SaTScan on a Cloud: On-Demand Large Scale Spatial Analysis of Epidemics

Ronald C Price [1], Warren Pettey [2], Tim Freeman [3], Kate Keahey [3], Molly Leecaster [2], Matthew Samore [2], James Tobias [4], Julio C Facelli [1,5]

[1] Center for High Performance Computing, Departments of Internal Medicine[2] and Biomedical Informatics[5], The University of Utah

[3] Mathematics and Computer Science Division of Argonne National Laboratory

[4] National Center for Public Health Informatics, Centers for Disease Control

Abstract: ***By using cloud computing it is possible to provide on- demand resources for epidemic analysis using computer intensive applications like SaTScan. Using 15 virtual machines (VM) on the Nimbus cloud we were able to reduce the total execution time for the same ensemble run from 8896 seconds in a single machine to 842 seconds in the cloud. Using the caBIG tools and our iterative software development methodology the time required to complete the implementation of the SaTScan cloud system took approximately 200 man-hours, which represents an effort that can be secured within the resources available at State Health Departments. The approach proposed here is technically advantageous and practically possible.***

Introduction

SaTScan [1] is a computer intensive application that is commonly used to detect cluster characteristics of epidemics that provide decision support to epidemiologists. In practical applications long ensemble runs of SaTScan provide public health analysts with insight into the epidemics' progression that result in higher confidence policy decisions. SaTScan ensemble runs test the alternative hypothesis that there is elevated disease risk within a defined cluster. The estimated p-value for these tests is based on the rank of the likelihood from the real data compared to that from the random data sets generated during the Monte Carlo randomizations. This rank is conditional on the random data sets generated and if the random seed were not set to a constant would vary for each replication of the software run. Although only one random set is realized, it is part of a distribution of possible ranks if the random seed were allowed to vary. The variance in this distribution depends on the number of Monte Carlo realizations. The more Monte Carlo realizations that are run, the variance in the p-value will be smaller and the estimate will be closer to the true p-value. For decisions in epidemiology that involve possible implementation of contact tracing or other

expensive and invasive processes where the statistical significance is close to the decision threshold, an estimated p-value close to the truth is especially important. An estimated p-value from a small number of Monte Carlo realizations has a greater chance of under or over estimating the truth and leading to an incorrect decision. An estimated p-value from a large number of Monte Carlo realizations is closer to the true value and is more likely to lead to a correct decision.

Unfortunately, ensemble runs long enough to provide adequate confidence in decisions require computational resources that are usually beyond those available at typical health department analytical facilities. Cloud computing provides such resources without deploying extensive computational resources for very limited and sporadic use. Moreover, cloud resources could be implemented on top of existing infrastructures dedicated to routine office tasks in public health departments or similar organizations.

Similar work reported in the literature includes the Visual Statistical Data Analyzer (VISDA), a grid-based analytical tool [2,3] that includes spatial analyses, and work done using the Open-Source Grid-Computing technology to improve processing time for geospatial syndromic surveillance [4]. Both projects illustrated the value of grid computing in spatial analysis. Our work leverages the cloud which has the ability to be flexible in the amount of nodes involved and is not limited by hardware constraints in terms of amount of computer resources available. Moreover, the cloud provides resources at a much lower level of abstraction than grids and eliminates many of the cumbersome infrastructural and sharing agreements needed to deploy computational grids [5].

This paper reports our successful implementation of a SaTScan cloud system using the Nimbus TP2.X software [6]. To demonstrate its use we present the analysis of epidemic data from high-fidelity, agent-based simulation of pertussis epidemics. The model was built by the Virginia Bioinformatics Institute using their EpiSimdemics simulation platform [7] and consists of the space-time details of 2.2 million in silico individuals modeled after Utah population and physical geography [8]. This model maintains a disease profile for each individual that simulates both the presence and severity of symptoms, infectivity, and likelihood for seeking the help of a doctor. Individuals who were treated became less infectious or non-infectious once treated. The disease transmission model was based on the van Rie and Hethcote compartmental model for pertussis [9].

