the HTML page that is downloaded and "run" by the browser on the client machine to run a controller in-browser. Any scenario that uses the web server to communicate with the robot also makes use of the Hypertext Transfer Protocol (HTTP) communication protocol. In this work HTTP uses TCP to actually facilitate the communication over the network.

The baseline for comparison will be the hardware-based simulation that uses a ROS-native controller. This baseline, and in fact, any scenario that features a ROS-native controller, directly uses TCP for network communication (and there is no additional communication overhead).

4 Results

In this work successfully capture the impact of the use of web and cloud standards on performance of a navigation task. In this section we first take a qualitative view, and then proceed to the quantitative assessment of the impact.

Figure 3, shows the trajectories of the robot under the baseline scenario, as well as when the hardware simulation is controlled with Python and a web browser over the network. These trajectories capture the robot's motion through nine cycles of the "butterfly" pattern. These three figures each capture the same number of samples of the trajectory of the robot. From these plots it can be seen that there are different types of errors evident in performance.



(a) Baseline (b) Python/CURL (c) Browser

Fig. 3. Qualitative view of the impact of various standards on performance

For the baseline scenario (ROS native client communicating over TCP with the simulation in hardware), there is no noticeable deviation from the target trajectory. For the other two scenarios however this is not the case. Comparing Figs. 3b & 3c, the data also appears to belie a difference in the types of deviations from the target. For example, the error observed for the Browser based controller varies in a uniform manner, while the Python/CURL controller appears to perform more consistently, except for jolts that move the robot from the expected position.

It should be clear from the data that the use of different standards (browser, URI, etc.) has visibly different effects on the observed movement of the robot. This is as we expected as the use of different standards is coupled with varying impact on time. Since time is modeled as fixed increments, when it varies, this changes the actual position of the robot in space, and thus this deviation is quite evident.

Focusing on the X-component of the robot's position, this qualitative assessment can be readily quantified. We recall that the X-component target velocity for the robot is defined as a pure sinusoid (1). The derivative of a pure sinusoid is another pure sinusoid of the same frequency but with a different phase. This means that it is possible to determine the intended position of the robot at the time that the position was actually sampled. The timing on the simulation machine influences the sampling time, so only approximations of the desired 10Hz update cycle are actually obtained.

To quantify the performance in the presence of different standards, error is measured. This error is defined as the difference between the target position and the actual position over a fixed interval (see Fig. 4). Using this formulation, the performance can be quantified and compared. For the following data, the positions were partitioned into 120 contiguous windows and the errors calculated. Each window (of length 550 samples) approximated the completion of one cycle of the "butterfly" pattern.

Because networking is required to access simulation in the cloud, it is noteworthy that while there is an impact of the use of the network, that impact is not statistically significant (See Fig. 5a). Moreover, the effect of the network appears to be a large contributing factor to negative effects on performance

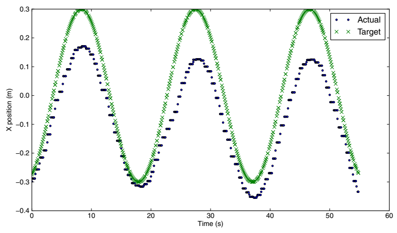
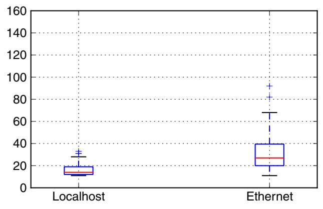
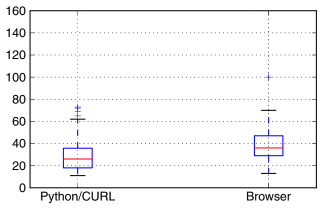


Fig. 4. Example of the difference between target and actual position during velocity based control of the simulated robot



(a) Impact of network

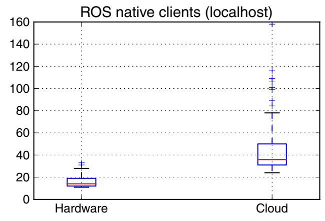


(b) Impact of web technologies

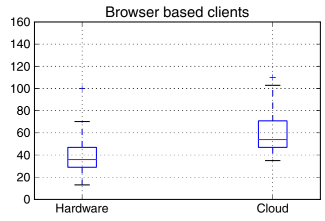
Fig. 5. Impact of web/network standards on velocity control of the simulated robot on a hardware platform

since when the web technologies are applied (Fig. 5b), there is no significant additional impact on performance with this metric. Fig. 5b also allows us to observe the impact of browser technologies on performance. The browser, using the same URI/HTTP access to the webserver, on average performs worse than the Python/CURL client. The takeaway from the results so far is that for remotely accessed simulations, the browser and the URI encoded access of the webserver both have a noticeable impact (on average), but this impact does not result in a statistically significant degradation of performance for the considered this task.

We next shift focus slightly to the impact of the final standard, the use of cloud virtualization as a substitute for hardware. To explore this impact we consider two types of cloud usage: one with no external network traffic and another with traffic initiated from a browser based client. As seen in Fig. 6, performance in both of these scenarios is impacted by the use of the cloud virtualization infrastructure. Fig. 6a shows this impact without concern for an intranetwork connection between client and robot simulation. This data suggest that the distributions of the error (a measure of performance) are again not statistically significant. The same pattern is seen when comparing the impact of virtualization on the browser-mediated control (Fig 6b). So again, the use of the standard (in this case cloud virtualization) has an impact, but not one that is statistically significant.



