A Hybrid MPI/OpenMP Parallel Algorithm and Performance Analysis for an Ensemble Square Root Filter Designed for Multi-scale Observations

Yunheng Wang[1], Youngsun Jung[1], Timothy A. Supinie[1,2] and Ming Xue[1,2]

Center for Analysis and Prediction of Storms[1] and School of Meteorology[2]
University of Oklahoma, Norman Oklahoma 73072

August 2012

Submitted to J. Atmospheric and Oceanic Technology

Revised January 16, 2013

Corresponding author address:
Ming Xue
Center for Analysis and Prediction of Storms
University of Oklahoma,
120 David L. Boren Blvd, Norman OK 73072
mxue@ou.edu

Abstract

A hybrid parallel scheme for the ensemble square root filter (EnSRF) suitable for parallel assimilation of multi-scale observations including those from dense observational networks such as those of radar is developed based on the domain decomposition strategy. The scheme handles inter-node communication through message passing interface (MPI), and the communication within shared-memory nodes via Open Multi-Processing (OpenMP) threads; it also supports pure MPI and pure OpenMP modes. The parallel framework can accommodate high-volume remote-sensed radar (or satellite) observations as well as conventional observations that usually have larger covariance localization radii.

The performance of the parallel algorithm has been tested with simulated and real radar data. The parallel program shows good scalability in pure MPI and hybrid MPI/OpenMP modes, while pure OpenMP runs exhibit limited scalability on a symmetric shared-memory system. It is found that in MPI mode, better parallel performance is achieved with domain decomposition configurations in which the leading dimension of the state variable arrays is larger, because this configuration allows for more efficient memory access. Given a fixed amount of computing resources, the hybrid parallel mode is preferred to pure MPI mode on supercomputers with nodes containing shared-memory cores. The overall performance is also affected by factors such as the cache size, memory bandwidth, and the networking topology. Tests with a real data case with a large number of radars confirm that the parallel data assimilation can be done on a multi-core supercomputer with a significant speedup compared to the serial data assimilation algorithm.

1. Introduction

With significant advances in computing power in recent years, advanced data assimilation (DA) techniques, such as the ensemble Kalman filter (EnKF) (Evensen 1994; Evensen and Leeuwen 1996; Burgers et al. 1998; Houtekamer and Mitchell 1998; Anderson 2001; Bishop et al. 2001; Whitaker and Hamill 2002; Evensen 2003; Tippett et al. 2003) and four-dimensional variational (4DVAR) (e.g., Le Dimet and Talagrand 1986; Courtier and Talagrand 1987; Sun and Crook 1997; Gao et al. 1998; Wu et al. 2000; Caya et al. 2005), are becoming more popular in both operational and research communities. However, they both incur a high computational cost, one of the biggest constraints for their operational applications at very high resolutions. Between EnKF and 4DVAR, the EnKF method appears to be more attractive for convective scale numerical weather prediction (NWP), where nonlinear physical processes have critical roles. EnKF can also provide a natural set of initial conditions for ensemble forecasting. EnKF has been applied at scales ranging from global to convective and has produced encouraging results (e.g., Snyder and Zhang 2003; Dowell et al. 2004; Tong and Xue 2005, hereafter TX05; Xue et al. 2006; Jung et al. 2008; Buehner et al. 2010; Dowell et al. 2011; Hamill et al. 2011; Snook et al. 2011; Jung et al. 2012).

Among variants of EnKF, the ensemble square-root Kalman filter (EnSRF) of Whitaker and Hamill (2002) is widely used in convective-scale DA studies involving radar data. The EnSRF, as well as the similar ensemble adjustment Kalman filter (EAKF, Anderson 2003) and the classic perturbed-observation EnKF algorithm (Evensen 2003), is an observation-space-based algorithm in which observations are assimilated one after another. Because of the sequential nature of the EnSRF (and EAKF and classic EnKF), parallelization of the algorithm at the observation level is not straightforward. It is possible to parallelize at the state variable level,

i.e., to perform the updating of the state variables in parallel because each observation updates many state variables within the covariance localization radius of the EnSRF, and these operations are independent. Such parallelization can be easily achieved on shared-memory platforms via OpenMP directives, and is done with the Advanced Regional Prediction System (ARPS, Xue et al. 2003) EnSRF system (e.g., Xue et al. 2006; Jung et al. 2008). A processing element (PE) on a shared-memory or distributed-memory platform is an individual processor with single-core processors or a processor core on multi-core CPUs. Each PE generally supports only a single process or a single thread. The number of PEs available on shared-memory nodes (the term "processing unit," abbreviated PU, will be used to refer to a shared-memory node) usually limits the scale of shared-memory parallelization (SMP) and the number of state variables that can be updated simultaneously. Distributed-memory parallelization (DMP) via the Message Passing Interface (MPI) library would allow the use of much larger computers, which are essential for very-high-resolution DA and NWP over large domains (Xue et al. 2007).

