REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
16-11-2016	Final Report	31-May-2013 - 30-May-2016

4. TITLE AND SUBTITLE Final Report: Modeling the Cloud to Enhance Capabilities for Crises and Catastrophe Management	5a. CONTRACT NUMBER W911NF-13-1-0117
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 206022
6. AUTHORS Eunice Santos	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES University of Texas at El Paso 500 West University Avenue Administration Building, Room 209 El Paso, TX 79968 -0587	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO 11. SPONSOR/MONITOR'S REPORT NUMBER(S) 62816-NS-REP.1

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

See attached.

15. SUBJECT TERMS

Cloud modeling, catastrophe management

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
UU	UU	UU	UU		Eunice Santos
					19b. TELEPHONE NUMBER 915-747-5170

Report Title

Final Report: Modeling the Cloud to Enhance Capabilities for Crises and Catastrophe Management

ABSTRACT

See attached.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received Paper

TOTAL:

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received Paper

TOTAL:

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

Received Paper

TOTAL:

Number of Manuscripts:

Books

Received Book

TOTAL:

Received Book Chapter

TOTAL:

Patents Submitted

Patents Awarded

Awards

Graduate Students

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period: 0.00

The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:...... 0.00

The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:...... 0.00

Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):...... 0.00

Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:...... 0.00

The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense 0.00

The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:...... 0.00

Names of Personnel receiving masters degrees

NAME

Total Number:

Names of personnel receiving PHDs

NAME

Total Number:

Names of other research staff

NAME	PERCENT_SUPPORTED

FTE Equivalent:
Total Number:

Sub Contractors (DD882)

Inventions (DD882)

Scientific Progress

See attached

Technology Transfer

Modeling the Cloud to Enhance Capabilities for Crises and Catastrophe Management

PI: Eunice E. Santos

Grant No.: W911NF1310117

Abstract

In order for cloud computing infrastructures to be successfully deployed in real world scenarios as tools for crisis and catastrophe management, where large amounts of dynamic information – even real time information have to be processed, novel algorithm designs, that can address the challenges of resource dynamism, scalability and virtualization in cloud environments, are needed. The overarching goal of this project is the design and development of a flexible mathematical modeling framework for the cloud infrastructure that can also leverage existing mathematical representations (e.g. graph theory), performance models (e.g. network models) and analysis tools (e.g. statistical analysis). In pursuit of this goal, we conducted an initial study to understand the impact of various cloud hardware and job parameters on performance. As part of this study, a cloud simulation environment on 32 compute nodes was used to run test programs under varying load conditions. The results and analysis of the initial performance study was used to explore adaptive algorithms designs for social network analysis for large and dynamic networks. We also identified a scenario, based on the challenging and computationally intensive problem of modeling resilience of social groups that we will use to validate our cloud modeling framework. It is worth noting that while the original performance period was 3 years, the project had a truncated performance period of less than 16 months.

1 Statement of the Problem Studied

As cloud computing becomes the dominant computational infrastructure[1] and cloud technologies make a transition to hosting applications that process large and dynamic information[2], there is a need for a novel high performance computing (HPC) algorithm design and performance analysis framework that will guide the deployment of mission critical real world scenarios such as crisis and catastrophe management. Over the three year performance period of the project, our overarching goal is to develop an algorithmic design and performance analysis framework that addresses the challenges of optimizing performance of HPC algorithms such as dynamically changing resources in cloud environments. We seek to design a flexible framework that allows for incorporating design features that will allow algorithms adapt to changing cloud environment by picking between different strategies such as in-memory and on-disc computation and between fine grained and coarse grained solutions. The flexible framework will also help in incorporating third party analyses tools and methodologies.

1.1 Project Research Objectives

The research objectives of the project are[1]:

1. Formulate rigorous mathematical models representing technological capabilities and resources in cloud computing for performance modeling and prediction especially under crises and catastrophe scenarios.
2. Develop algorithms to optimize performance when a cloud is under real-time emerging scenarios.
3. Initial investigation into methodologies to construct social networks based on cloud access and utility of users.
4. Validation of the models and algorithms.

