Cloud Computing for Fundamental Spatial Operations on Polygonal GIS Data

Dinesh Agarwal, Satish Puri, Xi He, and Sushil K. Prasad[1]
Department of Computer Science
Georgia State University
Atlanta - 30303, USA
Email: {dagarwal2, spuri2, xhe8}@student.gsu.edu, sprasad@gsu.edu

Abstract*—Efficient end-to-end parallel/distributed processing of polygon-based spatial data (also known as vector-based data) has been a long-standing research question in GIS community. The irregular and data intensive nature of the underlying computation has impeded the exploratory research in this space. We have created an open architecture based system named *Crayons* for Azure cloud platform using state-of-the-art techniques. The design and development of *Crayons* system is an engineering feat both due to (i) the emerging nature of the Azure cloud platform which lacks traditional support for parallel processing and (ii) the tedious exploration of design space for right techniques for parallelizing various workflow components including file I/O, partitioning, task creation, and load balancing. *Crayons* is an open-source system available for both download and online access, to foster academic activities. We believe *Crayons* to be the first distributed GIS system over cloud capable of end-to-end spatial overlay analysis. We demonstrate how Azure platform's storage, communication, and computation mechanisms can support high performance application (HPC) development. *Crayons* scales well for sufficiently large data sets, achieving end-to-end relative speedup of over 40-fold employing 100 Azure processors. For smaller, more irregular workload, it still yields over 10-fold speedup.

I. Introduction

Cloud computing is emerging as a promising platform for compute and data intensive scientific applications. Researchers in Geographic information systems and science (GIS) have always perceived large scale vector-data computation as a challenge due to the underlying data intensive and irregular computational nature - in contrast to raster data which is regular and highly parallelizable. When large volumes of vector data are deployed for spatial analysis and overlay computation, it is a time consuming task, which in many cases is also time sensitive, such as for hurricane path prediction [1].

Source	Example Type	File Size
US Census [3]	Block Centroids	705 MB
	Block Polygons	108 MB
	Blockgroup Polygons	14 MB
GADoT [4]	Roads	130 MB
USGS [5]	National Hydrography Data set	13.1 GB
	National Landcover Data set	3-28 GB
JPL [6]	Landsat TM	4 TB
Open Topography [7]	LIDAR	0.1-1 TB

TABLE I: Example GIS data sets and typical sizes

Table I shows some example data sets with typical file sizes. Depending on the resolution and geographic extents, data sets can get extremely large [2].

Comprehensive analysis of such data sometimes is not at all possible by employing a standard desktop system (the state-of-art), and even if it is, it usually takes hours to days before the application can obtain any analytical results. Even for non-emergency response applications, spatial data processing routines run for extended periods of time.

Why Cloud? For a wide range of large scale distributed computing applications from Geosciences, the demand for resources varies significantly during the course of execution. While a set of dedicated resources for such applications could result in under-utilization most often, at other times the system could perform better by utilizing more resources than available. Therefore, the emerging cloud platforms, such as Microsoft's Azure have the promise to be the platform of choice for such GIS applica-

[1]This work is partially supported by NSF/Microsoft Eager grant CCF 1048200.

tions.

Cloud computing promises scientists with a new infrastructure and paradigm for large scale distributed computing [8]. There have been some initial work to understand the pros and cons of this new framework [8]–[10]. However, only a few Geoscience-related projects have been initiated on the cloud platform. Most relevant among these include ModisAzure project for download, reprojection, and reduction of satellite imagery [8], [10], [11], and Smart Sensors and Data Fusion applications project for ocean observation [12].

Contributions: We have engineered *Crayons*[2] system over Azure cloud with a parallel, open software architecture for polygon overlay analysis. We believe *Crayons* to be the first cloud-based system for end-to-end spatial overlay processing on vector data. Our specific technical contributions are as follows:

- Engineering an end-to-end spatial overlay system by way of designing and implementing three versions: (i) Centralized Dynamic Load Balancing, (ii) Distributed Static Load Balancing, and (iii) Distributed Dynamic Load Balancing.
- Open architecture of *Crayons* for interoperability with any third party domain code (GPC library) for sequential execution of primitive overlay computation over two polygons.
- End-to-end relative speedup of more than 40x, using input GML files with comparatively uniform load distribution, and more than 10x using input GML files with skewed load distribution, using 100 Azure processors.