Methods

Design Decisions & Software Implementation

While the work presented here could be implemented using SOAP, the WSRF implementation is a better approach because it allows the integration of the cloud version of SaTScan into emerging public health grid infrastructures [10, 11]. At the time of this implementation the only WSRF (grid) solution for cloud computing accessible to the authors was the Nimbus cloud deployed at Argonne National Lab. Nimbus is an open source toolkit

that allows developers to turn a cluster into an Infrastructure-as-a-Service (IaaS) cloud (http://workspace.globus.org).

We accomplished our implementation using an iterative development approach with short iterations: iteration 1 involved installing and configuring SaTScan on a Linux based computer at the University of Utah and then wrapping SatSCan into a grid service; iteration 2 consisted of the deployment of this service on the Argonne Nimbus cloud; and, the final stage consisted of testing the performance and scalability of the cloud version of SatSCan.

A major design decision that we faced was where to implement the cloud client logic that would provide the on-demand functionality of the SaTScan cloud system. The choice was either to create a grid service that had the ability to stand-up SaTScan grid services and another grid service that could be invoked to run SaTScan jobs or to create a single SaTScan grid service that could perform both functions. Because the Nimbus server provides a general interface that allows users to stand-up various virtual machines, such as the SaTScan grid node virtual machine, there was no need to duplicate the Nimbus server-side capability to stand up a virtual machine (VM), greatly simplifying our deployment efforts.

SaTScan is a legacy application and in order to rapidly create a SaTScan grid service we used Introduce, gRAVI and the caGrid portal [12]. These tools and others developed by the caGrid project (http://cagrid.org/display/introduce/Home) provide a set of tools and a layer of abstraction around Globus WS-Core that significantly reduce the amount of effort required to deploy grid services. Introduce is an extensible toolkit to support easy development and deployment of WS/WSRF compliant grid services by hiding low level details of the Globus Toolkit and enabling the semi automatic implementation of strongly-typed grid services. Introduce has many useful plug-ins that are also available for further assistance. We used the Grid Rapid Application Virtualization Interface (gRAVI), a plug-in that allowed us to quickly wrap and deploy legacy application as Globus compliant grid services (http://dev.globus.org/wiki/Incubator/gRAVI).

Our development started by determining the parameter set required to execute SaTScan from the command line, then we used gRAVI and Introduce to wrap the SaTScan command line interface into a grid service. The caGrid portal provided an efficient and effective way to verify that the SatSCan grid service was deployed correctly. The caGrid portal leverages Google maps to depict grid services from the particular grid for which it has been configured. To test the deployment of the SaTScan grid service we used the caGrid training grid. As depicted in Figure 1, the SaTScan grid service appears correctly on the caGrid portal implying that the deployment has been successful. To verify that the SaTScan grid service was functional, we invoked the service using the SaTScan grid service client that was also created automatically by Introduce and gRAVI.

