A MapReduce Technique to Mosaic Continental-Scale Weather Radar Data in Real-time

Valliappa Lakshmanan[1,2], Timothy W. Humphrey[2,3]

***Abstract*—It is necessary to create continental-scale mosaics of radar data in real-time for applications ranging from precipitation estimation, hail diagnosis and tornado warnings to public awareness. Because of the high temporal and spatial resolution of data available from the United States' network of weather radars, creating radar mosaics in real-time has been possible only through compromises on the quality, timeliness or resolution of the mosaics. MapReduce is a programming model that can be employed for processing and generating large data sets by distributing embarrasingly parallel computations and data storage across a distributed cluster of machines. In this paper, a MapReduce approach to computing radar mosaics on a distributed cluster of compute nodes is presented. The approach is massively scalable, and is able to create high-resolution 3D radar mosaics over the Continental United States in real-time.**

***Index Terms*—weather radar, radar mosaic, MapReduce**

I. Mosaics of Weather Radar Data

Real-time 3D multi-radar grids of weather radar data (such as that in Figure 1) are essential to many meteorological applications. Weather radar is the preferred instrument for remotely sensing precipitation over land [1], for carrying out "nowcasts" of thunderstorms [2] and is a critical input into stormscale [3] and mesoscale [4] numerical weather prediction (NWP) models. However, the conical nature of the radar sensing volume is challenging for applications that require range-insensitive values [5] or constant-sized grid cells [6] througout the domain, or operate in earth-relative coordinate systems or projections [7]. For this reason, radar data are commonly mapped to a Cartesian grid before it is used for nowcasting, precipitation estimation, numerical weather prediction or flood forecasting.

It is often desired that radar data from single radars be mosaiced into a multi-radar Cartesian grid before it is used by automated applications or displayed to end-users. The range of a single radar is limited (420 km at the lowest tilt in the case of US WSR-88D radars) and so the domain of interest for a meteorological application is often covered by multiple radars [8]. Because weather systems move over time, it is preferable to process and display weather radar data seamlessly over large domains, regardless of the radar from which the data originated [9], [10].

Furthermore, it is desirable that the multi-radar Cartesian grid be a 3D grid. The need for diagnostics and displays of 3D multi-radar weather data was illustrated by [11]. The Vertical

Corresponding author: V Lakshmanan, University of Oklahoma, Norman OK lakshman@ou.edu [1] Cooperative Institute of Mesoscale Meteorological Studies, University of Oklahoma [2] National Severe Storms Laboratory, Norman, OK [3] School of Meteorology, University of Oklahoma

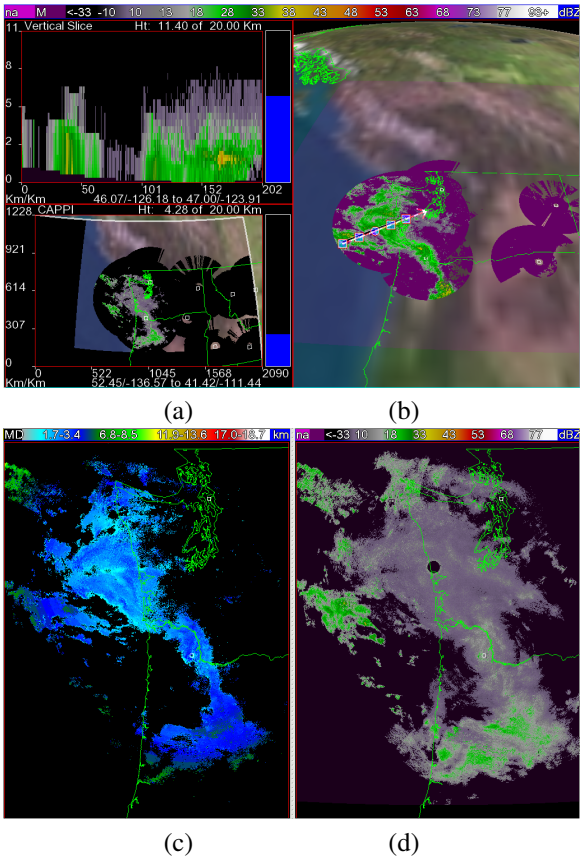


Fig. 1. A real-time 3D multi-radar grid of weather radar reflectivity data and products derived from it. (a) Vertical and horizontal slices through 3D grid (b) One level of the 3D grid (c) Height of 18 dBZ echo (d) Average reflectivity between the 0 C and -20 C isotherms.

Integrated Liquid (VIL), introduced in [12], is commonly employed in weather forecasting as a proxy for a storm's capability to produce severe weather such as hail, lightning or tornadoes. As shown by [11] in Figure 2, storm cell VIL values computed using data from multiple radars are more robust than values computed using data from a single radar. A naive or busy forecaster who saw the trend in single-radar VIL may have wrongly believed that the storm was weakening. The multi-radar VIL, however, is more robust and provides better guidance since it is not as affected by range effects. This is true of many weather diagnostics derived from a single radar because the radar beam spreads and covers a greater volume with increasing range and because there is a conical area right above the radar that is not sensed by the radar [9], [13].

In this paper, we focus on the challenges in creating 3D multi-radar Cartesian grids in real-time over domains that are

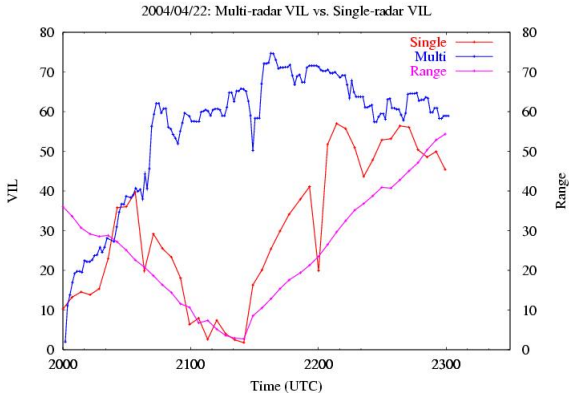


Fig. 2. VIL computed on cells detected from data blended from individual radars is more robust than VIL computed on storms cells identified from a single radar. As the storm approaches the radar, the single radar VIL drops since higher elevation data are unavailable. The multi-radar product does not have that problem. Graph from [11].

continental-scale, i.e. in the thousands of kilometers in each dimension. For smaller domains, even a brute-force solution will suffice. It is when a network has 100 or more radars spread over an area larger than $10^6 km^2$ that scaling challenges start to arise. There are currently only a few such regions around the world – the United States, Europe, China and India. But as the European experience of combining together data from 17 countries [14] illustrates, weather radar data are useful when mosaiced even across countries. Thus, the massively scalable approach of this paper should be applicable in other areas of the world as surface-based weather radar networks proliferate.

A. Mosaic formulae

[15] introduced an intelligent-agent model for mosaicing radar reflectivity data, and that formulation is followed in this paper because it results in mosaics that transform the input spherical coordinate data into a Cartesian 3D grid with no missed areas ("holes"), while retaining to a large extent the original resolution[1] and structure of the remotely sensed storms.

Whenever a new elevation tilt is received from a radar, an agent is created for each of the range gates in that elevation tilt. This agent encodes the coordinates in the radar-centric spherical coordinate system (range, azimuth, elevation angle), the elevation start time and the radar from which the observation came. The agent's coordinates in the earth-centric latitude-longitude-height coordinate system of the resulting 3D grid are then computed following the 4/3 effective earth radius model of [16] (i.e., assuming standard beam propagation). In order to avoid unfilled areas in the output 3D grid, the mapping is carried for every voxel in the output grid instead of mapping the radar data in the spherical coordinate system to the cylindrical-equidistance projection.

Given a voxel in the output 3D grid at latitude α_g, longitude β_g and height h_g above mean-sea-level, and a radar located at (α_r,β_r,h_r), the range gate from that radar that fills that voxel at distance r from the radar on a radial at an angle a from due-north and on a scan tilted e to the earth's surface where a is given by:

$$a = \sin^{-1}(\sin(\pi/2 - \alpha_g)\sin(\beta_g - \beta_r)/\sin(s/R)) \quad (1)$$

where R is the radius of the earth and s is the great-circle distance, given by [17]:

$$\begin{aligned} s = R\cos^{-1}(\cos(\pi/2 - \alpha_r)\cos(\pi/2 - \alpha_g) + \\ \sin(\pi/2 - \alpha_r)\sin(\pi/2 - \alpha_g)\cos(\beta_g - \beta_r)) \end{aligned} \quad (2)$$

the elevation angle e is given by:

$$e = \tan^{-1}\frac{\cos(s/k_eR') - \frac{k_eR'}{k_eR'+h_g-h_r}}{\sin(s/k_eR')} \quad (3)$$

where k_e under the same standard atmospheric conditions may be assumed to be $4/3$ [16], R' is $R + h_r$ and the range r is given by:

$$r = \sin(s/k_eR')(k_eR' + h_g - h_r)/\cos(e) \quad (4)$$

Since the sin^{-1} function has a range of $[-\pi/2, \pi/2]$, the azimuth, a, can be mapped to the correct quadrant ($[0, 2\pi]$) by considering the signs of $\alpha_g - \alpha_r$ and $\beta_g - \beta_r$. In our computations, we approximated R' by R since $h_r << R$.