The rest of this paper is organized as follows: Section II describes our parallel Azure framework and its three flavors. Our experimental results are presented in Section III. Section IV concludes this paper with comments on future work.

II. Architecture of Crayons system

We have spent considerable effort analyzing the still-emerging Azure platform's nitty-gritty to gain insights into Azure framework, parallel reading and writing from and to cloud storage[3], and load balancing. In the process, we created three different architectures for *Crayons* to thoroughly test the Azure platform's design artifacts and to carve out path for future development. Due to space constraints, we will only discuss the *Crayons*' architecture with distributed dynamic load balancing, our best performing version, in detail. However, we will briefly discuss the differences with respect to other two versions.

A. Crayons Architecture with Distributed Dynamic Load Balancing

Figure 1 shows the architectural diagram of *Crayons* with distributed dynamic load balancing. The entire workflow for this architecture is divided into four steps as outlined below:

Step I. The web role presents the interface with a list of GML files available to be processed along with the supported operations (currently union, intersection, x-or, and difference). The user selects the GML files to be processed along with the spatial operation to be performed on these files. The web role puts this information as a message on the input queue.

Algorithm 1 Algorithm to create polygon intersection graph (approach similar to [15])

INPUT: Set of Base Layer polygons S_b and Set of Overlay Layer polygons S_o

OUTPUT: Intersection Graph (V,E), where V is set of polygons and E is edges among polygons with intersecting bounding boxes.

```
Quicksort set $S_o$ based on X coordinates of bounding boxes
for all base polygon $B_i$ in $S_b$ do
    Find $S_x \subset S_o$ such that $B_i$ intersects with all polygons in set $S_x$ over $X$ coordinate (binary search over $S_o$)
    Quicksort $S_x$ on y coordinates of bounding boxes
    for each polygon $O_j$ in $S_x$ that $B_i$ intersects with in Y coordinate do
        Create an edge ($B_i$, $O_j$ ) in graph G
    end for
end for
```

Step II. Worker roles continuously check the input queue for new tasks. If there is a task (message)

[2]For Azure code, extended manuscripts, and rigorous experimental data, see [13]

[3]AzureBench suite has detailed benchmarking of the blob, queue, and table storage mechanisms [14].