1.2 Major Research Foci and Accomplishments

The project had a truncated performance period of less than 16 months. The major research accomplishments are summarized below. We also discuss the important research accomplishments in more detail in a later section.

1) Initial representations for infrastructure clouds: We focused on determining the critical performance parameters in a generic cloud environment. Parallel performance modeling techniques based on the LogP model [3], which is a communication model for distributed memory architectures, were used to study changes in network parameters such as latency with varying loads and computational resources. These can be used in graph representations that also take into account other performance parameters, such as overhead due to virtualization and load, for performance prediction.

2) Initial validation tests for basic cloud model: We used a cloud simulation on 32 compute nodes to study the performance of test problems under varying load conditions. Matrix multiplication in the ScaLAPACK libraries[4-5] was used as a test problem. The impact of communication and virtualization factors on the run time was studied. The results and analyses from this initial validation test are being used in the design and performance prediction of social network analysis algorithm with specific focus on ego-betweenness centrality algorithm. An initial design of the ego-betweenness algorithm has been implemented.

3) Sub-models to represent crises/catastrophe scenarios: In keeping with our objective to develop sub-models for complex, dynamically changing real world catastrophe scenarios[6], we explored the possibility of cross-leveraging our work in modeling resilience of communities, which was conducted as part of a DOD project.

Background

As part of this project we seek to leverage a*nytime-anywhere algorithms*[7-8] to address the challenges of processing extremely large and dynamic data on cloud infrastructures. We will now discuss the properties of anytime and anywhere algorithms and provide a description of the anytime anywhere algorithm framework that we seek to leverage for this project. The concept of *anytime algorithms* is not new. In fact there has been substantial research work in various sub-domains in computer science especially agent planning[9] and heuristic search[10] . Anytime algorithms can be interrupted at any point during its execution, and still provide a viable or approximate solution. The viability of the solution is

[1] As listed in the project proposal

measurable, and is usually termed as quality. One of the important characteristics of the anytime algorithms is that the quality of the solutions produced is monotonically non-decreasing with respect to the amount of processing. Therefore, the more time the algorithm is allowed to progress, the better its solution. Also the quality of the anytime algorithm are useful tools when processing large problem instantiations within time and resource bounds as it can provide at least a partial solution, where ordinary algorithms would fail to provide any solution at all. The partial solutions produced due to the anytime property is critical for enabling the anywhere property of our algorithm design methodology. Anywhere algorithms[8, 11] have the ability to incorporate complete or partial solutions produced by some other algorithm/methodology/processor in its local solutions. Anywhere property also refers to the ability to incorporate dynamic changes in the algorithm input with minimal overhead. It is here that the partial solutions generated by the anytime aspect become useful as the partial solutions can be reused to reduce computational overheads and recalculations triggered due to dynamic data.

1.3 Anytime Anywhere Framework

The anytime anywhere algorithm framework [1, 2, 6] is a generic framework that was used to design social network analysis (SNA) algorithms in large scale parallel and distributed environments, and has been validated for All-Pairs-Shortest-Paths (APSP), centrality and maximum clique problems with large and dynamic social networks in a cluster computing environment. The original anytime anywhere framework was developed through support by a prior DOD grant, and in this project, we focused on how to effectively adapt this for the cloud catastrophe environment. The algorithm designed using this framework go through the following phases.

1. **Domain Decomposition (DD)**: The technique used to partition the graph should lead to a balanced partition and minimal inter-processor communication. Balancing the load is a challenge as the computations required for the network vertices vary, and depends on the graph algorithm and the vertex connectivity. Therefore partitioning algorithm should seek to minimize the number of edges that need to be cut when assigning network vertices to compute nodes. Selecting the optimal partitions in many cases is NP-Complete. In order to reduce the overhead of graph partitioning, domain decomposition may use heuristics to generate the sub-graphs. Such heuristics may use the idea of cut edges to minimize the number of edges that need to be disconnected to form the sub-graphs, and therefore reducing potential communications between compute nodes.

