Using MapReduce to Speed Up Storm Identification from Big Raw Rainfall Data

Kulsawasd Jitkajornwanich*, Upa Gupta*, Ramez Elmasri*, Leonidas Fegaras*, and John McEnery†

*Computer Science and Engineering Department
University of Texas at Arlington
{kulsawasd.jitkajornwanich, upa.gupta}@mavs.uta.edu,
{elmasri, fegaras}@cse.uta.edu

†Department of Civil Engineering
University of Texas at Arlington
mcenery@uta.edu

***Abstract*— This paper describes an efficient MapReduce algorithm for converting raw rainfall data into meaningful storm information, which can then be easily analyzed and mined. Our previous work proposed a method to identify relevant storm characteristics from raw rainfall data. The original storm identification system takes too long to produce the summarized storm characteristics, because: (1) the raw rainfall data, which is considered as big data, is stored in a traditional relational database based on CUAHSI (Consortium of Universities for the Advancement of Hydrologic Science, Inc.) ODM (Observations Data Model), which leads to substantial disk I/O; (2) the storm identification algorithm is based on recursion and regular depth-first-search (DFS), which leads to multiple retrievals for parts of the data. In this paper, we obtain a substantial improvement in performance by utilizing MapReduce. We also utilize the original raw rainfall data text files instead of using the data in the relational database. In our experiments, the performance of the new storm identification system is significantly improved compared to the previous one. With this new system, it will dramatically benefit hydrologists in helping them performing rainfall-related analysis (both location-specific and storm-specific) such as flood prediction using our identified storms.**

Keywords-storm analysis; rainfall; big data; MapReduce; distributed computing; CUAHSI

I. INTRODUCTION

This paper describes an efficient MapReduce algorithm for converting raw rainfall data into meaningful storm information, which can then be easily analyzed and mined. Our previous work [1] proposed a method to identify relevant storm characteristics from raw rainfall data. The original storm identification system takes too long to produce the summarized storm characteristics, because: (1) the raw rainfall data, which is considered as big data [7][8], is stored in a traditional relational database based on CUAHSI (Consortium of Universities for the Advancement of Hydrologic Science, Inc.) ODM (Observations Data Model) [17][18][9], which leads to substantial disk I/O; (2) the storm identification algorithm is based on recursion and regular depth-first-search (DFS), which leads to multiple retrievals for parts of the data. In this paper, we obtain a substantial improvement in performance by utilizing MapReduce. We also utilize the original raw rainfall data text files instead of using the data in the relational database. In our experiments, the performance of the new storm identification system is significantly improved compared to the previous one. With this new system, it will dramatically benefit hydrologists in helping them performing rainfall-related analysis (both location-specific and storm-specific) such as flood prediction using our identified storms.

Our raw rainfall data, called MPE (Multi-sensor Precipitation Estimates) [19][20][21], is estimated by using combination of radars and physical rain gauges (multi-sensors) and is retrieved from National Weather Service (NWS) - West Gulf River Forecast Center (WGRFC) [19]. The raw data is supplied as hourly text files using the HRAP (Hydrologic Rainfall Analysis Project) standard grid coordinate system [20][17]. The raw rainfall data is converted into a relational database in order to follow the CUAHSI ODM standard, which was required for the HydroDesktop system [22] that allows hydrology users to search the rainfall data.

Our previous storm identification system used the relational data as input. Due to the relational database I/O overhead, and the tremendous amount of data, system performance was too slow. The data covers 17 years (1996 - 2012) of historical hourly precipitation, which is translated to 8.004123763 billion records in the database. We receive the rainfall data on an hourly basis covering 4 states (Texas, Colorado, New Mexico, and Louisiana) (mainly Texas) and part of Mexico (see Figure 1) covering 69,830 site locations. The number of records inserted per hour, day, month, and year is 69,830, 1,675,920, 50,277,600, and 603,331,200, respectively.

In this paper, we develop more efficient storm identification algorithms using the original text file formats, and the MapReduce framework to parallelize the processing.