A voxel at α_g, β_g, h_g can be affected by any range gate that includes (a,r,e). Assuming that a radar beam can be neglected beyond twice its half-power beamwidth, there is a finite volume implied by these equations. Intelligent agents corresponding to all the radar range gates that cover a voxel collaborate to set the final value of the voxel in the output 3D grid. Several weighting schemes have been studied ([18], [19]), with [20] demonstrating that the best weighting scheme is one that carries out (a) nearest-neighbor mapping in range and azimuth (b) linear vertical or vertical-horizontal interpolation between elevation data from the same radar and (c) weighting of different radars' contributions at a voxel by an exponential function that falls with distance. This is the weighting scheme that is carried by the intelligent agents at a voxel. Thus, amongst multiple contributors all belonging to the same elevation scan, the one closest to the voxel center is selected. Amongst contributors that come from the same radar, but different elevations, the final output value is a weighted average where the weight varies from 1 (at the center of the elevation) to 0.5 (at the boundary between the two elevations) with the weight given by δ_e:

$$\begin{aligned} \delta_e = exp(\alpha^3 ln(0.005)) \\ \alpha = |e - \theta_i|/(|\theta_{i\pm1} - \theta_i| \vee b_i) \end{aligned} \quad (5)$$

where α is the angular separation of the voxel from the center of the beam of an elevation scan (at elevation e) as a fraction either of the angular distance to the next higher or lower beam or the beamwidth. The $\vee$ is a maximum operation, b_i the beamwidth of the elevation scan and θ_i the elevation angle of its center. Finally, amongst multiple contributors belonging to different radars, the final value is chosen as a weighted average where the weighting function is:

$$\delta_r = e^{-r^2/2500} \quad (6)$$

[1]The term "resolution" is used in this paper in the sense in which it is commonly used in digital image processing, as the smallest detectable distance between features recorded digitally. In meteorology, this is often called the "grid spacing".

where r is the range from the radar in km, with the number 2500 chosen heuristically after some experimentation [15].

Because the individual radars in the network are not synchronized (they follow different volume coverage patterns and do not all finish an elevation scan at the same time), echoes at the same position are sampled by different radars at different times. Consequently, an intelligent agent updates itself with a motion vector, computed using the tracking method described in [21], [22] and moves to the location anticipated at the nominal time of the output grid. With the increase in speed of volume coverage patterns in the United States' weather radar network (now once every four minutes in precipitation mode), the difference between persistence and motion correction has become less significant. Therefore, the results shown in this paper were created without motion correction. The capability is still supported by the software, however, so that motion correction can be employed in countries where the radar scanning is not quite as fast.

B. Computational complexity and data rate

The networks of radars that cover the continental United States, Europe ex-Russia, or mainland China contain about 150 radars. The ability to mosaic continental-scale radar data in real-time has been an active area of research with constant leapfrogging between mosaicing algorithms and radar scanning strategies. Even as better mosaicing algorithms are developed and computer capabilities increase, making it possible to mosaic higher and higher resolution radar data in less and less time, radar scanning technology also improves, increasing the spatial and temporal resolution of the input radar data.

For estimating the computational complexity, it is assumed that the desired grid is $7000 \times 3500 \times 36$ in size and has to be created once every 5 minutes from the data provided by 150 radars each of which has a polar resolution of $0.25km \times 0.5^o$ and produce a volume scan once every 5 minutes. A grid that size is large enough to cover the Continental United States and Southern Canada at 1 km resolution and the vertical resolution is sufficient to sample the lower levels of the atmosphere at 0.25 km resolution and 16-20 km heights at a 1 km resolution. A 5-minute "volume coverage pattern" is realistic, though somewhat slower than the current capability of the US radar network. A 1 km resolution is, however, considerably coarser than the 0.25 km resolution of the polar data. The 36 vertical levels are the number needed to support both severe weather analysis (these require high altitudes) and precipitation estimation (these require fine resolution close to the surface). Therefore, the spatial and temporal resolution that are assumed are realistic, but are nevertheless a low bar to clear. This "basic" requirement can serve as a convenient benchmark, with higher resolutions represented as requiring certain multiples of the basic requirement.

Computation of Equations 1-6 scales by the number of voxels and number of radars, i.e. they need to be computed once at every pixel for every radar. Thus, those equations have to be computed 3e12 times an hour. On a Linux machine with a 2.3 GHz CPU, computing Equations 1-6 for a $7000 \times 3500 \times 36$ grid for every elevation scan of NEXRAD's Volume Coverage Pattern 12 took 27 minutes. Since that volume coverage pattern is completed in 4.5 minutes, it is clear that creating a mosaic of radars can not be completed in real-time using only a naive approach. Using lookup tables to cache the results of these computations, as suggested in [15], [20] cuts the computation time but at the expense of increasing disk I/O. The effective latency using such a lookup table approach is about 15 minutes, and can therefore not be used in real-time for any higher temporal resolution.

C. Mosaic methods

Because of the computational complexity and data rate, several mosaic methods have been devised to create continental-scale mosaics. These methods, except for the intelligent agent method of [15], have involved compromises on the resolution, accuracy or timeliness of the resulting mosaic.

1) A 2D Mosaic: The simplest compromise, to reduce the computational complexity and the data rate, is to first compute derived (2D) products on data from individual radars and then to mosaic these 2D products (See Figure 3a). This is the approach suggested by [23] and carried by the U.S. National Weather Service to create mosaics operationally at 2 km resolution. The savings in computational complexity and data rate are significant when using this approach of mosaicing only 2D products. The computation of Equations 1-6 need to be performed at only one level of the basic grid, thus leading to a 35-fold savings in computational complexity. Creating lower-resolution mosaics also leads to computational savings – a 2 km resolution mosaic represents a 4-fold reduction in the number of times that the mapping equations need to be computed. In addition, there are data rate savings that arise from creating 2D products only at the end of a volume scan, and not every time a new elevation scan is received (i.e., not in a "virtual volume" [24] sense). Because the typical volume scan has 5-10 elevation scans, a 5-10 fold savings in data rate can be realized by computing the single radar 2D products only at the end of a volume scan.

These savings in computational complexity and data rate come at the expense of accuracy, however. This is because the 2D products lack the height information that is critical for severe weather forecasting and for dealing with complex terrain in the case of precipitation estimation. It is more accurate to create 3D reflectiviy mosaics, and not just a 2D one. The problem, as shown in Figure 4, is that distance-weighted 2D mosaics underestimate quantities such as maximum reflectivity and VIL because they lose the full spectrum of height information that would be available from multiple radars in areas of overlap. In existing continental scale networks, there is significant overlapping coverage at mid-level heights (3-8 km) and it is important that computational and data rate constraints do not undermine the use of the extra height information available. Height information is critical in the computation of severe weather diagnosis products such as VIL, hail and azimuthal shear [10] and in the estimation of the vertical profile of reflectivity (VPR) that is an important component of precipitation estimation [25], [26]. Discarding the height