Anderson and Collins (2007, hereafter AC07) proposed a modification to the standard EAKF algorithm that is also applicable to EnSRF. In their algorithm, multiple observation priors (background converted to observed quantities via observation operators) are first calculated in parallel, and the observation priors corresponding to as yet unused observations are updated by the filter together with the state vector, allowing easier parallelization at the state vector level (for a given observation, multiple elements in the state vector are updated in parallel). However, its state update procedure requires broadcasting the observation priors from one PU to the rest, and more importantly, the processing of observations is still serial. Because of this, the algorithm does not scale well when the number of PUs increases to the point where the cost of communication starts to dominate or when the ratio of the number of observations to that of state

variables is large. Other parallel approaches have also been proposed by Keppenne and Rienecker (2002) and Zhang et al. (2005). While both methods utilize domain decomposition, they differ in whether communication among PUs is allowed. Because there is no cross-PU communication in the algorithm of Zhang et al. (2005), the analysis near the PU boundaries is not the same as that of scalar implementation, which is a potentially serious drawback of their algorithm. Keppenne and Rienecker (2002), on the other hand, allow observations in other PUs to update the states in the current PU, but their communication cost is potentially very high because message passing is executed many times to properly exchange information among PUs.

In this paper, we develop a new parallelization algorithm for EnSRF (also suitable for other similar serial ensemble filters) that is especially suitable for dense observations that typically use relatively small horizontal covariance localization radii. Most NWP models, including the ARPS and the Weather Research and Forecasting model (WRF), use horizontal domain decomposition for effective parallelization (Sathye et al. 1997; Michalakes et al. 2004). A domain-decomposition-based parallel DA strategy is attractive because it can share much of the parallelization infrastructure with the prediction model. If the DA system and prediction model use the same number and configuration of subdomains, transfer of model grids between the two systems will be more straightforward either through disk or within computer memory. Furthermore, with typical ensemble DA systems, the state arrays are usually moved between the prediction model and DA system through disk I/O within the DA cycles; such I/O can take more than half of the total wall clock time within each cycle (Szunyogh et al. 2008), making high-frequency assimilation of observations on large, high-resolution, grids prohibitively expensive. Our eventual goal is to achieve data exchange through message passing within computer memory, bypassing disk I/O altogether; adopting a domain decomposition parallelization

strategy would simplify this process. Finally, the domain decomposition strategy makes grid-based calculations within the DA system, such as spatial interpolation, easier.

The domain-decomposition-based strategy we propose takes advantage of the relatively small localization radii typically used by very dense observations within ensemble algorithms, because observations that do not influence state variables at the same grid points can be processed in parallel. More sparse conventional observations tend to require larger localization radii (Dong et al. 2011) and are therefore more difficult to process in parallel. In this case, a strategy similar to that of AC07 is taken, in which observations are processed serially but still using the same decomposed domains. Parallelization can be achieved at the state variable level in the case; in other words, different parallelization strategies can be used in combination, taking advantage of the serial nature of the ensemble algorithms. Note that this approach scales well only for observations whose localization radius is large enough to impact most of the grid points in the model domain, unless additional steps are taken to balance the load, as in AC07.

In addition to domain-decomposition-based parallelization, we also want to take advantage of SMP capabilities of multi-core compute nodes that are available on essentially all large parallel systems of today. SMP among cores on the same node eliminates explicit data transport among the cores, thus reducing communication costs and contention for interconnect ports. By performing domain decomposition for the nodes while parallelizing across the PEs (e.g., cores) on the same PUs (e.g., nodes), the decomposed domains can be larger relative to the localization radii, increasing the chance that observations on different decomposed domains can be processed independently.

For the EnSRF algorithm, SMP is easily achieved at the state variable level, because each observation will need to update all state variables within its localization radius, and these update

operations are independent. Thus, the state variable update can be parallelized using OpenMP directives applied to the loops over the state variables. The combination of MPI and OpenMP strategies gives hybrid parallelization. This paper describes a hybrid parallel scheme implemented for the ARPS EnSRF system. In addition, observation data are organized into batches to improve the load balance when assimilating data from a number of radars.

This paper is organized as follows. Section 2 reviews the EnSRF formulation and briefly describes the ARPS model used in timing experiments. Section 3 introduces the parallel algorithms for high-density radar data and conventional observations separately. It also describes the OpenMP/MPI hybrid strategy as well as the observation organization. Validation of the parallel implementation and its performance are examined in section 4. A summary and conclusions are presented in section 5.