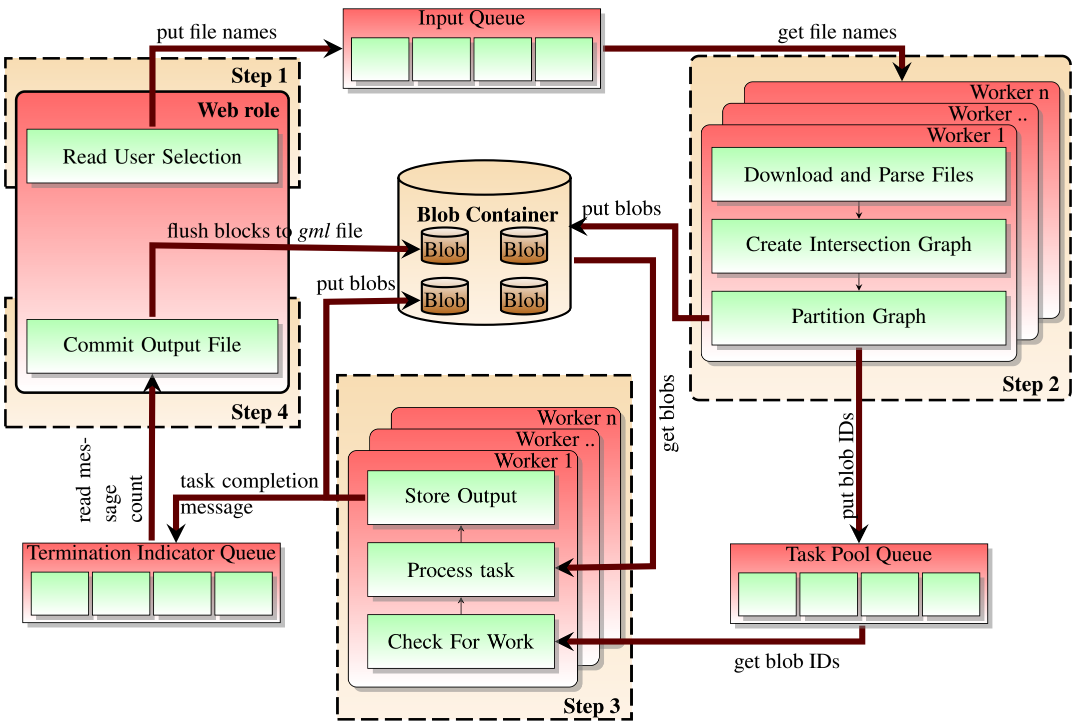


Fig. 1: *Crayons* architecture with distributed dynamic load balancing

in the queue, the worker roles read the message, download the input files, parse them, and create the intersection graph to find the independent tasks. To achieve this, *Crayons* finds each overlay polygon that can potentially intersect with the given base polygon and only performs spatial operation on these tasks. As shown in Algorithm 1, this is achieved using the coordinates of bounding boxes generated during parsing of input files. Then each worker role shares the tasks it creates among all the worker roles by storing the task IDs in a common task pool (Task Pool Queue in Figure 1).

Step III. After the workers finish task creation, they fetch work from the task pool and process them by invoking *GPC* library [16]. The advantage of this approach is that the worker role instances can also process the work of other worker role instances and hence achieve decent performance even with skewed load.

After each task is processed, the corresponding worker role permanently deletes the message related to this task from the task pool queue. Additionally, each worker role puts a message on the termination indicator queue to indicate successful processing of the task.

Step IV. The web role keeps checking the number of messages in the termination indicator queue to update the user interface with the current progress of the operation. On completion, the web role commits the resultant blob (using API $PutBlockList$) and flushes it as a persistent blob in the Blob storage. The output blob's *URI* is presented to the user for downloading or further processing.

The Queue storage mechanism provided by Azure platform comes handy for fault tolerance during processing. In the event of a worker role failure to process the task message, the message reappears in the queue after a stipulated amount of time.

B. Other versions of Crayons

The distributed dynamic load balancing version of *Crayons* is different from the other two versions in terms of load distribution. In the *centralized load balancing* version, the web role itself downloads the files, parses them, and creates intersection graph (Step 1 and Step 2). Rest of the process is similar to the distributed dynamic load balancing version. The *distributed static load balancing* version is different in that the worker roles do not share their work with other worker roles. Each worker role creates the intersection graph for its share of base layer polygons (statically apportioned based on worker id) and processes its own local tasks, i.e., Step 2 and Step 3 are combined as one step.

III. Performance of Crayons System

We have a maximum quota of 100 Azure cores that we can employ for our experiments. For the centralized version of *Crayons*, 1 core is used by the user interface process, 8 cores are used by the web role that acts as the producer and the rest 91 cores are used by the worker roles acting as the consumers. For the sake of fair comparison, we continue to utilize a maximum of 91 cores for worker role instances (consumers) in both distributed load balanced versions too.

A. End-to-end Speedups

Figure 2 shows the load distribution across two data sets used for experiments. As shown in Figure 2(a), the smaller data set has a skewed load distribution; few base polygons have more than 10K intersecting overlay polygons, while few others have no intersecting overlay polygons at all. The larger data set, shown in Figure 2(b), is comparatively uniformly distributed.

Figure 3 shows the absolute speedup of *Crayons* system for smaller, skewed data set. The baseline sequential timing is calculated over the distributed static version with only one worker role as this version does not store messages in the queue and thus avoids that overhead.

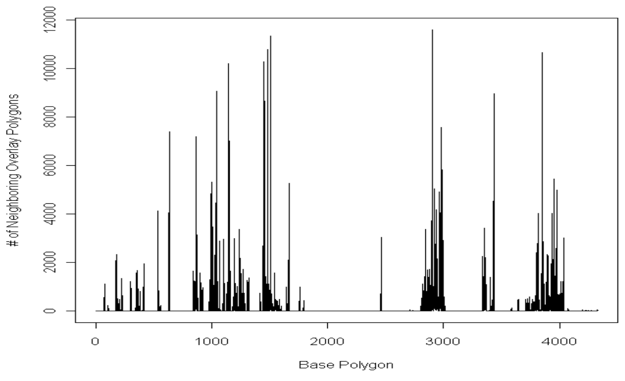
The overall end-to-end (starting from input GML files to producing output GML files) acceleration of *Crayons* system is more than 9x as shown in Figure 3. It can be clearly seen that both of the distributed load balanced versions scale better than the centralized load balanced version. The reason is the demand-supply imbalance due to only one virtual machine working as a producer while the number of consumers keep increasing for the centralized version.