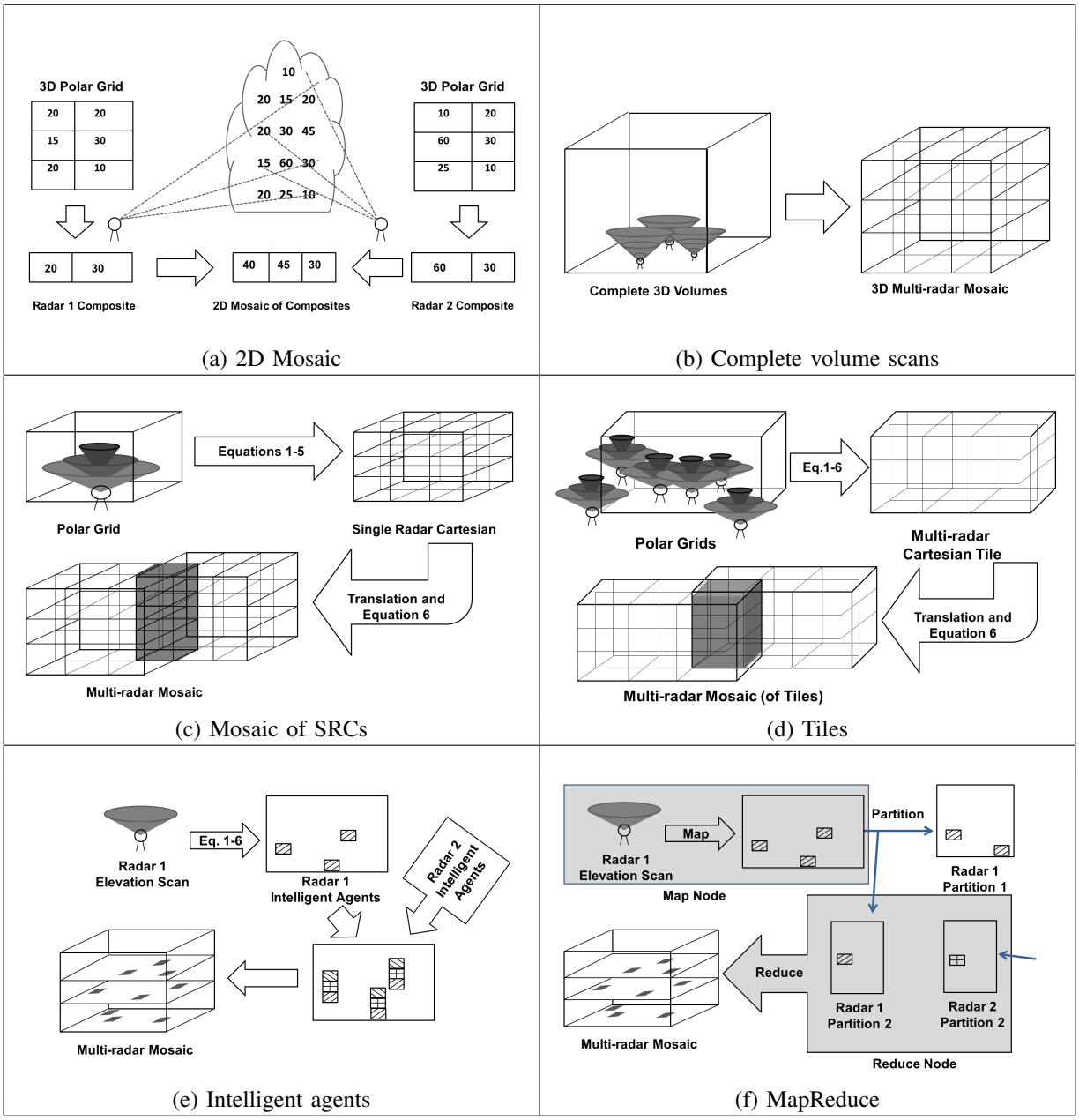


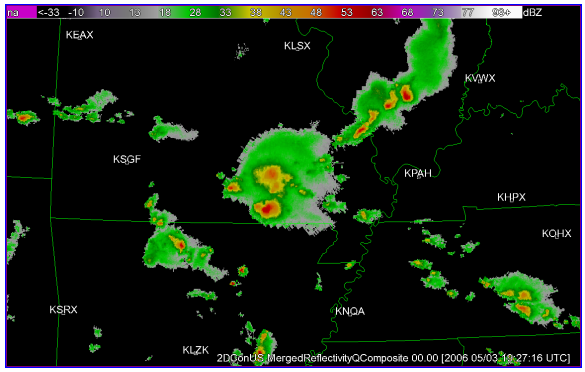
Fig. 3. Methods of creating multi-radar mosaics.

information also compromises one of the key motivations behind multi-radar mosaics (see Figure 2).

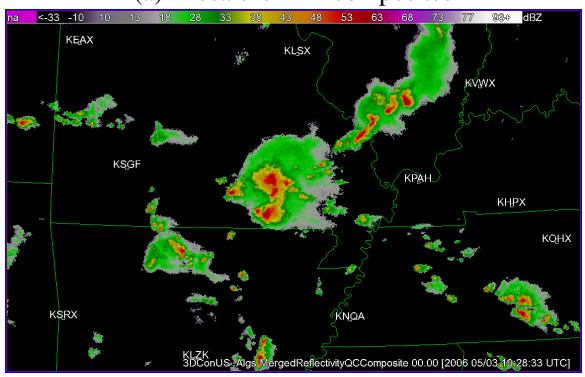
2) Complete volume scans: To retain the height information, [27] suggested compromising on the size of the domain and on the timeliness of the data. Users would interactively choose a domain (smaller than the continental scale of the network and capped at the processing power of the machines available) and 3D volume scans from the individual radars would be periodically mosaiced into a 3D grid of radar reflectivity (See Figure 3b). This is the approach currently followed in China [28] to create 1 km mosaics on subregions of the country. The computational complexity savings due to processing only a subregion are clear: they are N-fold if the subregion is $1/N$ the size of the domain of the radar network.

A different compromise on data rate is followed by the European Opera project where polar data from 130 radars are composited into 2D composites of maximum reflectivity precipitation rate and hourly precipitation accumulation but only at a 2 km resolution and only every 15 minutes [14]. The polar volume data from all the radars over a 15 minute period are directly composited by updating the lowest layer reflectivity and the maximum reflectivity at every grid point.

The timeliness compromise carried out in China and Europe is similar to that employed by the 2D mosaics of the US NWS [23] in that the data from the individual radars are not processed as each elevation tilt arrives, but only at the end of a volume scan. Because a volume scan in a mechanically-scanning radar takes 4-15 minutes, the data in the mosaic can be 4-15 minutes old. If the mosaic is produced every 5 minutes, this can lead to latencies as high as 9-20 minutes. The



(a) Mosaic of 2D composites



(b) Composite derived from 3D mosaic

Fig. 4. Because the 2D composites have lost height information, mosaics of 2D products underestimate thunderstorm severity. Note that in the mosaic of 2D composites, the storm in the center of the image features lower values than in the composite derived from a 3D mosaic. The storm in the center of the image is sampled by multiple radars (the radar locations are marked by four-letter acronyms) but is not near any of them.

average latency in the 3D grid is half the maximum latency – this is the mean time that elapses between data being available from the radar and that data being depicted in the 3D mosaic. Thus, even though processing volume scans only when they are complete leads to a data rate savings (a M-fold saving if there are M elevation scans in the average volume scan), these data rate savings come at a significant cost in terms of the timeliness of the data.

3) Single-radar Cartesian grids: In order to retain height information while retaining the desired spatial resolution and extent of the grid, [29] suggested first creating 3D Cartesian grids comprised of data from each individual radar. These single-radar Cartesian grids (SRCs) are then mosaiced into a 3D grid covering the entire domain as illustrated in Figure 3c.

The computational complexity of the SRC approach involves the computation of Equations 1-6 on each voxel of the single radar grid. Thus, the computational complexity is exactly the same as directly computing the mosaic from polar data. However, by performing the mosaic as a 2-step operation, it is possible to distribute the processing, so that the SRCs are created on a cluster of machines (perhaps the SRCs for 8-10 radars are created on one node) and the resulting SRCs are transmitted to a central node where they are mosaiced. The computational complexity of the second step is somewhat low, since the data simply need to be translated to the right position in the final grid and then linearly weighted based on the distance of the voxel in the SRC from the center of the SRC using Equation 6 (the center of the SRC is the radar location, so that this distance is the radar range). It was also pointed out in [20] that Equations 1-6 could be precomputed and the results stored for each voxel in each SRC. This lookup table could be stored on the nodes on which the SRCs were computed, thus trading computational overhead for disk I/O. This lookup table approach has been subsequently followed in all the mosaicing methods since developed (for e.g. [15] and [28]).

Distributed processing and caching contribute savings in computational complexity that make it possible to create continental scale mosaics in real-time. The SRC approach scales very well to the number of radars – one can simply add more machines on which to create the SRCs if one has more radars.

If only complete volume scans of data are used to create the SRCs, timeliness remains an issue. The latency of the multi-radar mosaic is the sum of the time taken to create the mosaic and the average age of the data within the SRCs. If the SRCs are computed from complete volume scans, the total latency is again on the order of several minutes. It is, however, possible to use the intelligent agent approach suggested by [15] and described below to create SRCs with the latest available data from the radar. If the SRCs are computed from virtual volumes, timeliness is less of an issue – if the SRCs are computed every 5 minutes, then the average age of the data within the SRC is 2.5 minutes.