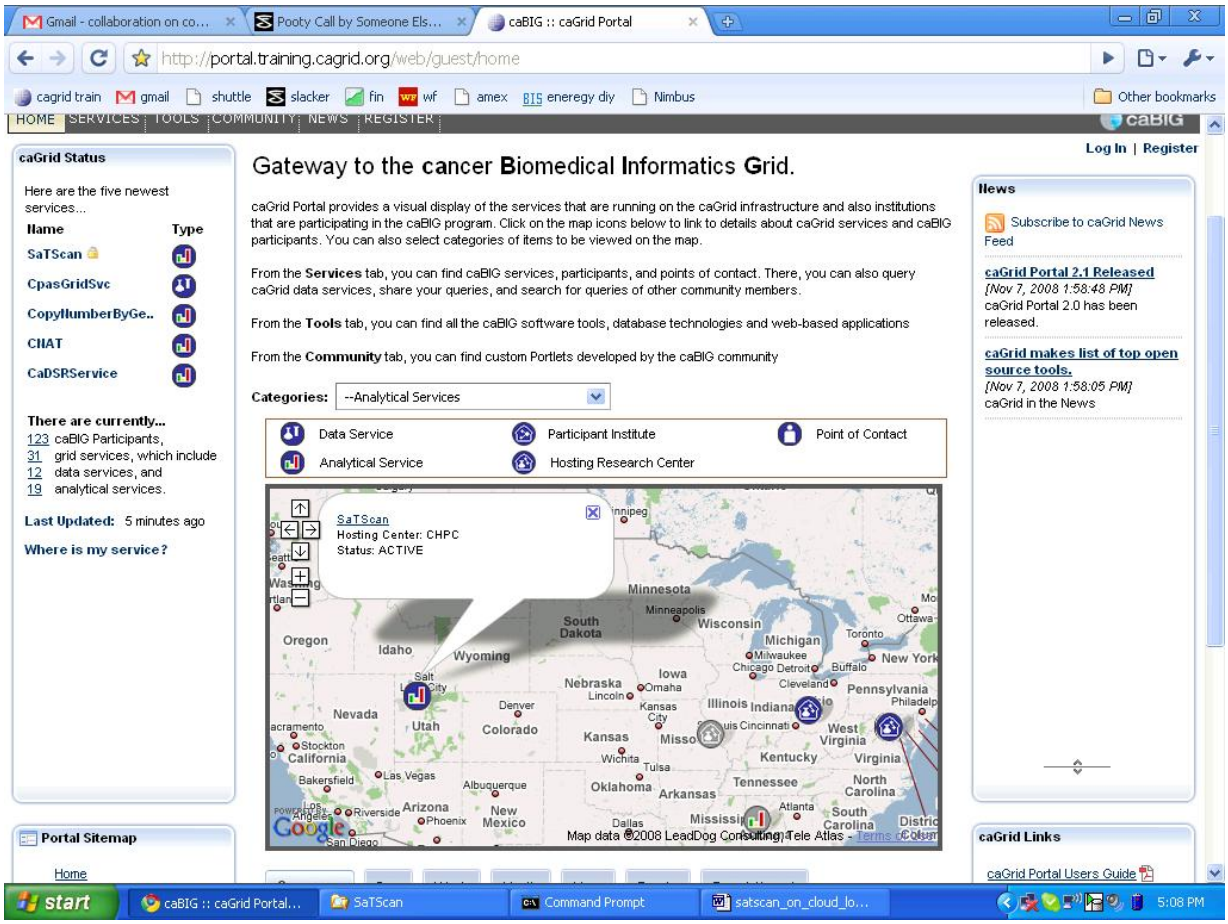

Figure 1: The caBIG portal for the test grid showing that the SaTScan grid service has been deployed successfully in Salt Lake City. UT

To demonstrate the dynamic scalability of our grid service with the goal to provide the on-demand SaTScan computer resources, we used the Argonne National Lab Nimbus cloud. To accomplish this we used the SaTScan grid service implemented on a Linux VM. The VM editing features available on the Nimbus client-side allowed us to use an existing Linux VM and edit it as needed. As part of the customization we added the caGrid software stack, the SaTScan grid service and configured a minimum of necessary services to initialize at boot time. At this point we were able to successfully stand-up a SaTScan grid node on-demand on the cloud and invoke it from a remote client.

In order to automatically manage the SaTScan clients we created a handler using the bash programming language. This handler, SaTScan Handler, manages all aspects of each SaTScan grid client including stage-in, job status progress and stage-out. The architecture of the handler is similar to the one used in our previous reported work on Digital Sherpa [13].

Scaling tests

For the scaling experiments we were able to stand-up up to 15 VMs in the Argonne Nimbus cloud. The first step in this process is to dynamically acquire the resources (VMs) needed for the desired run by invoking the Nimbus Workspace Service using the Nimbus Workspace Client. Fig. 2 depicts the different systems involved in this process.