2. The ARPS ensemble DA system

The ARPS (Xue et al. 2000; Xue et al. 2001; Xue et al. 2003) model is a general-purpose, multi-scale prediction system in the public domain. It has a non-hydrostatic, fully compressible dynamic core formulated in generalized terrain-following coordinates. It employs the domain decomposition strategy in the horizontal for massively parallel computers (Sathye et al. 1997; Xue et al. 2007), and has been tested through real-time forecasts at convection-permitting/allowing resolutions for many years (e.g., Xue et al. 1996), including forecasts in continental US (CONUS-scale) domains at 4 and 1 km grid spacing (e.g., Xue et al. 2011), assimilating data from all radars in the WSR-88D radar network using a 3DVAR method.

As mentioned earlier, the current ARPS EnKF DA system (Xue et al. 2006) is primarily based on the EnSRF algorithm of Whitaker and Hamill (2002). In addition, an asynchronous (Sakov et al. 2010) four-dimensional EnSRF (Wang et al. 2013) has also been implemented. The

system includes capabilities for parameter estimation (Tong and Xue 2008), dual-polarimetric radar data assimilation (Jung et al. 2008), simultaneous reflectivity attenuation correction (Xue et al. 2009), and the ability to handle a variety of data sources (Dong et al. 2011). Additionally, it has been coupled with a double-moment microphysics scheme (Xue et al. 2010; Jung et al. 2012). To be able to apply this system to large, convection-resolving domains such as those used by ARPS 3DVAR for continental scale applications (e.g., Xue et al. 2011) and be able to assimilate frequent, high-volume observations, efficient parallelization of the system is essential.

Briefly, in EnSRF, the ensemble mean and ensemble deviations are updated separately. The analysis equations for ensemble mean state vector $\overline{\mathbf{x}}$ and the ensemble deviations $\mathbf{x}_i'$ are, respectively,

$$\overline{\mathbf{x}}^a = \overline{\mathbf{x}}^b + \boldsymbol{\rho} \circ \mathbf{K}\left[\mathbf{y}^o - H(\overline{\mathbf{x}}^b)\right], \tag{1}$$

$$\mathbf{x}_i'^a = \beta(\mathbf{I} - \alpha\boldsymbol{\rho} \circ \mathbf{K}H)\mathbf{x}_i'^b \tag{2}$$

where $\mathbf{K}$ is the Kalman gain and $\boldsymbol{y^o}$ the observation vector. Subscript i denotes the ensemble member and ranges from 1 to N with N being the ensemble size. H is the forward observation operator that projects state variables to observed quantities, which can be nonlinear. Symbol $\circ$ in the equations represents the Schur (element-wise) product and $\boldsymbol{\rho}$ is the localization matrix, containing localization coefficients that are typically functions of the distance between the observation being processed and the state variable being updated. The analysis background $\overline{\mathbf{x}}^b$ projected into observation space, i.e., $H(\overline{\mathbf{x}}^b)$, is called the observation prior. Superscripts a, b, and o denote analysis, background, and observation, respectively. State vector $\mathbf{x}$ includes in our case the grid point values of the three wind components (u, v, w), potential temperature (θ), pressure (p), the mixing ratios of water vapor (q_v), cloud water (q_c), rain water (q_r), cloud ice (q_i),

snow (q_s), and hail (q_h). When a two-moment microphysics parameterization scheme is used, the total number concentrations for the 5 water and ice species are also part of the state vector (Xue et al. 2010). Background state vectors $\overline{\mathbf{x}}^b$ and $\mathbf{x}_i'^b$ are either forecasts from the previous assimilation cycle or the states updated by observations processed prior to the current one. The parameter β is the covariance inflation factor. Variable α is a factor in the square root algorithm derived by Whitaker and Hamill (2002),

$$\alpha = \left[1 + \sqrt{\mathbf{R}\left(\mathbf{H}\mathbf{P}^b\mathbf{H}^{\mathrm{T}} + \mathbf{R}\right)^{-1}} \right]^{-1}. \tag{3}$$

Here, $\mathbf{R}$ is observation error covariance matrix, $\mathbf{P^b}$ the background error covariance matrix, and $\mathbf{H}$ the linearized observation operator. The Kalman gain matrix $\mathbf{K}$ is given by

$$\mathbf{K} = \mathbf{P}^b\mathbf{H}^{\mathrm{T}}\left(\mathbf{H}\mathbf{P}^b\mathbf{H}^{\mathrm{T}} + \mathbf{R}\right)^{-1}. \tag{4}$$