There is, also, a new problem: that of accuracy. The SRC approach involves two stages of smoothing, one within the radar volume and one across data from multiple radars. Since the weighted average of four values differs from the average of two weighted-averages, there is information lost in the SRC process unless one keeps track of the weights involved. As long as this loss in accuracy or increase in memory is acceptable, a combination of the SRC approach and the intelligent agent approach is a reasonable compromise when creating continental scale mosaics. However, the MapReduce method introduced in this paper provides a better solution with more scalability, lower data rate and higher accuracy.

4) Tiles: While the SRC method scales well to the number of radars (since one can always add more single-radar processing nodes), it does not scale quite as well if the size or resolution of the domain is increased. The second stage machine that mosaics the SRCs is overloaded in terms of data rate because the SRCs are larger than the polar data (more voxels). Even though the computational complexity of translating the voxels in quite low, there are $7000 \times 3500 \times 36$ voxels to translate every 5 minutes if one is using the basic grid.

Taking the idea of distributed processing further, one can think of creating, not just single-radar Cartesian grids, but multi-radar Cartesian grids ("tiles") over subdomains and then mosaicing the 3D grids corresponding to the subdomains over the complete domain. The problem with creating multi-radar tiles and then mosaicing the tiles is that this is subject to diminishing returns. On the boundary of tiles will be radars that are outside the tile but whose coverage area includes

voxels within the tile (See Figure 3d). Because of this, the scalability of the tiling approach is problematic. Another problem with the tiling approach is that there is a further loss in accuracy, because there are now three stages of smoothing and two time delays.

5) Intelligent agents: [15] introduced an intelligent agent approach to mosaicing radar data whereby each range gate of the radar serves as the impetus to the creation of one or more intelligent agents (See Figure 3e). Each intelligent agent monitors the movement of the storm at the position that it is currently in, and finds a place in the resulting grid based on time difference. Then, at the next time instant, the range gate migrates to its new position in the grid. The agents remove themselves when they expect to have been superseded. When new storm motion estimates are available, the agent updates itself with the new motion vector. In Figure 3e, intelligent agents from different radars and elevation scans are shaded differently and it can be seen that there will be multiple intelligent agents for each voxel, with one intelligent agent for every radar elevation scan that affects that voxel. Periodically, the different agents for a given voxel in the 3D grid collaborate to come up with a single value following the weighting equations given in Equations 5-6.

The agents' coordinates in the earth-centric 3D mosaic grid are obtained from lookup tables that are precomputed using Equations 1-4 and the vertical and multi-radar weights (δ_e and δ_r) are also precomputed for every possible volume coverage pattern. Given the domain, radar, elevation scan and volume coverage pattern of the elevation scan, the lookup table that provides the coordinates and weights of all the agents corresponding to that elevation scan can be retrieved. Only intelligent agents that correspond to data above the signal-to-noise threshold are retained, thus limiting the memory usage of the application.

Because the intelligent agents are created as new data are received from any radar, the 3D grid of reflectivity always contains the latest available data from every radar in the domain, i.e., the intelligent agent approach uses "virtual volumes" [24] and does not wait for the end of a volume scan. Instead, data are processed elevation scan by elevation scan. Not waiting for the end of a volume scan has another advantage in the intelligent agent approach – because the 3D grid of reflectivity is continuously updated over the entire domain, the computational load is spread evenly through time.

The intelligent agent approach by spreading the computational load evenly over time and precomputing and organizing the results of expensive operations makes it possible to mosaic weather radar reflectivity data in real-time on a continental scale. The intelligent agent approach sufficed to handle data from the United States' network of weather radar circa 2005 in real-time to create a $7000 \times 3500 \times 36$ grid in real-time every 5 minutes.

D. Increased requirement

Since 2005, however, the United States' weather radar network has been upgraded in several significant ways that have increased the spatial and temporal resolution of the data. Even though computational capability has increased over that time (faster computers, better network bandwidth), it has become increasingly impossible to mosaic radar reflectivity data in real-time using just the intelligent agent approach.

The fastest volume coverage pattern in 2005 consisted of 14 elevation angles in 5 minutes. Frequently, radars operated in "clear-air mode", which consisted of 5 elevation angles every 10 minutes. Since then, a volume coverage pattern was introduced that combines the benefits of the old "precipitation mode" and clear-air mode volume coverage patterns. The new volume coverage pattern scans 14 elevation angles in 4.5 minutes [30], thus retaining the speed and height information of the old precipitation mode scans. In addition, by carrying out multiple scans at the same elevation angle at different pulse repetition frequencies, it retains the ability of the clear-air scans to sense fine lines and boundaries. Because of this, many of the radars are now routinely left at this new volume coverage pattern. A change from an average of 60 seconds between elevation scans to 19 seconds represents a more than 3-fold increase in the data rate.

A second enhancement to the US radar network has been an increase in spatial resolution. Whereas radar reflectivity data circa 2005 used to have a spatial resolution of 1 km x 1 degree, it now has a spatial resolution of 0.25 km $\times$ 0.5 degree at the lower (most informative) tilts, representing a 8-fold increase in data rate.

It should be clear that this 24-fold increase in data rate refers to the size of the raw input data stream that needs to be processed. The true information content of the stream has not increased that dramatically because radar range gate values at adjacent locations and adjacent times tend to be highly correlated. Also, because weather data tends to be sparse in the domain, the intelligent agent approach of [15] realizes efficiency gains by creating intelligent agents only for values that are above the signal-to-noise threshold.

The actual increase in information content between 2004 and 2012 can be gauged by examining the actual size of the input radar data. In 2004, after compression, the typical quantity of radar data per day was 17.5 GB.[2] In contrast, the typical daily size of the data in 2012 was 78.5 GB. In other words, in terms of information content, the data rate has increased by 4.5-fold. A mosaic algorithm that processed only truly new information would see a 4.5-fold increase whereas a naive mosaic algorithm that did not take advantage of any redundancy in the data stream would see a 24-fold increase. Real-world mosaic implementations such as [10] would see between a 4.5 and 24-fold increase in the data rate.

With the increased spatial and temporal resolution of the input polar data, the basic grid resolution of 1 km every 5 minutes no longer suffices. Instead, it represents a coarsening of the data. It would be preferable to create a grid at 0.25 km resolution every 2 minutes to better match the input resolution.

[2]This statisic and the corresponding one for 2012 was kindly compiled by Steve Ansari of the National Climate Data Center and provided to us via personal communication. The size of compressed data is reported because the size of compressed data is related to the Shannon entropy [31] or information content of the data and the typical quantity or median is more robust to outliers than a simple arithmetic mean.

Because the computational complexity of Equations 1-4 scale up by the number of voxels, a 0.25 km $\times$ 0.25 km grid represents a 16-fold increase in computational complexity over a 1 km $\times$ 1 km grid. The increased temporal resolution requirement from every 5 minutes to every 2 minutes represents a 2.5-fold increase in computational complexity.

Thus, not only has the input data rate increased 5 to 24-fold, the increase in resolution has neccessitated a corresponding increase in the output resolution that represents a 40-fold increase in computational complexity.

Consequently, even a two-stage approach of using intelligent agents to create SRCs and then mosaicing the SRCs together no longer scales to the US weather radar network. The second stage of the process, involving the creation of mosaics from the SRCs, can no longer be carried out on a single node. A better, scalable approach is needed.

II. MapReduce

MapReduce [32] is a programming model and distributed application framework that was introduced as a massively scalable approach to indexing the web. The Google distributed file system [33] and MapReduce API have since been approximated in an open-source framework called Hadoop[3] and in this form, the idea has found wide applicability in a variety of domains [34]. MapReduce is well-suited to embarrasingly parallel problems, i.e. problems that can be broken up into small chunks that can be computed in parallel with no shared data requirements. This is true, in particular, of applications whose aim is to compute distributive statistics over large quantities of data.

MapReduce and MapReduce frameworks such as Hadoop were designed for large scale data analysis, not for the real-time processing of streaming data. Consequently, we did not employ Hadoop's distributed file system or its application programming interface. Instead, we used the Warning Decision Support System (WDSS-II; [10]) as the application programming interface and Unidata's Local Data Manager (LDM; [4]) software to explicitly transfer data rather than implement a distributed file system. The data were stored and transferred as direct copies of in-memory data structures. This solution worked well because all our machines had the same 64-bit architecture and all our programs were written in the same programming language (C++), thus avoiding the need to perform expensive serialization or parsing of the transmitted data.