Figure 1. Coverage of WGRFC observations [19][21]

 ISBN: 978-1-61208-271-4

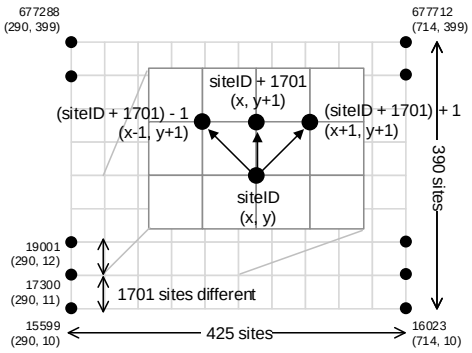

Figure 2. Relationships among neighboring sites

MapReduce is a programming paradigm developed by Google in 2004 [5] and now becoming a new standard for distributed computing. Our previous storm identification algorithms, based on recursion and depth-first-search, traverse the data exhaustively without taking advantage of the known regular grid structure of the raw rainfall data. We greatly improved the performance by using the original raw rainfall data and applying MapReduce to every component of the storm identification process. However, only local storm and hourly storm identifications (also known as event separator and sub storm identification in [1]) are discussed in this paper. The details of each component are described in Sections IV and V, which are MapReduce for local storm and hourly storm identifications, respectively. First, we review the storm identification concepts from [1] in Sections II and III. Section VI discusses experimental results. Related work is discussed in Section VII.

II. INPUT DATA STRUCTURE

The raw rainfall data is supplied as text files. The file name indicates a particular date and time (hourly, e.g., 2011041323_2011041400), and includes the precipitation data for all sites during that hour. Each row consists of row number, site id, and precipitation value (inches). The data is ordered by site id in a row major order from west to east and south to north. Sites are in an HRAP regular grid and four kilometers apart to north, south, east, and west. Each row in the grid has 425 sites and each column has 390 sites as shown in Figure 2. Because of the systematic grid structure, given any site, we can determine the neighboring sites by using the formulas in Figure 2. Moreover, given any site id, we can determine its HRAP local X and Y coordinates, and vice versa using the following equations: (1) and (2).

$$x = 290 + \big((siteID - 15599)\ mod\ 1701\big) \quad (1)$$

$$y = 10 + \big((siteID - 15599)\ div\ 1701\big) \quad (2)$$

III. STORM-RELATED CONCEPTS

In this section, we review some key components of the previous work [1] that are needed for this paper. Two main components are: (1) storm formalization and (2) storm identification process.

A. Storm Formalization

We formalize storms into three different categories (local storms, hourly storms, and overall storms), the goal of which is to develop a storm identification process and storm characteristics analysis. The following is some terminology needed for the storm formalization.

- *storm duration*: the time length over which precipitation occurs (hours) [23].
- *storm coverage*: the number of sites covered by a storm.
- *storm area*: the total area of a storm.

1) Local Storms

Generally speaking, *local storm* is a site-specific storm, which considers each site location independently when analyzing a storm. An example of local storms is the set of storms that occurred at site location 586987 last month. Local storm is one type of storm, which was researched by most hydrologists [11][12][13][14]. This may be due to the traditional way of storm analysis, which does the analysis primarily based on how raw rainfall data are collected and stored without applying distributed computing technology.

Formally, a local storm is a set of time points and associated rainfall data at a particular spatial site. Two distinct local storms are separated by at least *h* consecutive time points with zero precipitation, where *h* is called the *inter-event time* [11][12][16]. In this paper, inter-event time (*h*) is set to 6 hours as suggested in [11][12]. Several consecutive time points with zero precipitation within a local storm, however, are allowed as long as it is less than *h* time points. For any local storm, there will not be a subsequence of *h* or more consecutive zeroes in the series. Figure 3 shows some examples of local storms at site id, 586987. Some storm characteristics for this storm type include:

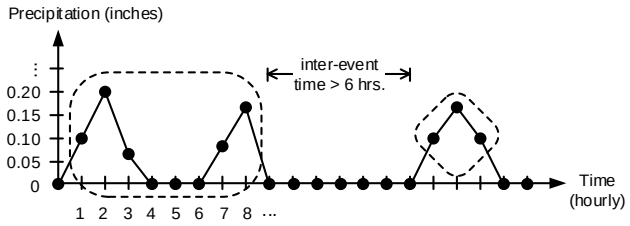

Figure 3. Examples of local storms at site id, 586987

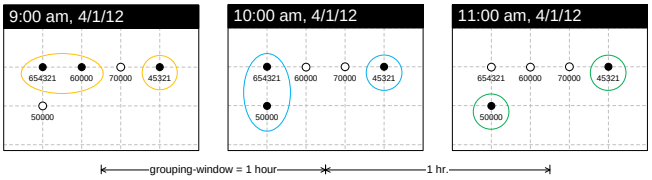

Figure 4. Examples of hourly storms at different hours on 4/1/12

- *storm depth*: the amount of precipitation occurring throughout the storm duration at a particular site [23].
- *storm intensity*: the storm depth divided by the storm duration (inches per hour) [23].

2) Hourly Storms

Informally, *hourly storm* is a time-specific storm, which has an orthogonal concept to local storm. It considers each hour independently when analyzing a storm. An example of hourly storms is the set of storms that occurred between 9:00 am and 10:00 am today. Hourly storm considers a specific time point (an hour) instead of considering a particular site location. In other words, local storm fixes one site and covers its data over many time points, whereas hourly storm fixes a time point and covers its data over many adjacent sites. Figure 4 shows some examples of hourly storms at different hours on April 1, 2012.

Formally, an hourly storm is a set of adjacent sites of local storms at a particular hour. However, a more relaxing definition can also be applied as discussed in our previous work [1] as *space-tolerance*. The concept of space-tolerance is to allow indirect neighboring sites to be considered as part of the same hourly storm. In this paper, we use the original definition of hourly storm, which takes into account only direct neighboring sites when identifying hourly storms. The following are storm characteristics that are applicable for this type of storm:

- *storm sites total*: the total amount of precipitation occurring at a particular hour for the sites of an hourly storm.
- *storm average*: the average precipitation (per site) for an hourly storm.

3) Overall Storms

Unlike local storm and hourly storm that consider either a site location or time (an hour) independently, it considers both location and time together when analyzing a storm. So, the result is the capture of storm as a whole, called *overall storm*, which can capture storm movement and other storm characteristics that could not be found in most hydrology papers [11][12][13][14]. An overall storm is built upon hourly storms. Some examples of overall storms are shown in Figure 5.

Formally, an overall storm is a set of hourly storms that meet two requirements: (1) *grouping-window* and (2) *spatial-window*. Grouping-window is the maximum time

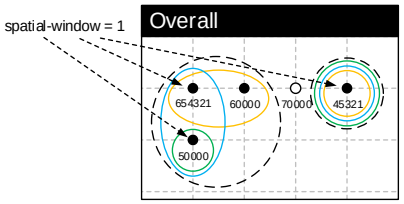

Figure 5. Examples of overall storms

interval within which hourly storms will be considered to be part of the same storm whereas spatial-window is the minimum number of common site(s) shared between two hourly storms. This formalization allows hourly storms that go to the same direction be considered together. However, in a rare situation, it is also possible that two different paths of hourly storms with different origins and/or destinations could end up being part of the same overall storm. In such case, the final path of the overall storm will be averaged based on those two paths. In this work, grouping-window and spatial-window are set to 1 hour and 1 site, respectively. Overall storm characteristics include:

- *storm overall depth*: the total amount of precipitation occurring throughout the storm duration across the hourly storms.
- *storm overall intensity*: the storm overall depth divided by the storm duration (inches per hour).
- *storm overall average*: the average precipitation (per site) for an overall storm.

B. Storm Identification Process

The main goal of our storm identification system is to analyze storms as a whole. Since a storm can start at one place and stop at another, we slice the whole storm into several pieces by hour. We then assemble each slice back together into the original overall storm. Each slice of storm is, in fact, an hourly storm. Figure 6 shows architecture of our previous storm identification system.

The storm identification process can be divided into three main components: (1) event separator (to identify local storms), (2) sub storm identification (to identify hourly storms), and (3) main storm identification (to identify overall storms). The architecture of our new storm identification system is shown in Figure 7.