2. **Initial Approximation (IA)**: This phase deals with calculating partial results using information local to each compute node. Therefore, the results are quick and coarse grained approximations of the final results. The quality or accuracy of the approximations with respect to the final results are also dependent on the partitions generated during domain decomposition. The initial approximation phase generally complete quickly with respect to the other two phases. The local graphs and results are then shared with other processors in the recombination phase to refine the results and converge to the final results.

3. **Recombination Phase (RC):** The third and final phase has two objectives. The first objective is to share the partial results generated during the IA phase and refine the local results. Through an iterative process of communication and assimilation of partial results, the local results are refined to the final value. The second objective is to incorporate the dynamic changes in the inputs. The changes are first incorporated in the local computation of the respective compute nodes, and then propagated to other compute nodes.

2 Research Results Summary

As discussed previously, the project, with the original performance period of 3 years, had a truncated performance period of less than 16 months due, in part, to the departure of the research team to a different organization.

For this project, we considered two major research foci: 1) initial study of test problems under varying load conditions to understand its impact on performance parameters, and 2) initial design of SNA algorithms for extremely large (millions of nodes and billions of edges) and dynamically changing graphs with specific focus on *k* order ego-Betweenness centrality measures. During the second year, the focus was on identifying a relevant scenario in the domain of catastrophe planning with an application on the cloud computing platform. The objective of the initial performance studies was to determine the critical cloud parameters that are most relevant for formulating performance models. We also conducted experiments using traditional high performance computing (HPC) benchmarks such as ScaLAPACK[5], and study the variation in these metrics with different problem sizes and varying resources (processors and network characteristics). The goal was to gain an insight into specific problems of cloud infrastructures such as VM overhead and contention for resources within a compute node. We also wanted to lay the ground work for studying the impact of extreme conditions such as crises and catastrophes on cloud performance. We were also interested in understanding how partial failures in cloud infrastructure can impact performance of application, and in developing techniques and design paradigms to mitigate the impact of such events. However, we note that the results for the first and second studies, described below, are preliminary, and will be refined, as appropriate, as part of future work.

We conducted the initial performance study using a cloud simulation environment called CloudStack. The experimental setup consisted of 32 dual processor Xeon processors (with 8 cores) and 10 GBPS Ethernet network. The Virtual Machines (VMs) in the CloudStack[2] are installed on top of a hypervisor program. The experiments were conducted using the matric multiplication kernel called DGEMM from the ScaLAPACK library as the test problem. Instrumentation code was added to the kernel to collect test data such as the computation and communication times. The experiments are conducted using the Message Passing Interface (MPI). The conditions in the cloud environment was varied by changing the background traffic, and performance results were gathered for different conditions. In the first study, we focused on studying the variations of the L, o, g performance parameters when moving from VMs that run on the host OS to VMs that run on the hypervisor middle ware (guest OS). These parameters are part of a well-known parallel distributed memory modeling framework called LogP[3]. LogP performs best in lightly loaded networks that are commonly found in low contention environments such as cluster computing environments. The cloud, on the other hand, seeks to optimize availability of computing resources and this comes at the price of varying load on processors, and contention of network resources. By studying the effects of varying cloud resources and the cloud architecture on L, o, and g values, we will be able to formulate realistic analytical models of cloud performance. We compared the values of L, o, g for VMs running on host machines with those for VMs on guest machines. These values are calculated by measuring the time taken to send a unit message between two VMs. This time is a combination of the latency and the overhead. Latency is caused by network based factors such as routing delay, and the overhead is caused by message preparation operations (creation of a message buffer and copying of the