Despite the name, the MapReduce algorithm consists of four steps – Map, Combine, Partition and Reduce (See Figure 5) – and works on key-value pairs. Formally, the MapReduce operations are (the k's are keys and v's are values):

1) Map is an operation mapping $(k1, v1)$ to $(k2, v2)$
2) Combine is an operation that is carried out on the Map nodes to combine all the $v2$ for a particular $k2$ to yield a single entry $(k2, v2)$.

[3]http://hadoop.apache.org/

[4]http://www.unidata.ucar.edu/software/ldm/

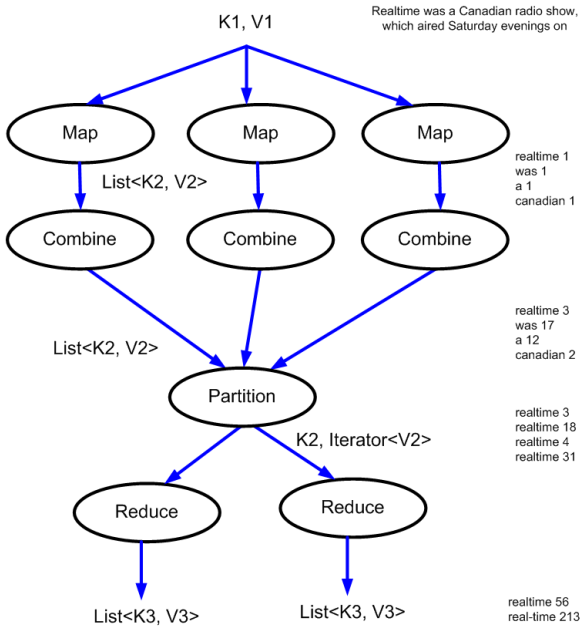


Fig. 5. There are four steps in solving an embarassingly parallel data analysis problem using MapReduce. The problem is split among multiple Map-Combine nodes that process different chunks of the input data in parallel. The results are then partitioned to Reduce nodes that work in parallel. On the right-hand side of the figure, the data and results of a MapReduce algorithm to count the number of occurences of a word in a corpus of documents is shown.

3) Partition is an operation by which the keys are shuffled and divided amongst the Reduce nodes. Each of the Reduce nodes processes a subset of $k2$'s.
4) Reduce is an operation reducing $(k2, list < v2 >)$ to $(k3, v3)$. Typically, Reduce is an online-processing algorithm that iterates through the list of $v2$'s.

For example, consider the problem of trawling through all of Google's archive of web documents and counting the number of times that a particular word is used. This example is illustrated on the right-hand side of Figure 5. [5] The MapReduce algorithm to compute the word count is as follows:

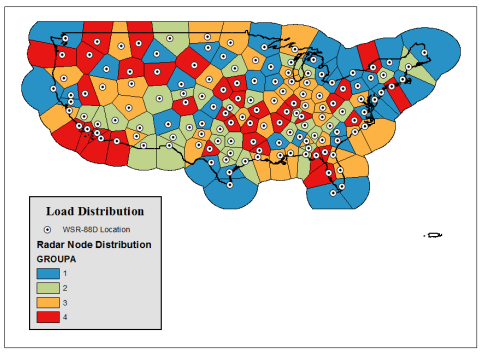
1) Split input data (the archive of documents) into chunks and process each chunk on a Map node. The Map operation's input is each line of text. The key here is ignored whereas the value is the line itself.
2) Map each word in the chunk to the number 1. In other words $k2$ is the word (such as "realtime") while $v2$ is always 1.
3) On that node, Combine the Map results so that the word ($k2$) is mapped to the number of times ($v2$) it has appeared in the chunk.
4) Transfer the Map results to a node that sorts the results and shuffles them by the key (in this case the word).
5) Partition the Map results to a number of Reduce nodes. For example, this could be done by sending all words that start with the letter "a" to a node.

[5]The size of the data that have been indexed can be gauged by the word counts returned. In November 2012, the authors were able to query Google and obtain the information that the word "realtime" was used 88 million times whereas the word "real-time" appeared 3830 million times.

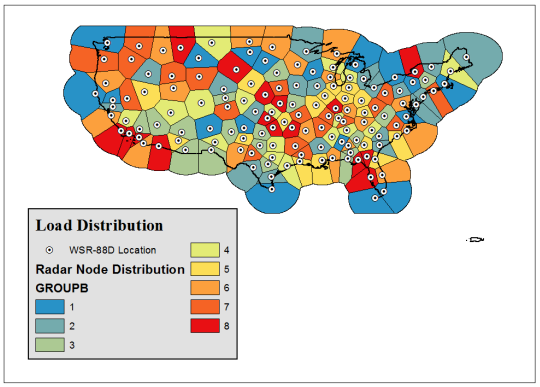
6) Reduce the results for a particular key by combining the results for that key (in this case, by adding up the word counts obtained from the various input chunks). Here $k3$ is the word itself (e.g. "realtime") while $v3$ is the total count of that word in the archive (e.g. 88 million).

Although all the keys here (except $k1$) are simply the word itself and the values are all integers representing the word-count, the keys and values can, in general, be different at each stage of the process.

The massive scalability of MapReduce can be explained with reference to the above process. The larger the trove of documents that need to be trawled, the more the number of Map nodes needed. The input chunks are all independent of each other (this is an embarrasingly parallel problem) and can be processed separately. The data required for each Map task is independent of other Map tasks (no shared data) and can be stored locally (minimizing disk I/O). The mapped, combined results can be partitioned and sent to a number of nodes each of which carries out a Reduce operation. The transmitted data are smaller than the data input to the Map tasks, thus reducing network overhead. As long as the algorithm to carry out the partition is fixed and predictable ("all words that start with the letter c go to machine 42"), there is no need for any centralized coordination. In practice, MapReduce frameworks use a central "name node" to coordinate tasks because of the need to provide failover support if one of the machines or disks fails and its job needs to be repeated. The Reduce tasks are also all independent of each other (embarrasingly parallel again) and can be carried out without any shared data. The Reduce tasks do not scale by the size of the input documents, but only the number of unique words in them. Thus, the number of nodes assigned to the Map and Reduce tasks are different, and can be scaled accordingly. In this case, the number of Reduce nodes can be far fewer than the number of Map nodes. The results of all the Reduce tasks are independent because they correspond to the counts of different words.



(a) Four nodes



(b) Eight nodes

Fig. 6. The effect of dividing Map processing among four (eight) nodes based on alphabetically ordering radar names is depicted by associating with a geographic location a color that represents the node most affected by weather at that location. The dots represent radar locations. Unlike a geographic grouping of radars, alphabetization provides a more equal distribution of effort when there is weather affecting only one part of the country.

III. Mosaic Radar Using MapReduce

It is possible to organize the computation of Equations 1-6 such that it is embarassingly parallel, uses only data local to the node and is non-redundant. Formulated in this manner, the entire mosaicing algorithm is amenable to being carried out using MapReduce, thus achieving massive scalability. It is then possible to create mosaics of arbitrarily high spatial and temporal resolution from any number of radars given enough computational nodes. This MapReduce approach, illustrated in Figure 3f, involves the steps of Map, Partition and Reduce and takes single radar polar data and creates multi-radar Cartesian 3D grids.

A. Map

The polar data that arrive from each radar are processed by separate Map nodes. At the Map nodes, the raw data are quality-controlled using the method of [35], [36] and the quality-controlled polar data are mapped onto a list of $(x, y, z, v, \delta_e, \delta_r)$. Here (x, y, z), which are the coordinates in the output 3D grid, form the key k_2 and (v, δ_e, δ_r) form the value v_2 of the MapReduce algorithm. δ_e and δ_r are the weights from Equations 5 and 6 while v is the scalar value that is being mosaiced (for example, radar reflectivity in dBZ). Data from a radar are mapped as it arrives, elevation scan by elevation scan, thus spreading the load over time and maintaining a virtual volume in the output grid. The only k_2, v_2's that are stored correspond to data values that meet the Quality Control (QC) criterion. Because radar data are sparse, this constraint achieves a large degree of compression. Because older data need to be removed from a location (x, y, z) if the newer scan does not show valid data at that point, the domain affected by each elevation scan is also stored as part of the mapping output.

Data from different radars are divided among the Map nodes based on an alphabetical ordering of the radar name (a unique four-letter call symbol assigned to every radar in the US weather network). An alphabetical sort is used primarily to make it easy to find which node a particular radar's data are being mapped at, but it also has the effect of load equalization because weather systems are geographically correlated. For

example, if there is a squall line in the central United States, all the Map nodes will be approximately equally occupied (See Figure 6). On the other hand, had the radars been geographically distributed among Map nodes, a few of the nodes would have become overloaded while the others would be unaffected. While it is possible to use algorithms such as [37] to devise an optimal division of radars given the present geometry (or even a dynamically optimal division of radars that changes with weather conditions), we did not do so.