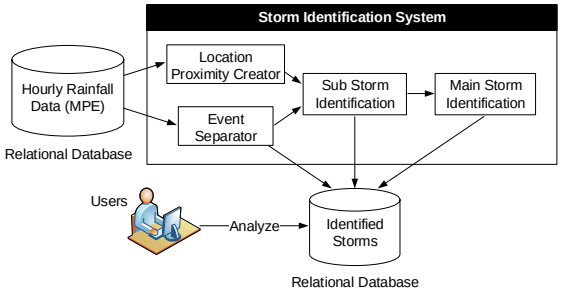

Figure 6. Architecture of previous storm identification system

IV. MAPREDUCE FOR LOCAL STORM IDENTIFICATION

The local storm identification identifies the storms at a particular site and specifies each storm duration (in hours) at that site. The previous implementation of local storm identification required the selection of data from the relational database and then sorting them. The computation is done based on the selected sorted data and the result is inserted back to the database. The selection, sorting, and insertion required substantial execution time, making it impractical to analyze the whole raw data.

Our new algorithms utilize MapReduce, and use the rainfall data text files as input. Each raw rainfall file contains the precipitation value of all the sites for a particular hour and hence, for the analysis of local storm, we need to group all the precipitation values by site and order them by time. Once all the values for a site are grouped together and ordered, then we can find all the local storms that occurred at that site. Thus, the local storm analysis contains two steps: (1) grouping precipitation values by site and ordering them by time and (2) finding the local storms for a site from the grouped values. In the MapReduce framework, there are three main phases: (1) map phase, (2) sorting and shuffling phase, and (3) reduce phase. The first two phases of MapReduce are used to perform the first step of our local storm identification and the reduce phase is used to find the local storms at the particular site.

Algorithm 1. Local Storm Identification

```
    Input:
        - Text file-format rainfall data
    Output:
        - Local storms data in text file format
1:  class MAPPER
2:  function MAP(key object, value line)
3:      key <-- (line.siteId, line.time)
4:      value <-- (line.precipValue, line.time)
5:      Emit(key, value)
6:  class REDUCER
7:  function REDUCE(key siteId, [val1, val2, ...])
8:      timeList, precipRec <-- null //timeList.size = inter-event + 2
9:      interEventTime <-- 0, lsId <-- 1
10:     timeList.Add(firstNonZeroPrecip.GetTime())
11:     precipRec.Add(firstNonZeroPrecip.GetPrecipValue())
12:     for all val ∈ values [val1, val2, ...] do
13:         precipRec.Add(val.GetPrecipValue())
14:         if (val.GetPrecipValue() = 0) then
15:             timeList.Add(val.GetTime())
16:             interEventTime++
17:         else
18:             tempTime <-- timeList[0], Clear(timeList)
19:             timeList.Add(tempTime; val.GetTime())
20:         end if
21:         if interEventTime ≥ 6 then
22:             initialTime, finalTime <-- timeList[0], timeList[1]
23:             value.Set(initialTime, finalTime, precipRec)
24:             Emit(siteId, lsId, value)
25:             Clear(timeList; precipRec), lsId++
26:         end if
27:     end for
```

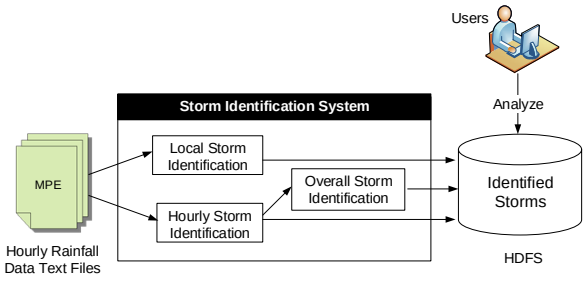

Figure 7. Architecture of current storm identification system

The pseudo code for the implementation for local storm analysis in the MapReduce framework is shown in Algorithm 1. Each of the map tasks takes one raw rainfall file and processes it line by line emitting the site and time together as the key and time and precipitation value together as the value. We take advantage of the key-comparator class and grouping-comparator class of MapReduce to group the data on the basis of site id and then sort them by time. The reducer gets a site id as a key and list of precipitation values sorted by time. This list is processed sequentially to identify all the local storms at that particular site.

V. MAPREDUCE FOR HOURLY STORM IDENTIFICATION

The second main component is hourly storm identification, the goal of which is to identify hourly storms for each particular hour.