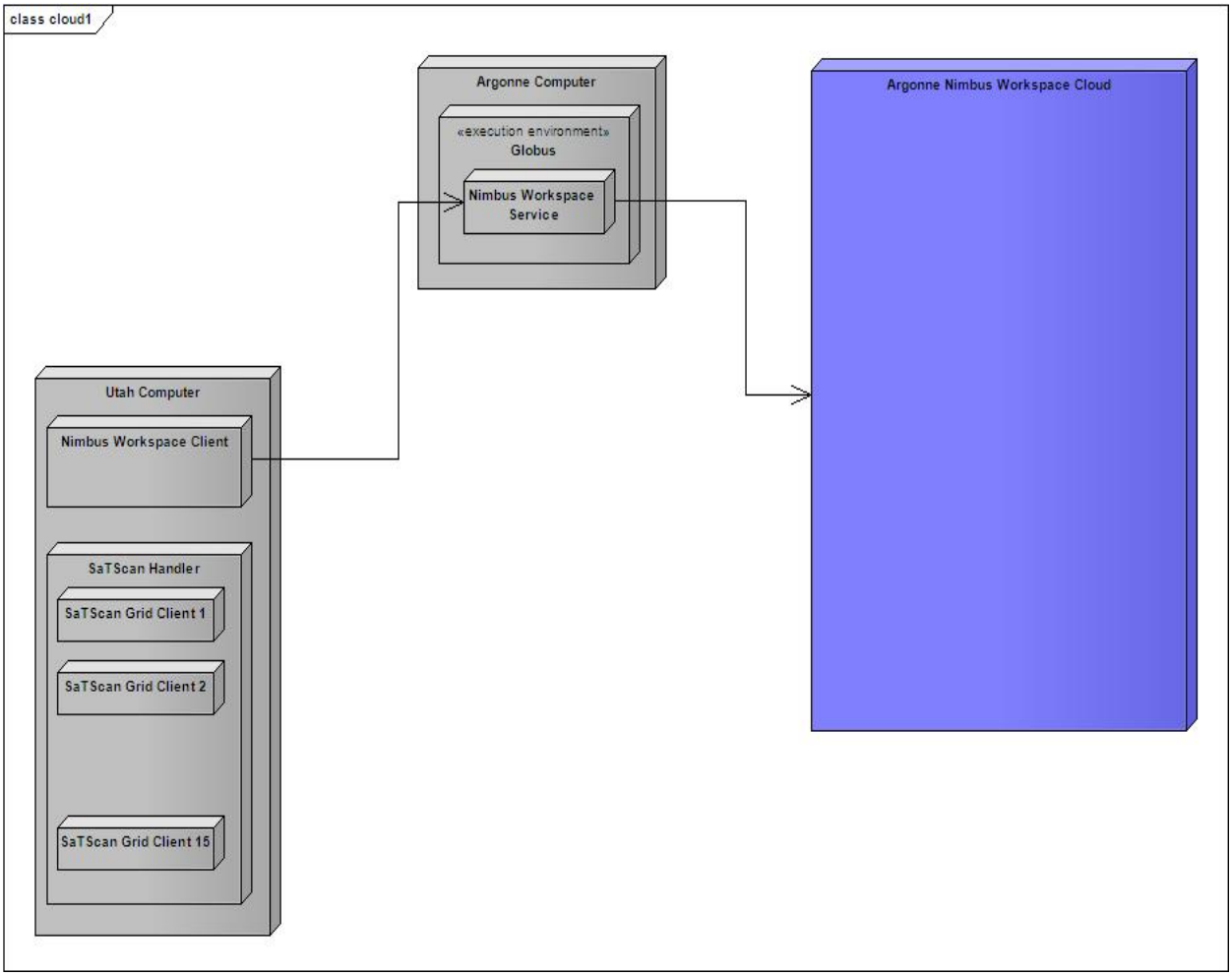

Figure 2: Initializing dynamic allocation of VMs using the Nimbus Workspace Service