A single Map process can quite easily handle all the radar data that arrives on a node, but it is better to have multiple Map processes running, place data from different radars onto different queues and have the data mapped into the list of $(x, y, z, v, \delta_e, \delta_r)$ in different processes. Because the mapping process is embarassingly parallel by radar, running multiple mapping processes provides parallel execution, especially on machines with multiple CPUs.

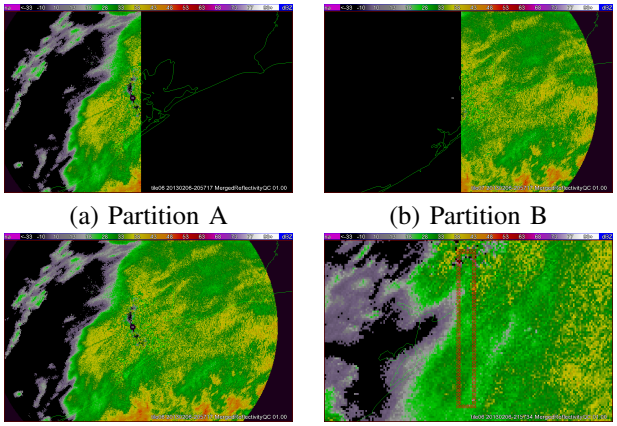
The mapping is carried out with the help of precomputing Equations 1-6, thus trading disk I/O (the precomputed lookup table needs to be read in every time a radar elevation is received) for CPU. Because a Map process corresponds to only one radar, and because radars typically maintain a constant volume coverage pattern over long periods, lookup tables that are read from disk can be maintained in memory, thus avoiding the disk I/O overhead except when the volume coverage pattern changes.

B. Combine

Even though multiple (v, δ_e, δ_r) might be produced for a single (x, y, z) on a Map node (because multiple elevation scans and multiple radars are processed on it), we found little benefit in terms of network bandwidth of combining these values on the Map nodes. Thus, there is no Combine operation. As soon as elevation data from any radar is mapped, the mapped output is sent to a Reduce node.

C. Partition

The Partition process is carried out implicitly on the Map node to avoid the need for a central clearing house. Avoiding centralization has the benefit, besides avoiding a single point of failure that a name node would represent, of increasing the speed at which data are transferred. The latter point is important because the radar data are streaming, and because the mosaicing needs to be carried out in real-time. The mapped data are partitioned into the number of Reduce nodes and each Reduce node computes the 3D mosaic grid for a single tile. This tile is different from the tiling discussed in Section I-C4 because the input for the tile comes from radars throughout the domain. In other words, the output is partitioned, but the input is not. In the tiling of Section I-C4, both the input and output were partitioned, leading to the problem of diminishing returns as data from individual radars had to be repeated on multiple tiles. Also, with the tiling approach, interpolation has to be carried out at tile boundaries if radars from outside the tile are not included. With the partitioning approach, each partition is independent and non-redundant. No interpolation needs to be carried out at partition boundaries (See Figure 7) and data from any radar are processed only once.



(a) Partition A (b) Partition B

(c) Stitched (d) Latency artifact

Fig. 7. (a,b) Data from a single radar is mapped only once, but sent to two partitions which are completely independent (i.e., have no overlap). (c) Reduced data from two partitions may be laid side-by-side ("stitched") in order to create a mosaics of of the entire area. (d) Under rare circumstances, due to network latencies, data into one of the partitions may be slightly delayed leading to artifacts that make the partition boundary somewhat more apparent.

The partition process consists of translating the computed (x, y, z) location to the location within the partitioned mosaic. As long as the partitioning is logical ("divide the domain into 2×4 sections"), this is a cheap arithmetic operation. The computed $(x, y, z, v, \delta_e, \delta_r)$ is thus partitioned into several lists of $(x_p, y_p, z_p, v, \delta_e, \delta_r)$ where (x_p, y_p, z_p) refer to the voxel location in the partitioned output. There is one such list for each partition and one partition per Reduce node.

D. Reduce

The data are now transferred from each Map node to several Reduce nodes using LDM. On the Reduce node, a mosaic process reads in the $(x_p, y_p, z_p, v, \delta_e, \delta_r)$ data as they arrive. Periodically, a grid consisting of (x_p, y_p, z_p, v_p) is created where v_p consists of the weighted average at a voxel of the v's from all the radar elevations that contribute towards it (See Figure 3f). In terms of the formal MapReduce process, k_3 is (x_p, y_p, z_p) and v_3 is v_p. Because both (δ_e, δ_r) are stored for each voxel, the choice of the combination mechanism can be deferred to the Reduce stage. By having multiple Reduce nodes, it is also possible to produce mosaics of the same product with different merging strategies. For example, choosing the v with the maximum δ_r will result in assigning a voxel with the value from the radar closest to it.

One of the problems with increasing the resolution (and hence the size) of the output grids is that simply writing the 3D grid out periodically takes longer and longer periods of time. Writing a $7000 \times 3500 \times 36$ grid in NetCDF [38] takes about 3 minutes. Increasing the spatial resolution from 1 km to 0.5 km makes the grid 4 times larger and the write time correspondingly longer. Taking 12 minutes to write a grid that is computed every 5 minutes means, of course, that the system would no longer be able to operate in real-time. There are,

however, some favorable aspects of this problem that can be taken advantage of:

1) The products that most users visualize tend to be 2D products and while these 2D products need to be created from the 3D grid, users do not need the 3D grid itself.
2) Users who wish to interrogate and visualize 3D data are commonly analyzing individual storm cells and therefore, it is enough to provide these users the output of a single Reduce node (at most four if the storm cell happens to be inconveniently located at the intersection of four partitions).
3) Radar-derived 2D products (composites, VIL, hail diagnosis, etc.) are all created through vertical integration. Therefore, creating these 2D products within the Reduce node and then stitching the output together to form a continental-scale grid is equivalent to first computing the continental-scale 3D grid and then computing the 2D products. It should be noted that this is different from the 2D mosaics discussed in Section I-C1 because these 2D products are created from a multi-radar 3D mosaic that has been partitioned, not mosaiced from 2D single-radar products.
4) The mosaic problem is also embarassingly parallel in the temporal dimension making it possible to stagger the output of the Reduce nodes.

E. Massive scalability

It is worth stressing the last point, because it provides the ability to arbitrarily scale the MapReduce algorithm to create mosaics at any desired spatial and temporal resolution. It is possible to run multiple sets of Reduce nodes that are staggered in time. For example, one set of Reduce nodes could be configured to write its output at even minutes (12.00, 12.02, etc.) while another set of Reduce nodes could be configured to write its output at odd minutes (12.01, 12.03, etc.). The output of the Map nodes would be sent (as elevations from the radars are mapped) to both these sets of Reduce nodes. Considering the two sets of Reduce nodes as a single entity, an output temporal resolution of 1 minute has been achieved even if a Reduce node can operate only at a 2 minute frequency in order to have time to write out its data. Time stagger only addresses the issue of output frequency (temporal resolution), not latency. The speed of disk I/O still plays a part in the time elapsed between the mosaic values being finalized and the 3D grid being available on disk. However, because our interest in the 3D grid is to compute 2D products and not to visualize it by itself, the latency is now reduced to the time taken to write out the output grid at each Reduce node. Because these 3D grids are smaller, they take less time to write than the 3D grid corresponding to the full domain. The size of a Reduce node grid is related to the number of partitions and can be made arbitrarily small by increasing the number of Reduce nodes. Thus, by using time stagger and MapReduce, it is possible to create mosaics at any desired spatial and temporal resolution.

The MapReduce approach to mosaicing radar data described in this paper is massively scalable. If the number of radars increases, one simply needs to add more Map nodes. The Map nodes are all independent of each other and can be scaled up without any penalties due to interprocess communication. Similarly, if the size or resolution of the output grid increases, one simply needs to add more Reduce nodes. It is also possible to time stagger the Reduce nodes so as to increase the temporal resolution and to increase the size of the partitions so as to be able to process the data within any desired latency tolerance.

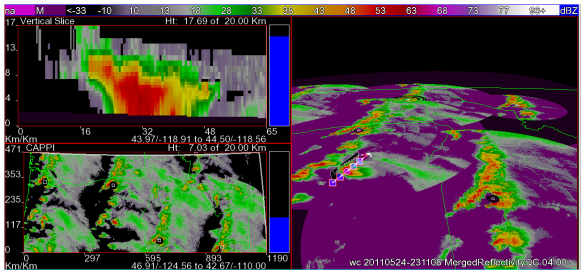


Fig. 8. The number of nodes required to implement a 3D radar mosaic of data covering the United States was determined using the worst case scenario shown above. The panels, clockwise from top-left, are a vertical slice, data at 4 km above mean sea level and 7 km above mean sea level. This overactive scenario was simulated by feeding single-radar data from an active weather day to all 154 radars that cover the domain.