In the previous approach [1], we assume that any non-zero precipitation site can be part of the hourly storm, meaning it can start at one site and stop at a very farther site as long as there are some connections among them. As a result, we implemented DFS to keep track of every possible site and perform site node revisiting when needed. This, however, led to a high time complexity problem. In addition, the algorithm interacts with the data in the relational database, which causes a large overhead.

In the new approach, the program is designed specifically to take full advantage of the original raw rainfall data text file structure. Since the grid (HRAP) is known and we know exactly which site is a neighbor of which, only those candidate neighboring sites need to be checked. Unlike previous approach which uses DFS to keep track of node, we use linked lists and append them together as we scan when necessary. Moreover, since the data in each text file is stored in row major order, we scan each grid row once. An overview of the hourly storm identification process is shown in Figure 8.

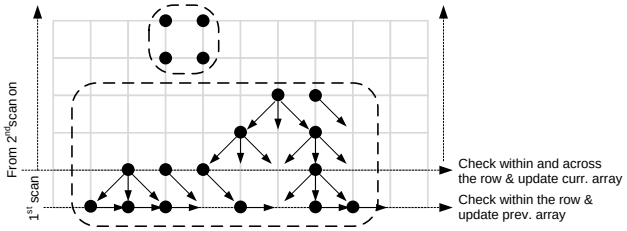

Figure 8. Overview of hourly storm identification process

Algorithm 2. Hourly Storm Identification

```
    Input:
        - Text file-format rainfall data
    Output:
        - Hourly storms data in text file format
1:  class MAPPER
2:  function SETUP()
3:      prev.InitializeArray(), curr.InitializeArray()
4:      hourlyStorms.InitializeArrayOfLinkedList()
5:      id <-- 0
6:  function MAP(key object, value r)
7:      if r ∈ first bottom grid sites then
8:          if r.precip = 0 then
9:              prev[r.site].hsId <-- 0 //no hourly storm
10:         else
11:             if r.site = first site or r.leftNeighborPrecip = 0 then
12:                 prev[r.site].hsId <-- id++
13:                 temp <-- CreateLinkedList(r.site)
14:                 hourlyStorms.AddLinkedList(temp)
15:             else
16:                 prev[r.site].hsId <-- id
17:                 hourlyStorms.GetLinkList(id).Add(r.site)
18:             end if
19:         end if
20:     else if r ∈ next above grid sites then
21:         if r.precip ≠ 0 then
22:             if r.site = first site or r.leftNeighborPrecip = 0 then
23:                 CheckPrevious(r, id, prev, curr, 1)
24:             else
25:                 CheckPrevious(r, id, prev, curr, 0)
26:             end if
27:         else
28:             curr[r.site].hsId <-- 0 //no hourly storm
29:         end if
30:     else
31:         prev <-- curr
32:     end if
33: function CLOSE()
34:     Emit(hourlyStorms)
35: function CHECKPREVIOUS(r, id, prev, curr, flag)
36:     if flag = 1 then
37:         if hsIds of all 3 neighbors of r in prev. array = 0 then
38:             curr[r.site].hsId <-- id++
39:             temp <-- CreateLinkedList(r.site)
40:             hourlyStorms.AddLinkedList(temp)
41:         else
42:             minId <-- MinHsId(r.all3Neighbors in prev. array)
43:             curr[r.site].hsId <-- minId
44:             hourlyStorms.GetLinkedList(minId).Add(r.site)
45:             UpdateHsId(r.neighbors, minId)
46:             minId <-- 0 //reset minId
47:         end if
48:     else
49:         curr[r.site].hsId <-- id
50:         hourlyStorms.GetLinkedList(id).Add(r.site)
51:         if hsId of r's southeast neighbor in prev ≠ id then
52:             UpdateHsId(r.southeastNeighbor, id)
53:         end if
54:     end if
```

The program starts from the very bottom grid row to the top by calling map function for each line in the text file. It begins to identifying hourly storms as soon as it reads in the data in order to minimize the number of checking. The data are then kept in two arrays called *previous* and *current arrays*, which are two-dimensional arrays and contains site ids and hourly storm ids. The current array always does the identification based on the previous array. There are two main parts of the program. The first part (line: 7-19) is executed only once for the very bottom row in a grid whereas another part (line: 20-32) is executed for the rest. The first part identifies hourly storms within the same row whereas the other part identifies hourly storms within and across the rows simultaneously. At the end of each row scan, the hourly storms so far are identified and are kept in an array of linked lists called *hourly storms list*, in which index of array indicates hourly storm id and linked list contains a set of adjacent non-zero precipitation sites of the hourly storm. When reached the last row, the final hourly storms are produced and already kept in the hourly storms list.