Once the Nimbus Work Space service has been secured it is possible to start booting the Linux VM with the SaTScan grid services. The system obtained after the boot process completes is depicted in Fig. 3

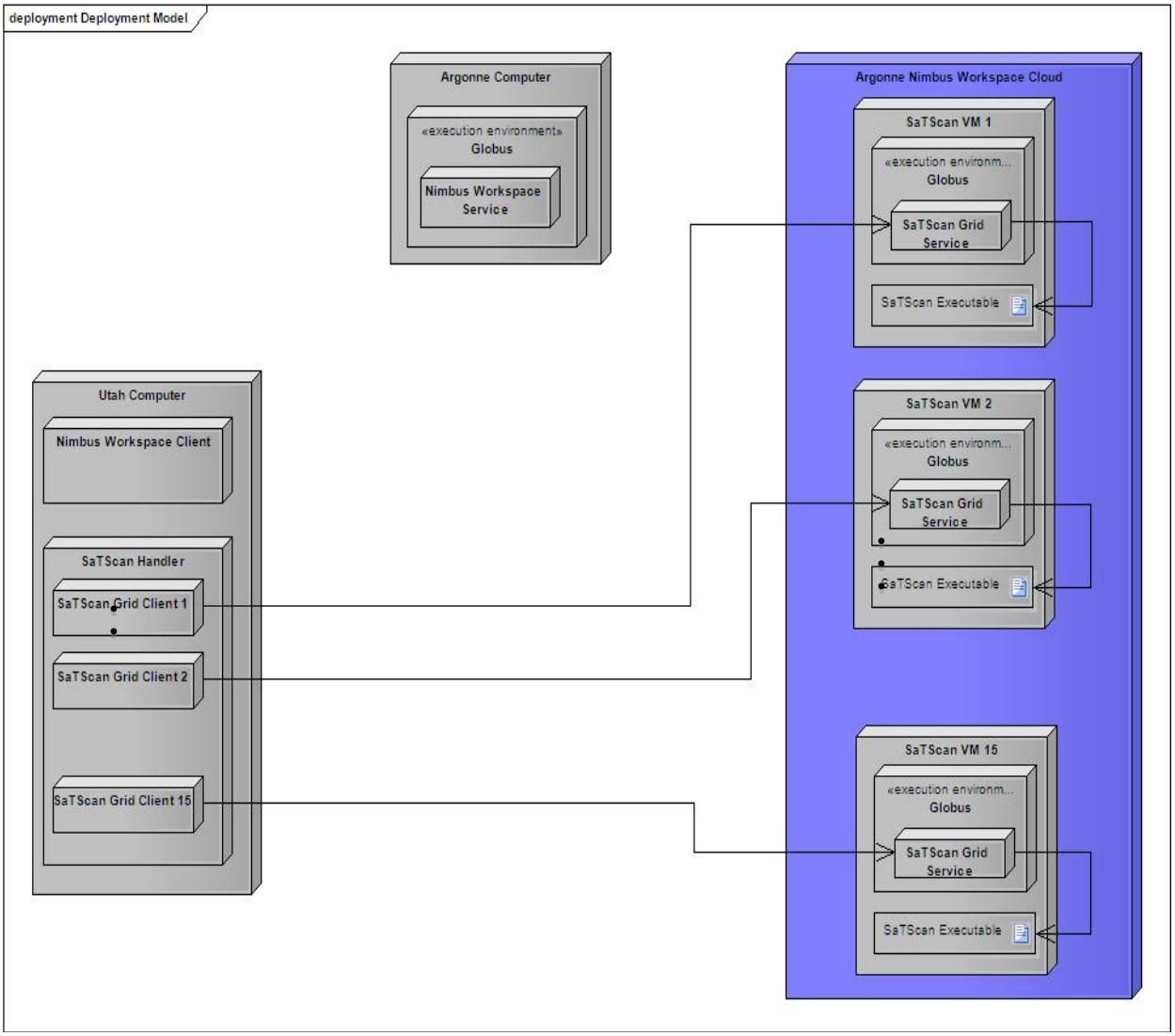

Figure 3: SaTScan VMs have been dynamically acquired and deployed in the cloud. They are ready to execute ensemble runs as they are sent by the SaTScan clients managed by the SaTScan Handler.