The reason for poor absolute speedup of distributed load balanced versions is the inherent bottlenecks prevalent in Azure platform including simultaneous file download, contention of task queues, and parallel access to Blob storage. Due to these inherent bottlenecks, scaling of such systems on Azure platform will be challenging.

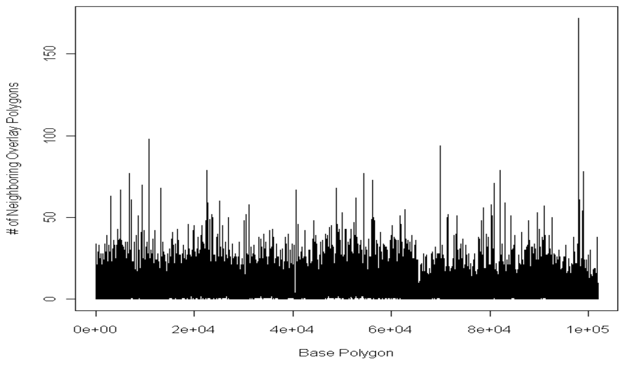
B. Timing Characteristics of Crayons

Figure 4 demonstrates maximum time taken by individual *Crayons* modules over small and large data sets on distributed dynamic load balancing version. The reported subprocess timings represent the time taken from first starting instance of that subprocess at any worker to the last finishing instance of that subprocess at any worker.

The smaller data set suffers from skewed data



(a) Smaller, skewed data set



(b) Large data set

Fig. 2: Load profile of data sets used for experiments

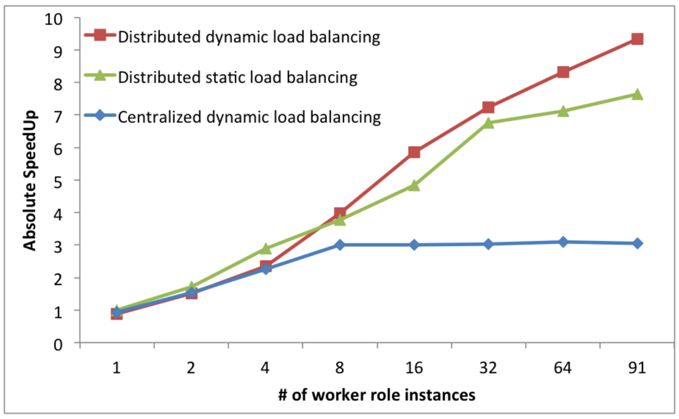
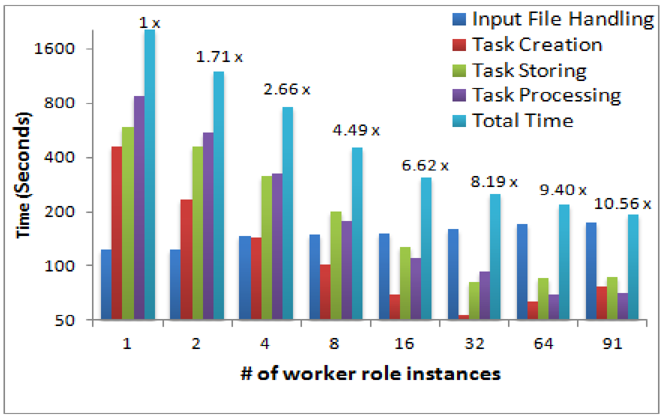
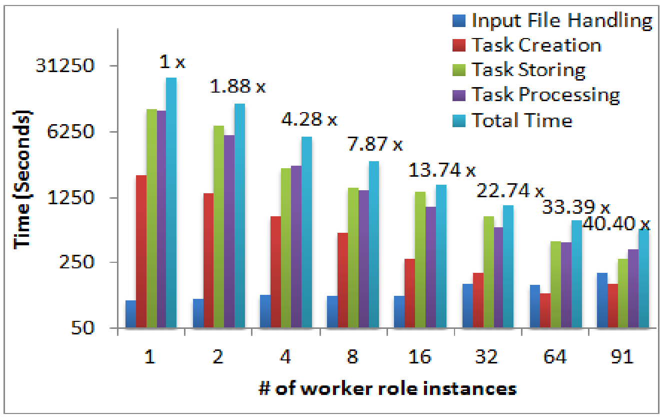