[2] https://cloudstack.apache.org/

message). The results (Figure 1, Table 1) show that the gap parameter *g* does not change for the VMs running on the guest machines or host machines. This is because *g* depends solely on the network bandwidth, and this remains the same for both the types of VMs. On comparing the value of the overhead parameter o, we see that guest VM has a much larger value for *o* than the host machine. This due to the fact that message buffers are created by system calls that have to go through the extra layer of the hypervisor in the guest VMs leading to delays. However the largest differences are seen for the latency parameter *L*. This is largely due to the contention between the guest VM and the host OS for system resources. This is borne out when the experiments are repeated with 11 co-located guest VMs (Figure 2, Table 2). With large number of VMs sharing system resources, there is increased contention, which in turn leads to higher values of L. However when we compare the performance parameters between 1, 8 and 11 co-located VMs (Figure 3 L, o, g values with varying co-located guest VMs[4]Table 3), we see that 1 and 8 co-located VMs have similar L, o, g values. This is because in both the cases, each processor core is assigned to at most one virtual machine and there is not overhead due to context switching. On the other hand, 11 co-located VMs have a drastically high values for L caused by high processor contention. Our conclusion is that contention for processor cycles by co-located VMs has a large impact on L, and that the values for o and g do not vary much with increasing number of co-located VMs.

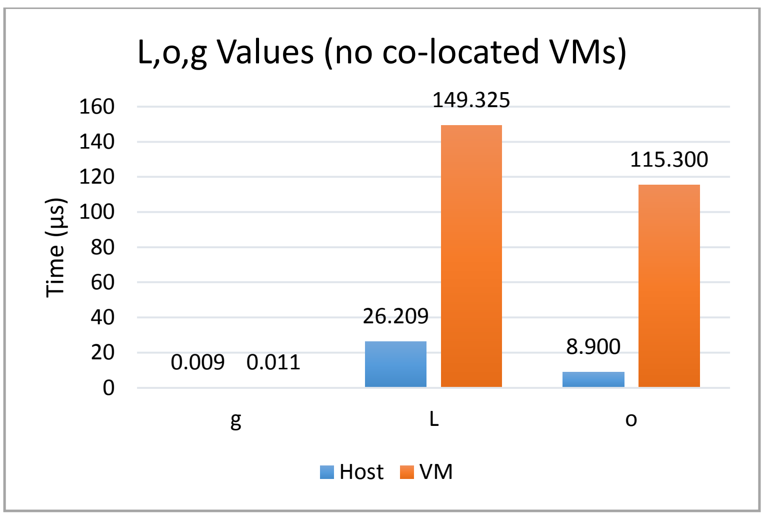

Figure 1 L, o, g values with no co-located VMs[3]

	g(µs)	L(µs)	o(µs)
Host	0.009	26.209	8.900
VM	0.011	149.325	115.300

Table 1 L, o, g values with no co-located VMs[3]

[3] Note that these are preliminary results

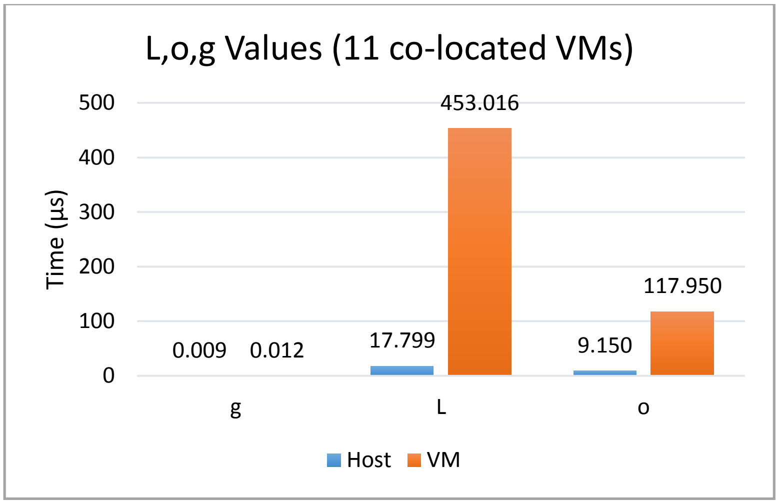

Figure 2 L, o, g values with 11 co-located VMs[4]

	g(µs)	L(µs)	o(µs)
Host	0.009	17.799	9.150
VM	0.012	453.016	117.950

Table 2 L, o, g values with 11 co-located VM0[4]