The SaTScan Handler manages the ensemble runs, which for these tests included up to 15 SaTScan Grid Clients. As depicted in Fig. 3, each SaTScan Grid Client in the SaTScan Handler submits a single SaTScan run to be processed by a SaTScan Grid Service, which delivers the task to the SaTScan executable. Upon completion the client moves the output files to the local host and the SaTScan Handler assembles the complete output of the ensemble run.

We prepared SaTScan's instruction files (.prm) to run a total of 9,990 Monte Carlo simulations using the same data files on 5, 8, 12, 13 and 15 VMs running the SaTScan Grid services, but using different seeds for the SaTScan's random number generator. The data files

included all 2.2 million individuals divided into the 292 Utah Zip Codes (population & coordinates files) for the full 210 simulated days (the full duration of the simulation) with resolution at the "day" level. A total of 4,521 cases were reported in the "case" file. For analysis type, we set SaTScan to run a retrospective space-time analysis using a Poisson distribution that assumes rare events. Aggregated and packaged data files for SaTScan, including case, coordinates and population, were approximately 1 MB and each of the SaTScan instruction files was approximately 8 KB. These are relatively small files and their transfer across the network does not increase the execution times significantly. Each VM in the cloud received approximately equal numbers of Monte Carlo simulations that are inversely proportional to the total number of VMs involved. For example, if we have ten VMs each one received 999 Monte Carlo replicates to compute. To establish a base line performance we also instructed one single node to run all 9990 Monte Carlo replicates using the same data files and analysis instructions. We verified that SaTScan ensemble runs performed in parallel on the cloud produce the same results as the sequential runs.

Results and Discussion

To evaluate the potential usefulness of the SaTScan cloud service for prospective users, we addressed the following user-oriented questions:

- When a user requests a cloud VM from the grid service, how long will it take before the VM is available for use?
- When using the cloud, what is the overhead incurred by the calculations?
- What is the overall speed up of the calculations and how does it reflect on the perceived turnaround?

The turnaround time of the simulations, which these questions address, is paramount for epidemiologists. Depending on the results of each simulation they must decide on either performing new simulations or taking preventive action through normal public health communication channels.

The first question was addressed under the assumption that there is no contention for the requested resources, i.e. we measured the time required to stand up a cloud node as the cumulative time of transferring the OS image that represents the VM and booting the guest OS on the Nimbus cloud. Further delays may be observed if the cloud available to the runs is oversubscribed. In order to make boot-up faster we created an image that initializes as few services as possible. Using this strategy we were able to reduce the time needed for one node to boot from 283 seconds to 207 seconds (all times plus or minus 10-15 seconds). We also tried compressing the image to improve transfer time but the overhead due to the time required to uncompress the image far outweighed the benefits. While this overhead is significant, it is only a small fraction of the total execution time of a typical ensemble run, for comparison our 9,990 Monte Carlo ensemble run required approximately 8,996 seconds in a single processor.

To address the second question we ran 999 and 9,990 Monte Carlo replicas. The execution times without the VM and the grid services overhead were 901 seconds and 8,996 seconds, respectively. When running the SaTScan grid service on a VM these times increased to 1,078 seconds and 10,700 seconds, respectively. This represents an increase of 10.87 % and 18.99 % of the execution time; the slow down for a larger job can be attributed to the deeper software stack and VM cpu overhead.