Since the raw rainfall data files are independent from each other, in which each file records hourly precipitation for an individual hour, MapReduce can easily be applied. Each hourly file is sent to different mapper nodes for the identification of hourly storms. At the closing of mapper, all hourly storms identified within the hour will be written back to a disk. Currently, no reducer is needed because there is no need to group the data or sort them in any order. The raw files, by themselves, are already grouped and sorted by site id in a row-major order as mentioned in Section II. An algorithm for hourly storm identification is shown in Algorithm 2.

VI. EXPERIMENTAL RESULTS

In the previous approach, the experiment was performed on the rainfall dataset, resided in a relational database, using a single server. The server runs on Microsoft® Windows Server® 2008 Enterprise operating system with 2.83 GHz Intel® Xeon® quad-core processors, 20 GB of RAM, 500 GB of local disk, and 10 TB of external disk. In the new approach, the experiment was performed on the same dataset that is in the original text file format rather than relational format using a Hadoop® cluster [6] of 1 frontend server and 18 worker nodes. Each worker node contains 3.2 GHz Intel® Xeon® quad-core processors, 4 GB of RAM and 1.5 TB of local disk allocated to HDFS. The server has the same specification but with 3 TB of local disk. The cluster is set up by using ROCKS Cluster 6.3 OS and then installing Hadoop® 1.0.3 on every node.

Both local storm and hourly storm identifications are analyzed over 16 months of data. The data has 11,488 hours and is 10 GB in size. The raw files are in text format. Each file is for all sites during a single hour and is zipped into one gunzip file. These files are fed into to the MapReduce job for the storm analysis. There are separate map tasks for each

 ISBN: 978-1-61208-271-4

of the files because each file is gunzipped into separate .gz files.

The comparison between the time taken by the previous implementations and the new MapReduce implementations is shown in Table I. Please also note that the processing time does not include the time taken to load the data into HDFS/SQL. In addition, each experiment was performed 10 times and an average processing time is calculated.

The experiments of the new approach give the same results for both local storms and hourly storms as the previous approach but is executed significantly faster. The new approach allows programs to be executed distributedly on multiple machines and hence the efficiency of the storm analysis is increased. For local storm (LS) identification, the time improved to 2.79 hours, compared to 53.44 hours in the previous approach. For hourly storms (HS), the MapReduce (MR) took 0.45 hours, compared to 6.78 hours in the previous method (DFS).

TABLE I. EXPERIMENTAL RESULTS

Regions / Number of Raw Data	Sites	Processing Time (in hours)			
		Previous work		*Current work*	
		LS (sec/site)	HS (DFS)	LS (sec/site)	HS (MR)
1. East Texas (48,953,130)	4,643	8.67 6.72s/site	1.44	<---------- 2.79 hours for all 10 regions (0.27seconds/site) ---------->	<---------- 0.45 hours for all 10 regions ---------->
2. Edwards Plateau (73,415,532)	6,962	8.72 4.51s/site	1.23		
3. High Plains (31,711,927)	3,008	4.50 5.39s/site	0.32		
4. Low Rolling Plains (24,965,521)	2,368	3.35 5.10s/site	0.28		
5. North Central (59,082,957)	5,604	8.66 5.56s/site	1.17		
6. South Central (31,102,334)	2,949	4.28 5.22s/site	0.67		
7. South Texas (31,949,386)	2,933	3.97 4.87s/site	0.48		
8. Lower Valley (5,324,898)	601	0.55 3.32s/site	0.07		
9. Trans-Pecos (65,136,216)	6,177	6.86 4.00s/site	0.55		
10. Upper Coast (22,863,789)	2,168	3.88 6.45s/site	0.57		
TOTAL	37,413	53.44 5.14s/site	6.78		

VII. RELATED WORK

There are two main parts of related work: (1) storm characteristics analysis and (2) MapReduce framework for spatial data computing.