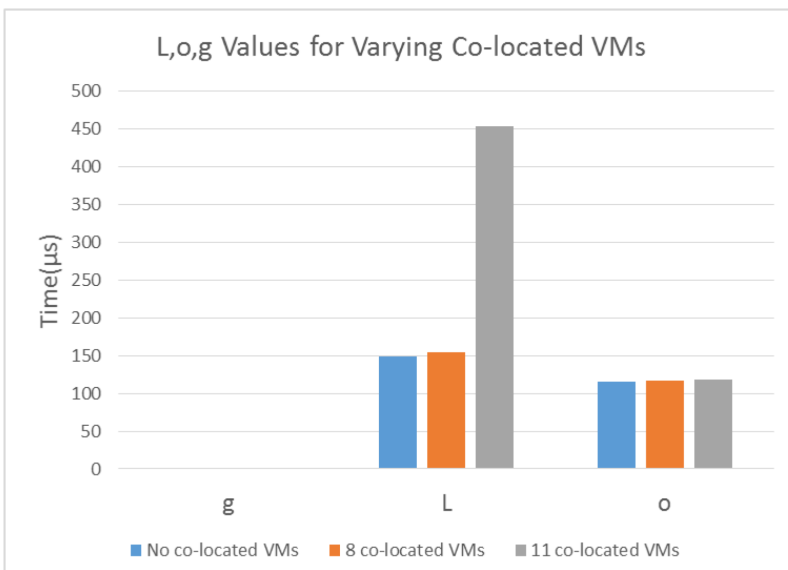

Figure 3 L, o, g values with varying co-located guest VMs[4]

[4] Note that these are preliminary results

	No co-located VMs	8 co-located VMs	11 co-located VMs
g(µs)	0.0107	0.015	0.119
L(µs)	149.32	153.99	453.016
o(µs)	115.3	116.2	117.95

Table 3 L, o, g values for varying sizes of co-located guest VMs[5]

The second study focused on the variation in computing and communication costs of test problems due to varying loads in the cloud environment. We used the parallel matrix multiplication kernel (ScaLAPACK DGEMM kernel) as the test problem. We also developed a traffic simulation program that simulated different traffic loads in the cloud. The simulator program used a combination of matrix multiplication and all-to-all broadcast (MPI libraries) to generate the traffic conditions. We calculated the computation and communication times under light, medium, heavy and zero traffic. We also varied the number of processor used in the simulations. Figure 4(Table 4) and Figure 5 (Table 5) show the difference in computation and communication time taken for running a parallel matrix multiplication kernel (ScaLAPACK DGEMM kernel) under varying workloads. As expected, we see the general trend from the results (Figure 4, Table 4) is that the computation time does not vary widely under different traffic conditions. The results for the communications times (Figure 5, Table 5) also demonstrate that communication time increases considerably in heavy traffic conditions. Future work will look into methods to analytically represent these trends in communication and computation times due to resource contention in our cloud performance model.

