

Performance issues in cloud computing for cyber-physical applications

Michael Olson, K. Mani Chandy
Computing + Mathematical Sciences
California Institute of Technology
Pasadena, USA
molson@cs.caltech.edu, mani@cs.caltech.edu

I. INTRODUCTION

A focus of the paper is on the performance of cloud computing servers in cyber-physical systems (CPS) applications that aggregate data over large spatial regions over long spans of time. Examples of such CPS applications include those that: sense and respond to change in the environment such as forest fires or mud slides; help utilities such as water management districts monitor water quality and flow; and traffic management systems. This paper evaluates the benefits of architectures based on Cloud computing systems for such CPS applications. We use earthquake detection as a concrete example for this application class.

The focus of the paper is on the performance of Platform-as-a-Service systems. The paper described the relative advantages of different types of services in the next few sections.

II. CLOUD COMPUTING SYSTEMS

Cloud computing systems can be characterized as Software-as-a-Service, Platform-as-a-Service, and Infrastructure-as-a-Service providers[1]. A service framework must be developed for sensor networks to make SaaS platforms are not a viable option for cyber-physical applications. Both PaaS and IaaS are viable architectures for CP systems.

IaaS applications provide system infrastructure as a service; they can allocate physical resources programmatically to serve their application's needs. IaaS applications are exemplified by Amazon's Elastic Compute Cloud (EC2). EC2 allows developers to allocate any number of systems to their application by spawning and destroying instances using API calls. According to Amazon's FAQ, "It typically takes less than 10 minutes from the issue of the RunInstances call to the point where all requested instances begin their boot sequences." [2]. Responses to crises, such as earthquakes, must be executed in seconds.

PaaS applications provide a more constrained environment than IaaS. PaaS applications are represented by Google's App Engine (GAE) in which an instance is not a physical machine, but rather a running instance of an application such as a specific Java Virtual Machine (JVM) running on a particular server. Physical machines may host multiple instances, and this fact is what provides GAE's primary

benefit: instance startup time is measured in seconds. Because machines have already booted up prior to the time that activation is required, an application can be initialized merely by downloading its code to the target machine. This makes PaaS suited to handling demand that varies quickly.

III. CSN OBSERVATIONS

A. Community Seismic Network

We are building the Community Seismic Network (CSN) to: (a) provide warning about impending shaking from earthquakes, (b) guide first responders to areas with the greatest damage after an earthquake (c) obtain fine-granularity maps of subterranean structures in areas where geological events such as earthquakes or landslides occur, and (d) provide detailed analysis of deformations of internal structures of buildings (that may not be visible to inspectors) after geological events. The CSN is a challenging case study of community sense and response systems because the community has to be involved to obtain the sensor density required to meet these goals, the benefits of early warning of shaking are substantial, and frequent false warnings result in alerts being ignored.

CSN's pursuit of high sensor densities leads to one of its key design characteristics: scalability. Google App Engine is used because of its ability to scale in small amounts of time from using minimal resources to consuming large amounts of resources. During quiescent periods the only data sent on the network is control traffic, which amounts to very little; however, the data sent during seismic events is substantial. We ran simulations to estimate the traffic load of a dense network. The results are shown in Figure 1. For a network of 10,000 sensors, we expect the number of queries per second (QPS) the server must handle when sensors detect a magnitude 5 earthquake to reach a maximum rate of 423 QPS for an earthquake 50 km distant to the center of the network and a maximum rate of 2,289 QPS for an earthquake centered relative to the sensor network.

B. Performance

1) *Loading requests:* The biggest impact on system performance is the occurrence of loading requests, which occur when a request causes a new instance to be created to serve it.

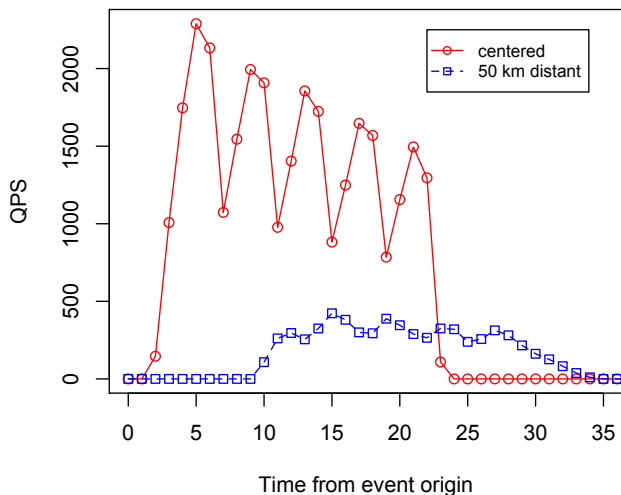


Figure 1. Estimates of the amount of server traffic generated by a magnitude 5 earthquake at different distances from the sensor network.