Table 1: Scalability results of the SaTScan grid services provided in the Nimbus cloud

(execution times in seconds).

VMs	Execution time	Speedup	Replicates per node
1	10700	1	9990
5	2144	4.99	1998
8	1289	8.30	1249
10	986	10.85	999
13	725	14.75	769
15	635	16.85	666

To address the third question we compared the run of a SatSCan job of 9,990 Monte Carlo replicates on a single VM using the SatSCan grid service with the execution of the same number of Monte Carlo replicas in different number of nodes using also the VMs and the SaTScan grid services. The results are entered in Table 1 and Fig. 4.

The excellent scaling (for a constant size problem) depicted in the table is due to the nature of ensemble runs with embarrassing parallel characteristics. In parallel computing, an embarrassingly parallel workload is one for which little or no effort is required to separate the problem into a number of parallel tasks. This is often the case when no dependency (or communication) exists between the parallel tasks.

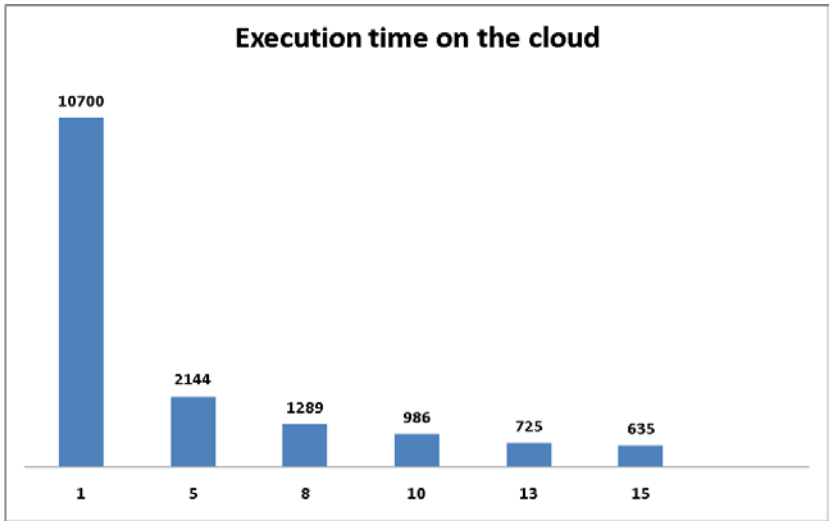

Figure 4: Total execution time of SatSCan on the Nimbus cloud as function of the number of VMs used. Execution times are in seconds.

Moreover, the overhead incurred by using a VM increases with the number of Monte Carlo replicates, and by running multiple copies of smaller number of replicates we are able to reduce this overhead, leading to the super-linear scaling depicted in Table 1. The scaling depicted in Table 1 is excellent, but for large number of VMs the start up cost may become a potential bottle neck. For the levels of parallelism explored here we believe that our results demonstrate that using a cloud approach provides on- demand computational resources for epidemiology surveillance. It is remarkable that when using 15 VMs the total execution time of 842 seconds, which includes 635 seconds of execution and the 207 seconds needed to stand up the VMs, is one order of magnitude smaller than the 8,896 seconds required to run the complete ensemble in one machine.

Conclusions

By using cloud computing and a computer intensive application like SaTScan, it is possible to provide on-demand resources for epidemic analysis. Therefore, implementing a cloud across the existing internal infrastructure of a health department may be a viable approach for large-scale epidemiology surveillance on demand. We have demonstrated that when using SaTScan we achieved an order of magnitude improvement in the turnaround, making possible a detailed analysis that may not be possible with the typical resources existing in public health departments. The techniques used for SaTSCan on the cloud could be generalized to any application that exhibits substantial parallel content. Using the caBIG tools and our software development methodology the time required to complete implementation took approximately 200 man-hours, an effort that could be secured with typical state health department resources. The approach proposed here is technically advantageous and can be practically implemented.