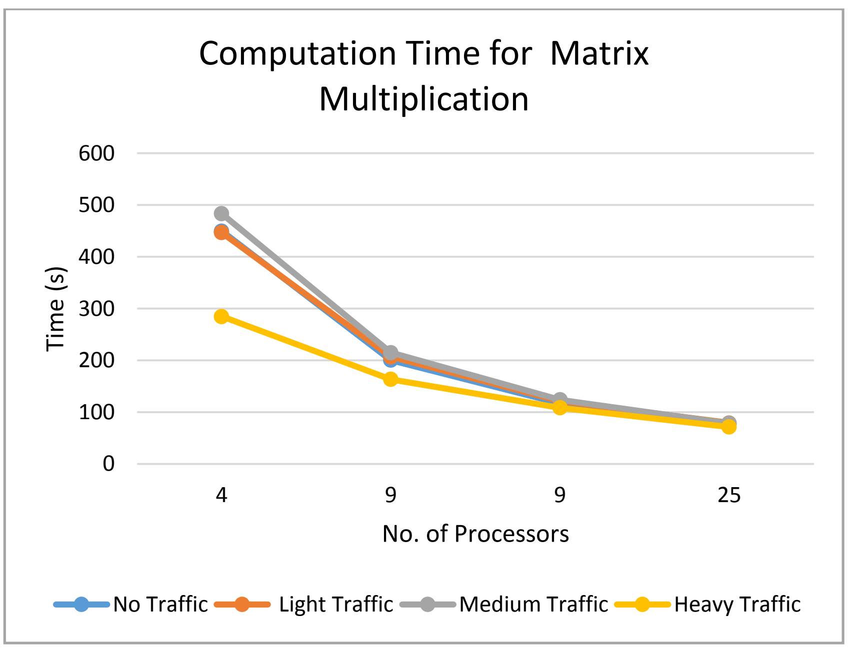

Figure 4 Computation time for parallel matrix multiplication on square matrix (n=10000)[5]

[5] Note that these are preliminary results

No. of processors	No Traffic (S)	Light Traffic (S)	Medium Traffic (S)	Heavy Traffic (S)
4	449.7	446.92	483.3	284.74
9	200.33	206.91	214.96	163.44
9	116.59	121.19	123.69	108.44
25	75.46	79.11	78.06	71.3

Table 4 Computation time for parallel matrix multiplication on square matrix (n=10000)[6]

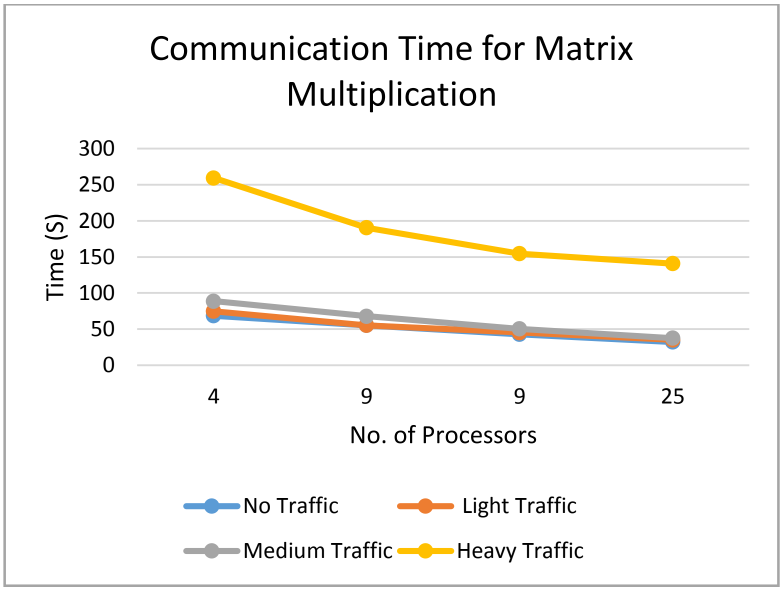

Figure 5 Communication time for parallel matrix multiplication on square matrix(n=10000) under varying workload[6]

No. of processors	No Traffic (S)	Light Traffic (S)	Medium Traffic (S)	Heavy Traffic (S)
4	68.66	74.85	88.72	259.62
9	55.37	55.26	67.97	190.63
9	42.96	45.2	50.43	154.55
25	32.12	35.14	37.63	140.9

Table 5 Communication time for parallel matrix multiplication on square matrix of (n=10000) under varying workload[6]

Social networks [13-15] use the graph theoretic notions of nodes and edges to represent complex social structures and relationships. Tools for analyzing social networks called social network analysis (SNA) have been formulated and have been successful in gaining new insights to challenging questions in social interactions. SNA techniques are able map abstract social concepts to quantitative measures that can be used to generate explanations, and even predictions. Centrality forms a widely used class of SNA measures called centrality. Centrality measure aims to quantify the importance of a node to its social network. One of the major centrality measure is betweenness centrality.