There is one practical caveat, however, with increasing the number of partitions. Even though the Map nodes process radar data as it arrives and partition them to the Reduce nodes, the mapped data do not all arrive at the Reduce node at the same time. This is because of the vagaries of network optimization – smaller packets are typically delivered before larger ones and therefore, partitions that are only minimally affected by a radar will receive updates from that radar before a partition that is heavily influenced by it. Even though the Reduce nodes are synchronized as far as their output time is concerned (because of time staggering), the Reduce processes that compute the 3D output grids do not necessarily process the radar data that arrives just as the 3D grid is being finalized. Therefore, when derived 2D products are stitched, partition boundaries may be apparent (See Figure 7d). The more the number of partitions, the greater this effect will appear. It should be noted that if the mosaics are produced every minute or every 2 minutes, the temporal lag between data on the two sides of a partition boundary is well below the 4.5-minute update interval of the single-radar data. It should also be noted that this is a very rare problem since it occurs only if the writing of a reduced grid is triggered in the fraction of a second delay between the data arrival on the two machines. The image in Figure 7d was created by artificially delaying the data on one of the reduce nodes so as to increase the likelihood of this possibility. Because this possibility exists, however remote, it is desirable to have as few partitions as possible.

F. Computational requirement

Because the approach is massively scalable, we present in Table I the minimum number of Map nodes, Reduce nodes and time staggers that are required to achieve a given spatial and temporal resolution over a domain that includes the Continental United States and Southern Canada. The nodes used for these tests represent a 2.3GHz CPU with 12 GB

TABLE I
NUMBER OF MAP AND REDUCE NODES REQUIRED TO IMPLEMENT A REAL-TIME 3D MOSAIC OVER A DOMAIN THAT COVERS THE CONTINENTAL UNITED STATES. THE TOTAL NUMBER OF REDUCE NODES RESULT FROM SPATIAL PARTITIONING AND TIME STAGGERS THAT EXECUTE IN PARALLEL.

Resolution	Map	Reduce	Partition	Stagger
1 km (5 min)	6	4	2x2	1
1 km (2 min)	6	8	2x4	1
1 km (1 min)	6	16	2x4	2
0.5 km (5 min)	22	10	2x5	1
0.5 km (2 min)	22	40	4x5	2
0.5 km (1 min)	22	80	4x5	4
0.25 km (5 min)	77	10	2x5	1
0.25 km (2 min)	77	40	4x5	2
0.25 km (1 min)	77	80	4x5	4

of RAM, so that a machine with two dual-core CPUs would consist of four nodes. The number of nodes required were determined by taking an episode of a very active weather day (data from the Oklahoma City radar, KTLX between 23:00 and 23:59 UTC on May 24, 2011 when the radar was in Volume Coverage Pattern 12) and supplying that data as the input to all 154 radars (See Figure 8). Therefore, this table represents a worst-case scenario.

The number of Map nodes required was estimated by adding more and more radars to a node until the Map node was unable to process the 60 minutes of data in under an hour. Because the Map nodes process radar data as they arrive, the number of Map nodes is not affected by the temporal resolution of the output grid. The larger the desired spatial resolution of the output grid, the more Map nodes are required because a 0.5 km resolution grid has four times as many voxels as a 1 km grid (only the horizontal resolution is doubled; the vertical resolution remains the same).

The number of Reduce nodes required was estimated by computing the Mapper output for all 154 radars and increasing the number of partitions until the Reduce operation for an hour of Mapper output for a single partition was able to complete on a single node in under an hour. The split of the partitions is also reported in the table. For simplicity of implementation, the partitions were equally sized, although one might obtain fewer partitions by using larger geographic areas in the Mountain West where there are fewer radars. As with the Map nodes, a larger spatial resolution in the output grid translates to more voxels and, therefore, to more Reduce nodes. Unlike with Map nodes, however, the number of Reduce nodes are also affected by the desired temporal resolution. The weighted average of Equation 6 needs to be computed more frequently with increasing temporal resolution. Consequently, a higher temporal resolution neccessitates more Reduce nodes. However, the scaling is not linear because the input-output needs of the Reduce node processing is determined mostly by the speed of the radars' VCPs, and not by the desired output temporal frequency.

The time required to write the 1 km resolution output grid is on the order of one minute. Therefore, time staggering is needed to achieve 1-minute or higher output frequencies at that resolution. The 0.5 km resolution grid would be four times larger, but the use of more partitions causes the resulting grids to be not quite as large. The time to write the 0.5 km resolution output grids is on the order of 2 minutes. Consequently, time staggering is required to achieve 2-minute or higher output frequencies at the 0.5 km resolution. Since time staggering involves piping the output of the Mappers to two independent sets of Reduce nodes, time staggering doubles the number of Reduce nodes.

None of the mosaic configurations in Table I are possible on present-day hardware using any other approach, unless one is willing to compromise on accuracy, timeliness or resolution. When using the intelligent agent method of [15] without MapReduce, the mean latency for a 1 km grid written out every 5 minutes is about 15 minutes. This illustrates why a 3D mosaic is not possible in real-time without the massive scalability made possible using the MapReduce approach of this paper. With the MapReduce approach, it is possible (given enough machines) to compute a mosaic of any resolution using any number of radars. With continuing improvements in computational infrastructure in terms both of I/O and of CPU speeds, the number of machines required to implement any particular configuration can be expected to reduce in the future.

G. Summary

Real-time, continental-scale 3D mosaics of radar data are required for applications ranging from precipitation estimation, hail diagnosis and tornado warnings to public awareness. In this paper, a MapReduce approach to computing radar mosaics on a distributed cluster of compute nodes is presented. Single radar data are mapped elevation scan by elevatation scan into pixel locations, values and weights in a 3D mosaic grid by means of precomputed equations. Partitioning is carried implicitly on the Map node and data from a single radar are sent to different Reduce nodes, where each Reduce node represents a partition. On the Reduce node, the set of locations, values and weights are periodically reduced into the 3D mosaic grid corresponding to that partition. Derived 2D products are created from the 3D mosaic and then stitched over the full domain.

The MapReduce approach is massively scalable, and is able to create high-resolution 3D radar mosaics over the Continental United States in real-time. There are advantages to the MapReduce approach beyond its massive scalability – it is also the most efficient way of parallelizing the creation of a 3D radar mosaic. In other words, if a domain is so large that the mosaic can not be created on a single machine, the MapReduce approach requires less hardware than either the SRC or intelligent agent approaches. Furthermore, the partitioning approach that MapReduce takes is the most effective way of splitting the output into geographical regions. The partitions are independent of each other and radar data processing is non-redundant. No interpolation needs to be carried out across partition boundaries and data from a radar are processed only once. The MapReduce approach, therefore, permits for the creation of radar mosaics without compromises on the quality, timeliness or resolution of the mosaic.

ACKNOWLEDGMENTS

Funding for the authors was provided by NOAA/Office of Oceanic and Atmospheric Research under NOAA-OU Cooperative Agreement NA11OAR4320072, U.S. Department of Commerce. The technique described in this paper has been implemented as part of the Warning Decision Support System – Integrated Information (WDSS-II; [10]) as the program w2merger. It is available for free download at http://www.wdssii.org/.

We thank Steve Ansari of NOAA/National Climate Data Center for compiling statistics on the median size of radar data in 2004 and 2012.

REFERENCES

[1] U. Germann and I. Zawadzki, "Scale-dependence of the predictability of precipitation from continental radar images.," *Mon. Wea. Rev.*, vol. 130, pp. 2859–2973, 2002.

[2] R. Mosier, C. Schumacher, R. Orville, and L. Carey, "Radar nowcasting of cloud-to-ground lightning over Houston, Texas," *Wea. Forecasting*, vol. 26, pp. 199–212, 2011.

[3] D. Dowell, L. Wicker, and C. Snyder, "Ensemble Kalman filter assimilation of radar observations of the 8 May 2003 Oklahoma City supercell: Influences of reflectivity observations on storm-scale analyses," *Wea. Forecasting*, vol. 23, pp. 373–391, 2008.

[4] Q. Zhao, J. Cook, Q. Xu, and P. Harasti, "Improving short-term storm predictions by assimilating both radar radial-wind and reflectivity observations," *Wea. Forecasting*, vol. 23, pp. 373–391, 2008.