A. Storm Characteristics Analysis

In most hydrology papers, most rainfall data analysis is either site-specific or region-specific and only few do storm analysis by integrating them across sites [10][11][12][13][14][15][16]. Asquith et al. [11][16][15] studies storm statistical characteristics by looking at the means of storm inter-event time, depth, and duration. In [10][12][14], Overeem and Asquith study storm characteristics through their DDF (depth-duration-frequency) properties. Lanning-Rush [13] studies storm characteristics by focusing on their extreme precipitation (EP) values. Within these, only a small amount of data and limited number of gauges were used. The storm analysis was conducted mainly based on how raw rainfall data is collected and stored, which is by location stored in different folders. This might be a reason why there are not many programs developed to process rainfall data across sites. Consequently, the flexibility in analyzing overall storm characteristics was lacking.

Our work, on the other hand, allows rainfall data to be analyzed in both location-specific (site-specific and region-specific) and storm-specific. Additionally, a much larger amount of data across a large number of gauges on HRAP standard grid coordinates can be analyzed. Our efficient algorithms were custom designed to take advantage of the format of the original raw rainfall data, as well as adopt renowned distributed computing technology, called MapReduce, to analyze storms in a storm-specific manner. This enables flexibility in analyzing overall storm characteristics.

B. MapReduce Framework for Spatial Data Computing

MapReduce has become the de-facto framework for the data-intensive applications. It is now being used for big data related to geography, sciences, humanities, statistics, etc. There has been previous work for spatial data analysis in MapReduce. Cary [2] shows the construction of R-Tree index from spatial data in MapReduce. It uses the mappers to partition the data and then every partition is sent to a different reducer which in turn build the R-Tree index on the input. Google used the MapReduce framework to study road alignments by combining satellite and vector data [3]. The work focused more on the complexity of the problem than the implementation in MapReduce. Hadoop® was also used to build octrees for later use in earthquake simulations at a large scale [4]. Octrees were built in the bottom up fashion in their approach. Mappers were used to first generate the leaf nodes and then reductions were performed to merge two homogeneous leaf nodes into a sub tree. This was done in iterations to build the final sub tree.

 ISBN: 978-1-61208-271-4

VIII. CONCLUSION AND FUTURE WORK

A. Conclusion

In this work, we use the MapReduce framework to analyze large amounts of raw rainfall data. With this new system, the original input data structure was fully utilized in order to create more efficient algorithms for storm identification. It eliminates the major performance issue with the previous system, which mostly has to do with the retrieval of relational data overhead. The experimental results show significant improvement on both local storm and hourly storm identifications processes. This will allow hydrologists to perform: (1) storm analysis (both location-specific and storm-specific) such as storm frequency and characteristics analysis and flood prediction and (2) storm mining such as clustering on types of the storm and trajectory analysis, more efficiently.

B. Future Work

We will work on the computation of overall storm identification using the MapReduce framework. We will also be working on parallelizing the computation of storm area, storm center, and within storm variations [24] by the use of MapReduce framework.

REFERENCES

[1] K. Jitkajornwanich, R. Elmasri, C. Li, and J. McEnery, "Extracting Storm-Centric Characteristics from Raw Rainfall Data for Storm Analysis and Mining," Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data (ACM SIGSPATIAL BIGSPATIAL'12), 2012, pp. 91-99.

[2] A. Cary, Z. Sun, V. Hristidis, and N. Rishe, "Experiences on Processing Spatial Data with MapReduce," Proceedings of the 21st International Conference on Scientific and Statistical Database Management (SSDBM'09), 2009, pp. 302-319.

[3] X. Wu, R. Carceroni, H. Fang, S. Zelinka, and A. Kirmse, "Automatic Alignment of Large-Scale Aerial Rasters to Road-maps, Geographic Information Systems," Proceedings of the 15th ACM International Symposium on Advances in Geographic Information Systems (ACM GIS'07), 2007.