Acknowledgements

This work was partially funded by the Centers for Disease Control and Prevention through the Rocky Mountain Center of Excellence in Public Health Informatics # 1P01HK000069-10, National Library of Medicine Training grant # LM007124 and NCRR Clinical and Translational Science Award 1KL2RR025763-01.

Conflicts of Interest: The authors do not declare any conflict of interest.

References

[1] Kulldorff M, Nagarwalla N. Spatial disease clusters: detection and inference. Stat Med 14(8):799-810.
[2] Wang J, Li H, Zhu Y, Yousef M, Nebozhyn M, Showe M, et al. VISDA: An open-source caBIG analytical tool for data clustering and beyond. Bioinformatics 2007; 23(15):2024-7.
[3] Zhu Y, Li H, Miller DJ, Wang Z, Xuan J, Clarke R, et al. caBIG VISDA: modeling, visualization, and discovery for cluster analysis of genomic data. BMC Bioinformatics 2008; 9:383.
[4] Grannis S, Olson K, Egg J, Overhage JM. Using Open-Source Grid-Computing Technology to Improve Processing Time for Geospatial Syndromic Surveillance Data. Advances in Disease Surveillance 2006.
[5] Rings T, Caryer G, Gallop J, Grabowski J, Kovacikova T, Schulz S, et al. Grid and Cloud Computing: Opportunities for Integration with the Next Generation Network. Journal of Grid Computing 2009; 7(3):375-93.
[6] Keahey K, Freeman T, editors. Science Clouds: Early Experiences in Cloud Computing for Scientific Applications,. Cloud Computing and Its Applications (CCA-08); 2008; Chicago, IL.
[7] Barrett C, Bisset K, Eubank SG, Feng X, M M, editors. EpiSimdemics: An efficient and scalable framework for simulating the spread of infectious disease on large social networks. International Conference for High Performance Computing, Networking, Storage and Analysis (SC08); 2008; Austin, Texas.
[8] Pettey W, Benuzillo J, Walker B, Parks A, Kramer H, Gesteland P, Rubin M, Drews F, Livnat Y, Samore M. Using agent-based simulations of infectious disease spread to enhance public health decision support tools. American Public Health Association 173th Annual Meeting; 2009; Philadelphia.
[9]Van Rie A, HW. H. Adolescent and adult pertussis vaccination: computer simulations of five new strategies. Vaccine 2004; 22:3154-65.
[10] Staes C, Xu W, LeFevre S, Price R, Narus S, Gundlapalli A, et al. A case for using grid architecture for state public health informatics: the Utah perspective. BMC Medical Informatics and Decision Making 2009; 9(1):32.
[11] Staes CJ, Xu W, LeFevre SD, Narus SP, Gundlapalli A, Samore M, et al. A case for using grid architecture in state public health informatics: the Utah perspective. HelthGrid 2008; Chicago 2008.
[12] Oster S, Langella S, Hastings S, Ervin D, Madduri R, Phillips J, et al. caGrid 1.0: an enterprise Grid infrastructure for biomedical research. J Am Med Inform Assoc 2008 Mar-Apr;15(2):138-49.

[13] Price RC, Wayne BB, Victor EB, Julio CF. Digital Sherpa: a set of high level tools to manage scientific applications in a computational grid. Proceedings of the 15th ACM Mardi Gras conference: From lightweight mash-ups to lambda grids: Understanding the spectrum of distributed computing requirements, applications, tools, infrastructures, interoperability, and the incremental adoption of key capabilities; Baton Rouge, Louisiana: ACM; 2008.

doi: 10.5210/ojphi.v2i1.2910
Cite this item as: Price, R., Pettey, W., Freeman, T., Keahey, K., Leecaster, M., Samore, M., Tobias, J., & Facelli, J. 2010 Apr 9. SaTScan on a Cloud: On-Demand Large Scale Spatial Analysis of Epidemics. *Online Journal of Public Health Informatics* [Online] 2(1):e4.