(a) ROS native clients



(b) Web technologies

Fig. 6. Impact of cloud virtualization (IaaS) on velocity control of the robot

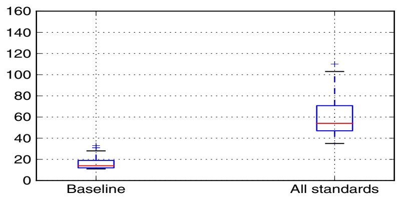


Fig. 7. Comparison of the use of all the considered standards and the baseline

These results effectively quantify the impact of the use of standards on performance of a robotic simulation. They also show that in five of the six scenarios, the impact of the use of standards has no significant effect on the performance of a mobile robot during a navigation task. By this metric, the only case where the difference is significant is when comparing the use of all of the standards to the use of none of the standards (Fig. 7).

5 Summary and Future Work

In this work we have applied several standards and pseudo-standards that can be used to amplify the potential of robotics simulation, and we have quantified how the performance of the open-loop control systems were impacted. Our results show that, as expected, on average the use of (web and cloud) standards does impact performance. There are indeed differences between running the robotic simulation in a cloud virtualization and running the same simulation in hardware. Some of these differences are well understood because of experiences gained from the networked controls community, but the effects of all of these new combinations of component/technologies are at present not well characterized.

Moreover, our results also confirm our hypothesis that the impact is not statistically significant in five of the six considered cases when compared to a baseline simulation. This baseline simulation used communication that bypassed the network interface hardware. With the metrics considered in this work, the differences in performance for any scenario that used the network interface were not significant. The implications are good for the use of cloud-hosted robotic simulation and motivate future work in the following directions.

First, since the browser has been shown to be a viable interface for remote access of the simulation, this means that it provides a path for human studies testing with remote subjects teleoperating mobile robots. Browsers like Chrome now have gamepad APIs and speech to text built-in so such resources reduce the burden on the client side code needed for studies. Of specific interest with user studies is exploring the impact of these standards on human performance of teleoperation tasks. Also, we would be interested in the potential variations due to network conditions, and the impact on performance.

The second area of interest includes increasing the operating frequencies by two orders of magnitude. Studies performed at such frequencies with cloud-hosted simulations over the network will help to quantify the impact of the use of these standards in cases where more autonomy is needed. For example, the PUMA robotic arm [17], a device often used when learning kinematics and controls, operates at 100Hz. If our results were not favorable, this could have provided a critical nail in the coffin as open-loop control at low frequencies provide the best opportunity to experience the negative impacts on performance.

References

1. Gourdeau, R.: Object-oriented programming for robotic manipulator simulation. IEEE Robotics Automation Magazine 4(3), 21–29 (1997)
2. Corke, P.I.: Robotics, Vision & Control: Fundamental Algorithms in Matlab. Springer (2011)
3. Ma, O., Buhariwala, K., Roger, N., MacLean, J., Carr, R.: MSDF – A generic development and simulation facility for flexible, complex robotic systems. Robotica 15, 49–62 (1997)
4. Lerner, M.A., Ayalew, M., Peine, W.J., Sundaram, C.P.: Does training on a virtual reality robotic simulator improve performance on the da Vinci surgical system? J. Endourol. 24(3), 467–472 (2010)
5. Mell, P., Gance, T.: The NIST Definition of Cloud Computing (September 2011)
6. Lin, H., Zhai, G., Antsaklis, P.: Robust stability and disturbance attenuation analysis of a class of networked control systems. In: 42nd IEEE Conference on Decision and Control, vol. 2, pp. 1182–1187 (December 2003)
7. Postel, J. (ed.): Information Sciences Institute: RFC 793 (1981), `http://www.ietf.org/rfc/rfc793.txt`
8. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol – http/1.1 (1999)
9. Raggett, D., Le Hors, A., Jacobs, I.: HTML 4.0 specification. World Wide Web Consortium, Recommendation REC-html40-19980424 (April 1998)
10. International, E.: Ecma-262 ecmascript language specification 5.1. JavaScript Specification, 1–245 (June 2011)
11. Berners-Lee, T., Fielding, R., Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax, RFC3986 (2005)
12. van Kesteren, A. (ed.): Information Sciences Institute: XMLHttpRequest Living Standard (February 2013), `http://xhr.spec.whatwg.org`

13. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: An Open-Source Robot Operating System. In: ICRA Workshop on Open Source Software (2009)
14. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2149–2154 (2004)
15. Vaughan, R.: Massively multi-robot simulation in stage. Swarm Intelligence 2(2), 189–208 (2008)
16. Foote, T. (ed.): ROS: geometry_msgs/Twist Message (February 2013), `http://www.ros.org/doc/api/geometry_msgs/html/msg/Twist.html`
17. Hayati, S., Lee, T., Tso, K., Backes, P., Lloyd, J.: A unified teleoperated-autonomous dual-arm robotic system. IEEE Control Systems 11(2), 3–8 (1991)