[6] Note that these are preliminary results

Betweenness Centrality: of a vertex V_i is calculated by finding the fraction of the shortest paths between all pairs of vertices in the graph that the vertex is a part of. Due to the time and memory complexity of finding the all pairs shortest paths, an approximation in the form of ego-Betweenness centrality can be used. An ego network of degree K for a vertex (called the ego vertex) is the sub-graph induced by the vertices (called alters) that are at a distance k or less from the ego vertex. Given a graph $\mathrm{G}(\mathrm{V},\mathrm{E},\mathrm{w})$ where V is the set of nodes, E is the set of all edges between the nodes in V and w is the set of weights for the edges. As mentioned before, the ego network of order k of a vertex V_i is a sub-graph $\mathrm{G}_\mathrm{i}^\mathrm{k}(\mathrm{V}_\mathrm{i}^\mathrm{k},\mathrm{E}_\mathrm{i}^\mathrm{k},\mathrm{w}_\mathrm{i}^\mathrm{k})$ induced from G using the vertex set $\mathrm{V}_\mathrm{i}^\mathrm{k}$, which is the set of nodes in G that are at a distance k or less from V_i. Using the definition of Ego-Betweenness centrality of a node V_i as the fraction of the all pairs shortest paths that pass through it, the Ego-Betweenness centrality $C_{ego}(v_i)$ can be formulated as:

$$C_{ego}(v_i) = \sum_{\alpha=1}^{|V_i^k|} \sum_{\substack{\beta=1 \\ \beta\neq i\neq\alpha}}^{|V_i^k|} \frac{S_{\alpha,\beta}(v_i)}{S_{\alpha,\beta}}$$

where,

$S_{\alpha,\beta}$: no. of shortest paths in G_i^k between nodes $v_\alpha,\ v_\beta \in V_i^k$

$S_{\alpha,\beta}(v_i)$: no. of shortest paths in G_i^k between nodes $v_\alpha,\ v_\beta$ that pass through v_i.

Our work in the first year of the project was to lay the foundations for a generic algorithm design framework for cloud computing. Initial designs of the betweenness algorithm that can adapt to changing resources and time constraints in the cloud were formulated. The next step was to identify a modeling problem where the anytime anywhere algorithm can be used in a cloud computing environment.

We explored the challenging problem of modeling the social resilience of communities during catastrophes as a potential validation scenario for our performance model. We cross-leveraged work on modeling social resilience that was conducted as part of another DoD project lead by the PI. Modeling social resilience is a challenging research problem. Myriad socio-cultural factors can have an impact on social resilience, and the emergent social structures and processes are very often hard to predict. Incorporating relevant factors in the models, and representing their influences on resilience are also modeling challenges. We explored cross leveraging the resilience models of the Somali fisherman community, and its impact on piracy to validate our performance models for the cloud environment during catastrophes.

3 Concluding Remarks

The overall goals of this project is to formulate a mathematically modeling framework for the performance of cloud infrastructures during catastrophes. In the truncated performance period of the project, we focused on three research objectives: 1) Research initial representations for infrastructure clouds, 2) Construction of simple validation test for basic cloud model, and 3) Develop sub-models to represent crises/catastrophe scenarios. As the first step towards formulating the performance model, we conducted experimental studies to study the impact of processor and network contention on performance. We selected matric multiplication kernel from ScaLAPACK as the test problem for these experimental results. We also leveraged the LogP framework to study the variation of latency (L), overhead (o) and gap (g) parameters with different number of virtual machines in the cloud environment. We also explored the possibility of using resilience of communities as a validation domain for our cloud modeling framework. The initial results provided in this report has laid a strong foundation, and we will leverage these results in future work for formulating a performance model for cloud environments. Future publications that are built on the results presented in this report will contain appropriate attribution of the support provided by this project.