Fig. 3: Absolute Speedup of *Crayons* system for small, skewed data set



(a) Small data set



(b) Large data set

Fig. 4: Execution times for subprocesses and end-to-end speedup for smaller data set

distribution and thus the overall relative speedup is limited to only 10x. On the other hand, for the larger data set, the load is comparatively uniform. Therefore, *Crayons* shows much better performance for this data set. The end-to-end relative speedup of *Crayons* increases to more than 40x. The individual relative speedup for subprocess of task processing is more than 48x, and for the subprocess of task creation it is more than 57x.

IV. Conclusions and Future Work

In this paper we have documented the details of an open-architecture-based overlay-processing system that addresses the critical issues that have hindered the research for an important class of data intensive irregular applications over Azure cloud platform. Our results are very promising showing 10 to 40 fold speedup for end-to-end processing. The system enables experimenting with third party overlay solutions for fundamental GIS operations based on user preferences. An MPI version of *Crayons* has also been developed [17].

We have initiated collaboration with GIS and Health Policy researchers to employ *Crayons* for domain science applications. The system enables experimenting with third party overlay solutions for fundamental GIS operations based on user preferences.

References

[1] HAZUS-MH, "Hazus-MH Overview," http://www.fema.gov/plan/prevent/hazus/hz_overview.shtm, May 2011.
[2] OJWS, "OnEarth, JPL WMS Server," http://onearth.jpl.nasa.gov/, 1936.
[3] Census.gov, "US census data," http://www.census.gov/, December 2011.
[4] GDOT, "Georgia department of transportation," http://www.dot.state.ga.us/Pages/default.aspx, 1916.
[5] USGS, "U.S. geological survey," http://www.usgs.gov/, 1879.
[6] NASA, "Jet propulsion laboratory," http://www.jpl.nasa.gov/, 1936. [Online]. Available: http://www.jpl.nasa.gov/
[7] Open Topography Facility, "Open topography," http://opentopo.sdsc.edu/gridsphere/gridsphere?cid=geonlidar.
[8] C. A. Lee, "A perspective on scientific cloud computing," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 451–459.
[9] G. Turcu, I. Foster, and S. Nestorov, "Reshaping text data for efficient processing on Amazon EC2," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 435–444.
[10] A. Thakar and A. Szalay, "Migrating a (large) science database to the cloud," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 430–434.
[11] J. Li, M. Humphrey, D. Agarwal, K. Jackson, C. van Ingen, and Y. Ryu, "eScience in the cloud: A MODIS satellite data reprojection and reduction pipeline in the Windows Azure platform," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, april 2010, pp. 1 –10.
[12] J. R. Delaney and R. S. Barga, *The Fourth Paradigm: Data Intensive Scientific Discovery*. Microsoft Research,, 2009, ch. Observing the Oceans - A 2020 Vision for Ocean Science.
[13] D. Agarwal, S. Puri, X. He, and S. K. Prasad. Crayons - a cloud based parallel framework for GIS overlay operations. [Online]. Available: http://cs.gsu.edu/dimos/crayons.html
[14] D. Agarwal and S. K. Prasad, "Azurebench: Benchmarking the storage services of the azure cloud platform," in *Parallel Distributed Processing workshops (IPDPSW), 2012 IEEE International Symposium on, to appear*, 2012.
[15] F. Wang, "A parallel intersection algorithm for vector polygon overlay," *Computer Graphics and Applications, IEEE*, vol. 13, no. 2, pp. 74 –81, mar 1993.
[16] A. Murta, "A general polygon clipping library," http://www.cs.man.ac.uk/ toby/alan/software/gpc.html, 1997.
[17] D. Agarwal, S. Puri, X. He, and S. K. Prasad, "A system for GIS polygonal overlay computation on linux cluster - an experience and performance report," in *Parallel Distributed Processing workshops (IPDPSW), 2012 IEEE International Symposium on, to appear*, 2012.