[5] J. Newman, V. Lakshmanan, P. Heinselman, M. Richman, and T. Smith, "Range-correcting azimuthal shear in doppler radar data," *Wea. Forecasting*, vol. 28, pp. 194–211, 2013.

[6] V. Lakshmanan, K. Hondl, and R. Rabin, "An efficient, general-purpose technique for identifying storm cells in geospatial images," *J. Atmos. Oceanic Technol.*, vol. 26, no. 3, pp. 523–37, 2009.

[7] E. Anagnostou and W. Krajewski, "Real-time rainfall estimation. part i: Algorithm formulation," *J. Atmos. Oceanic Technol.*, vol. 16, pp. 189–197, 1999.

[8] Y. Zhang, J. Smith, A. Ntelekos, M. Baeck, W. Krajewski, and F. Moshary, "Structure and evolution of precipitation along a cold front in the Northeastern United States," *Bull. Amer. Meteor. Soc.*, vol. 88, pp. 1899–1911, 2007.

[9] V. Vasiloff, K. Howard, R. Rabin, H. Brooks, D. Seo, J. Zhang, D. Kitziller, M. Mullusky, W. Krajewski, E. Brandes, B. Brown, D. Berkowitz, J. McGinley, and R. Kuligowski, "Improving QPE and very short term QPF: An initiative for a community-wide integrated approach," *Bull. Amer. Meteor. Soc.*, vol. 88, pp. 1899–1911, 2007.

[10] V. Lakshmanan, T. Smith, G. J. Stumpf, and K. Hondl, "The warning decision support system – integrated information," *Wea. Forecasting*, vol. 22, no. 3, pp. 596–612, 2007.

[11] G. J. Stumpf, T. M. Smith, and J. Hocker, "New hail diagnostic parameters derived by integrating multiple radars and multiple sensors," in *22nd Conf. on Severe Local Storms*, (Hyannis, MA), p. P7.8, Amer. Meteor. Soc., 2004.

[12] D. R. Greene and R. A. Clark, "Vertically integrated liquid water – A new analysis tool," *Mon. Wea. Rev.*, vol. 100, pp. 548–552, 1972.

[13] M. Miller, V. Lakshmanan, and T. Smith, "An automated method for depicting mesocyclone paths and intensities," *Wea. Forecasting*, vol. 28, pp. 570–585, 2013.

[14] A. Huuskonen, L. Delobbe, and B. Urban, "EUMETNET OPERA: Achievements of OPERA-3 and challenges ahead," in *Preprints, 7th European Conf. on Radar in Meteorology and Hydrology*, (Toulouse), Meteo France, June 2012.

[15] V. Lakshmanan, T. Smith, K. Hondl, G. J. Stumpf, and A. Witt, "A real-time, three dimensional, rapidly updating, heterogeneous radar merger technique for reflectivity, velocity and derived products," *Wea. Forecasting*, vol. 21, no. 5, pp. 802–823, 2006.

[16] R. Doviak and D. Zrnic, *Doppler Radar and Wea. Observations*. Academic Press, Inc., 2nd ed., 1993.

[17] W. Beyer, ed., *CRC Standard Math Tables*. CRC Press Inc., 18th ed., 1987.

[18] R. J. Trapp and C. A. Doswell, "Radar data objective analysis," *J. Atmos. Ocean. Tech.*, vol. 17, pp. 105–120, 2000.

[19] M. Askelson, J. Aubagnac, and J. Straka, "An adaptation of the Barnes filter applied to the objective analysis of radar data," *Mon. Wea. Rev.*, vol. 128, no. 9, pp. 3050–3082, 2000.

[20] J. Zhang, K. Howard, and J.J.Gourley, "Constructing three-dimensional multiple-radar reflectivity mosaics: Examples of convective storms and stratiform rain echoes," *J. Atmos. Ocean. Tech.*, vol. 22, pp. 30–42, Jan. 2005.

[21] V. Lakshmanan, R. Rabin, and V. DeBrunner, "Multiscale storm identification and forecast," *J. Atm. Res.*, vol. 67, pp. 367–380, July 2003.

[22] V. Lakshmanan and T. Smith, "Data mining storm attributes from spatial grids," *J. Ocea. and Atmos. Tech.*, vol. 26, no. 11, pp. 2353–2365, 2009.

[23] J. Charba and F. Liang, "Quality control of gridded national radar reflectivity data," in *21st Conf. on Wea. Analysis and Forecasting/17th Conf. on Numerical Wea. Prediction*, (Washington, DC), p. 6A.5, Aug. 2005. http://www.nws.noaa.gov/mdl/radar/mosaic_webpage.htm.

[24] R. Lynn and V. Lakshmanan, "Virtual radar volumes: Creation, algorithm access and visualization," in *21st Conf. on Severe Local Storms*, (San Antonio, TX), Amer. Meteor. Soc., 2002.

[25] B. Vignal, H. Andrieu, and J. Creutin, "Identification of vertical profiles of reflectivity from volume-scan radar data," *J. Appl. Meteor.*, vol. 38, pp. 1214–1228, 1999.

[26] M. Kitchen, R. Brown, and A. Davies, "Real-time correction of weather radar data for the effects of bright band, range and orographic growth in widespread precipitation," *Quart. J. Roy. Meteor. Soc.*, vol. 120, pp. 1231–1254, 1994.

[27] J. J. Levit, V. Lakshmanan, K. L. Manross, and R. Schneider, "Integration of the Warning Decision Support System - Integrated Information (WDSS-II) into the NOAA Storm Prediction Center," in *22nd Conf. on Severe Local Storms*, Amer. Meteor. Soc., Oct. 2004.

[28] H. Zhou, "China new generation weather radar three dimensional mosaic software system and application," in *Int'l Conf. on Computer Science and Service System (CSSS)*, (Nanjing), pp. 271–275, 2011.

[29] J. Zhang, K. Howard, C. Langston, S. Vasiloff, and B. Kaney, "National mosaic and multi-sensor QPE (NMQ) system: Description, results, and future plans," *Bull. Amer. Meteor. Soc.*, vol. 92, pp. 1321–1338, 2011.

[30] National Weather Service, "RDA/RPG build 9.0 training," tech. rep., Warning Decision Training Branch, 2007. http://www.wdtb.noaa.gov/buildTraining/Build9/pdfs/build9deploy.pdf.

[31] C. Shannon, "A mathematical theory of communication," *Bell System Tech. J.*, vol. 27, pp. 379–423, July 1948.

[32] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[33] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[34] S. Chen and S. Schlosser, "Map-reduce meets wider varieties of applications," Tech. Rep. IRP-TR-08-05, Intel Research, Pittsburg, 2008.

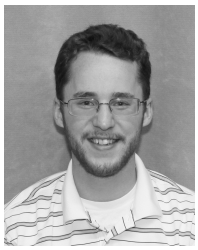
[35] V. Lakshmanan, A. Fritz, T. Smith, K. Hondl, and G. J. Stumpf, "An automated technique to quality control radar reflectivity data," *J. Applied Meteorology*, vol. 46, pp. 288–305, Mar 2007.

[36] V. Lakshmanan, J. Zhang, and K. Howard, "A technique to censor biological echoes in radar reflectivity data," *J. Applied Meteorology*, vol. 49, pp. 435–462, 3 2010.

[37] N. Robertson, D. Sanders, P. Seymour, and R. Thomas, "A new proof of the four colour theorem," *Electron. Res. Announc. Amer. Math. Soc.*, vol. 2, pp. 17–25, 1996.

[38] H. L. Jenter and R. P. Signell, "NetCDF: A freely-available software-solution to data-access problems for numerical modelers," in *Proceedings of the American Society of Civil Engineers Conf. on Estuarine and Coastal Modeling*, (Tampa, Florida), pp. 72–82, 1992.

Valliappa Lakshmanan is a research scientist at the Cooperative Institute of Mesoscale Meteorological Studies between the National Oceanic and Atmospheric Administration (NOAA) and the University of Oklahoma. His work focuses on machine intelligence and statistical approaches for the automated analysis of geospatial grids. He received degrees from the Indian Institute of Technology, Madras (B.Tech, 1993), The Ohio State University (M.S., 1995) and the University of Oklahoma (Ph.D., 2002).



Timothy W. Humphrey is a graduate student in the School of Meteorology at the University of Oklahoma. His research interests include probabilistic forecasting of severe weather phenomena from multi-radar mosaics. He received his B.S. degree in Atmospheric Science from the State University of New York (SUNY), Albany in 2012.