4 Bibliography

[1] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Futur. Gener. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, Jan. 2013.

[2] V. Mauch, M. Kunze, and M. Hillenbrand, "High performance cloud computing," *Futur. Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1408–1416, 2013.

[3] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: Towards a Realistic Model of Parallel Computation," *SIGPLAN Not.*, vol. 28, no. 7, pp. 1–12, Jul. 1993.

[4] J. W. Demmel, J. Dongarra, B. Parlett, W. Kahan, M. Gu, D. Bindel, Y. Hida, X. Li, O. Marques, E. J. Riedy, and others, "Prospectus for the next LAPACK and ScaLAPACK libraries," in *Applied Parallel Computing. State of the Art in Scientific Computing*, Springer, 2007, pp. 11–23.

[5] J. Choi, J. J. Dongarra, R. Pozo, and D. W. Walker, "ScaLAPACK: A scalable linear algebra library for distributed memory concurrent computers," in *Frontiers of Massively Parallel Computation, 1992., Fourth Symposium on the*, 1992, pp. 120–127.

[6] H. Gao, G. Barbier, and R. Goolsby, "Harnessing the crowdsourcing power of social media for disaster relief," *IEEE Intell. Syst.*, no. 3, pp. 10–14, 2011.

[7] E. E. Santos, L. Pan, D. Arendt, and M. Pittkin, "An Effective Anytime Anywhere Parallel Approach for Centrality Measurements in Social Network Analysis.," in *IEEE International Conference on Systems, Man and Cybernetics*, 2006, pp. 4693 – 4698.

[8] E. E. Santos, L. Pan, D. Arendt, H. Xia, and M. Pittkin, "An Anytime Anywhere Approach for Computing All Pairs Shortest Paths for Social Network Analysis," in *Integrated Design and Process Technology*, 2006.

[9] T. L. Dean and M. S. Boddy, "An Analysis of Time-Dependent Planning.," in *AAAI*, 1988, vol. 88, pp. 49–54.

[10] S. Koenig, M. Likhachev, Y. Liu, and D. Furcy, "Incremental heuristic search in AI," *AI Mag.*, vol. 25,

no. 2, p. 99, 2004.

[11] E. Santos Jr, S. E. Shimony, E. M. Williams, and others, “Solving Hard Computational Problems through Collections (Portfolios) of Cooperative Heterogeneous Algorithms.,” in *FLAIRS Conference*, 1999, pp. 356–360.

[12] L. Pan and E. E. Santos, “An anytime-anywhere approach for maximal clique enumeration in social network analysis,” in *IEEE International Conference on Systems, Man and Cybernetics*, 2008, pp. 3529–3535.

[13] E. Santos, E. Santos Jr, J. Korah, R. M. George, Q. Gu, J. Jurmain, K. Kim, D. Li, J. Russell, S. Subramanian, J. Thompson, and F. Yu, “Incorporating Social Theories in Computational Behavioral Models,” in *Proceedings of the Seventh International Conferences on Social Computing,Behavioral-Cultural Modeling and Prediction*, 2014, pp. 341–349.

[14] E. E. Santos, E. Santos Jr., J. T. Wilkinson, J. Korah, K. Kim, D. Li, and F. Yu, “Modeling Complex Social Scenarios Using Culturally Infused Social Networks,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2011, pp. 3009–3016.

[15] E. E. Santos, E. Santos, L. Pan, J. T. Wilkinson, J. E. Thompson, and J. Korah, “Infusing social networks with culture,” *Syst. Man, Cybern. Syst. IEEE Trans.*, vol. 44, no. 1, pp. 1–17, 2014.