[4] S. W. Schlosser et al., "Materialized Community Ground Models for Large-Scale Earthquake Simulation," Proceedings of the 2008 ACM/IEEE International Conference for High Pergormance Computing, Networking, Storage and Analysis (SC'08), 2008.

[5] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI'04), 2004.

[6] C. Lam, Hadoop in Action. Dreamtech Press, New Delhi, 2011.

[7] B. Franks, Taming The Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics. John Wiley & Sons, Inc., Hoboken, New Jersey, 2012.

[8] DevZone, Big Data Bibliography. O'Reilly Media, 2011.

[9] R. Elmasri and S. Navathe, Fundamentals of Database Systems, 6th ed. Pearson Education, Massachusetts, 2010.

[10] A. Overeem, T. A. Buishand, and I. Holleman, "Rainfall Depth-Duration-Frequency Curves and Their Uncertainties," Journal of Hydrology, vol. 348, 2008, pp. 124-134.

[11] W. H. Asquith, M. C. Roussel, T. G. Cleveland, X. Fang, and D. B. Thompson, "Statistical Characteristics of Storm Interevent Time, Depth, and Duration for Eastern New Mexico, Oklahoma, and Texas," Professional Paper 1725. U.S. Geological Survey (USGS), 2006.

[12] W. H. Asquith, "Depth-Duration Frequency of Precipitation for Texas," Water-Resources Investigations Report 98-4044. U.S. Geological Survey (USGS), 1998.

[13] J. Lanning-Rush, W. H. Asquith, and R. M. Slade, "Extreme Precipitation Depth for Texas, Excluding the Trans-Pecos Region," Water-Resources Investigations Report 98-4099. U.S. Geological Survey (USGS), 1998.

[14] W. H. Asquith and M. C. Roussel, "Atlas of Depth-Duration Frequency of Precipitation Annual Maxima for Texas," Scientific Investigations Report 2004-5041 (TxDOT Implementation Report 5-1301-01-1). U.S. Geological Survey (USGS), 2004.

[15] W. H. Asquith, D. B. Thompson, T. G. Cleveland, and X. Fang, "Synthesis of Rainfall and Runoff Data used for Texas Department of Transportation Research Projects 0-4193 and 0-4194," Open-File Report 2004-1035. U.S. Geological Survey (USGS), 2004.

[16] W. H. Asquith, "Summary of Dimensionless Texas Hyetographs and Distribution of Storm Depth Developed for Texas Department of Transportation Research Project 0-4194," Report 0-4194-4. U.S. Geological Survey (USGS), 2005.

[17] J. S. Horsburgh, D. G. Tarboton, D. R. Maidment, and I. Zaslavsky, "A Relational Model for Environmental and Water Resources Data," Water Resources Research, 2008.

[18] Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI), "ODM Databases," retrieved: October 26, 2011, from: http://his.cuahsi.org/odmdatabases.html.

[19] National Oceanic and Atmospheric Administration (NOAA), "National Weather Service River Forecast Center: West Gulf RFC (NWS-WGRFC)," retrieved: December 31, 2011, from: http://www.srh.noaa.gov/wgrfc/.

[20] NOAA's National Weather Service, "The XMRG File Format and Sample Codes to Read XMRG Files," retrieved December 31, 2011, from: http://www.nws.noaa.gov/oh/hrl/dmip/2/xmrgformat.html.

[21] J. McEnery, "CUAHSI HIS: NWS-WGRFC Hourly Multi-sensor Precipitation Estimates," retrieved: December 31, 2011, from: http://hiscentral.cuahsi.org/pub_network.aspx?n=187.

[22] Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI), "HydroDesktop," retrieved: October 26, 2011, from: http://his.cuahsi.org/hydrodesktop.html.

[23] Virginia Department of Conservation and Recreation, "Stormwater Management: Hydrologic Methods," retrieved: May 2, 2012, from: http://dcr.cache.vi.virginia.gov/stormwater_management/documents/Chapter_4.pdf.

[24] A. Suyanto, P. E. O'Connell, and A. V. Metcalfe, "The Influence of Storm Characteristics and Catchment Conditions on Extreme Flood Response: A Case Study Based on the Brue River Basin," U.K. Surveys in Geophysics, vol. 16, 1995, pp. 201-225.

 ISBN: 978-1-61208-271-4