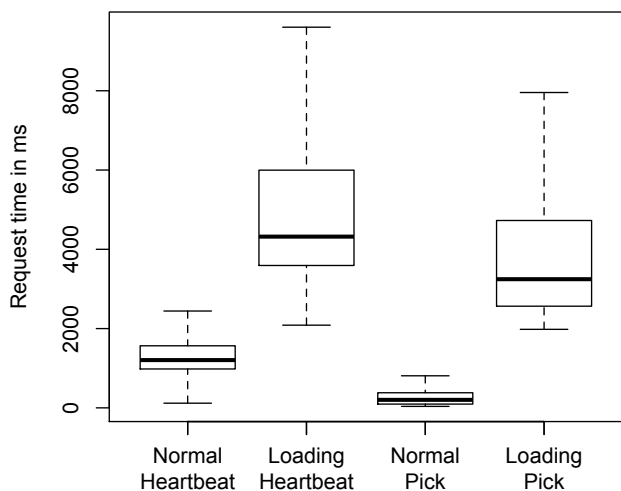


Figure 2. Latency for pick and heartbeat messages for both normal and loading requests.

We took measurements on 120,104 heartbeat messages over 94 days. Of those requests, 33,921 were loading requests, giving a loading request incidence rate of 28.24%. Given the small volume of our current network and the relatively random occurrence of heartbeats, this is not a surprising incidence rate. The detection of anomalous readings, called a "pick", are more correlated in time but occur less frequently. There were 10,454 pick messages over the same time period with 2,277 loading requests for an overall loading request incidence rate of 21.78%. Because other types of network requests in our network are extremely highly correlated in time, the overall incidence rate of loading requests was only 7.34%.

As you can see from figure 2, the penalty that the CSN

application currently pays for a loading request is quite high. That is, to process even a simple message, such as a pick, the median processing time increases from 202 ms to 3,246 ms, a difference of over 3 seconds. Heartbeats paid a similar penalty, with the median processing time increasing from 1,205 ms to 4,320 ms.

In the version of our application from which this information was derived, heartbeat and pick messages, despite structural dissimilarity, shared a common entry path in the application. It is therefore expected that the penalty paid by both messages would be similar.

2) *Error rates:* Error rates are another important factor. The most common type of error caused by App Engine's environment are deadline exceeded errors. These occur when requests are terminated for exceeding the processing deadline imposed by App Engine. The extreme variability in the processing of a request cannot be reasonably attributed to developer code, but rather to conditions within the cloud system. This is one side effect of sharing servers.

For instance, heartbeat requests that reach a warm instance have a median processing time of 1,205 ms. This might sound like a lot, but in the current stage of the network, in addition to normal processing we submit a 1 MB waveform with every heartbeat that is processed for validity before it is stored. The first quartile of heartbeat processing time is 981 ms while the third quartile is 1,566 ms, giving an interquartile range of only 585 ms. The standard deviation, however, is 3,511 ms, which gives an idea of the dispersion of the remaining values outside the interquartile range. This still leaves the 30 second cutoff at 8.2 standard deviations above the median. Given that a full 1.03% of non-loading heartbeat messages result in a deadline exceeded error, this is a clear indication that the volume of deadline exceeded requests is far too large to be attributable to any reasonable variation in processing time.

IV. CONCLUSION

In conclusion, we find that while PaaS applications in general and Google App Engine in particular can fulfill the needs of cyber-physical systems, it's important to pay close attention to the design characteristics of the chosen platform. The performance implications of even seemingly small or obvious choices can make substantial differences in how applications behave in the long term.

REFERENCES

[1] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," *International Joint Conference on INC, IMS and IDC*, Jan 2009.

[2] Amazon. (2011, 2) Amazon ec2 faqs. [Online]. Available: <http://aws.amazon.com/ec2/faqs/>