In the above, matrices $\mathbf{P}^b\mathbf{H}^{\mathrm{T}}$ and $\mathbf{H}\mathbf{P}^b\mathbf{H}^{\mathrm{T}}$, representing the background error covariance between the state variables and observation priors, and that between observation priors, respectively, are estimated from the background ensemble, according to

$$\mathbf{P}^b\mathbf{H}^{\mathrm{T}} = \frac{1}{N-1}\sum_{i=1}^{N}\left(\mathbf{x}_i^b - \overline{\mathbf{x}}^b\right)\left[H(\mathbf{x}_i^b) - \overline{H(\mathbf{x}^b)} \right]^{\mathrm{T}}, \tag{5}$$

$$\mathbf{H}\mathbf{P}^b\mathbf{H}^{\mathrm{T}} = \frac{1}{N-1}\sum_{i=1}^{N}\left[H(\mathbf{x}_i^b) - \overline{H(\mathbf{x}^b)} \right]\left[H(\mathbf{x}_i^b) - \overline{H(\mathbf{x}^b)} \right]^{\mathrm{T}}. \tag{6}$$

The overbars in Eqs. (5) and (6) denote the ensemble mean. When a single observation is analyzed, $\mathbf{P}^b\mathbf{H}^{\mathrm{T}}$ becomes a vector having the length of the state vector $\mathbf{x}$. In practice, due to covariance localization, all elements in $\mathbf{P}^b\mathbf{H}^{\mathrm{T}}$ are not calculated; those for grid points outside the localization radius of a given observation are assumed to be zero. In fact, it is this assumption that makes the design of our parallel algorithm practical; *observations whose domains of influence (as constrained by the covariance localization radii) do not overlap can be analyzed*

209 *simultaneously*. Another basic assumption with this algorithm (and most atmospheric DA
210 algorithms) is that observation errors are uncorrelated, so that observations can be analyzed
211 sequentially in any order. When the observations are processed serially, one at a time, the
212 observation error covariance matrix $\mathbf{R}$ reduces to a scalar, as does matrix $\mathbf{HP}^b\mathbf{H}^{\mathrm{T}}$. In this case,
213 $\mathbf{HP}^b\mathbf{H}^{\mathrm{T}}$ is the background error variance at the observation point.

214 After an observation is analyzed based on Eqs. (1)-(6), the analyzed ensemble states $\mathbf{x}_i^a$
215 ($i = 1 \cdots N$), the sum of ensemble mean and deviations, become the new background states $\mathbf{x}_i^b$
216 for the next observation, and the analysis is repeated until all observations at a given time are
217 analyzed. An ensemble of forecasts then proceeds from the analysis ensemble until the time of
218 new observation(s); at that time the analysis cycle is repeated.

219 **3. The parallel algorithm for EnSRF**

220 For convective-scale weather, Doppler weather radar is one of the most important
221 observing platforms. The US National Weather Service (NWS) operates a network of over 150
222 Weather Surveillance Radar-1988 Doppler (WSR-88D) radars that continuously scan the
223 atmosphere, at a rate of one full volume scan every 5-10 minutes, producing radial velocity and
224 reflectivity data. One volume scan in precipitation mode typically contains 14 elevations with
225 approximately several million observations every 5 minutes.

226 The number of conventional observations, such as surface station measurements, upper
227 air soundings, and wind profiler winds, is small compared to radar observations; because they
228 typically represent weather phenomena of larger scales, their assimilation in EnKF typically uses
229 larger covariance localization radii, and therefore their influence reaches larger distances (Dong
230 et al. 2011). Because of the different characteristics of each data type, different parallel strategies
231 are employed for conventional and radar data.

a. The parallel algorithm for high-density observations with small covariance localization radii

The algorithm partitions the entire analysis domain into subdomains defined by the number of participating MPI processes in the horizontal x and y directions. No decomposition is performed in the vertical direction, and therefore, state variables are always complete in the vertical columns. High-density radar observations (and other high-resolution observations including those of satellite) are distributed to each subdomain according to their physical locations. Fig. 1 illustrates an analysis domain that is partitioned into 4 physical subdomains horizontally, to be handled by 4 PUs in the computing system. Each computational domain is comprised of the physical subdomain (in darker gray for P1, separated with thick solid lines) and extended boundary 'halo' zones surrounding the physical subdomain (in light gray for P1, bounded by thin lines); the physical domain and the boundary halo zones combined together are called computational subdomains. The width of the extended boundary halo zone for the DA system is typically larger than the halo zone or 'ghost cells' needed for boundary condition exchanges in parallel NWP models based on domain decomposition (e.g., Sathye et al. 1997). The width of the halo zone in the ARPS model, for example, is only one grid interval on each boundary.

The extended boundary zone on each side must be at least as wide as the maximum localization radius (R) of observations handled by the algorithm in the subdomain. For radar observations, R is usually equal to a few grid intervals. Each physical subdomain is further divided into 4 patches that are separated by bold dashed lines in Fig. 1, and these patches are labeled S1, S2, S3 and S4, respectively. The horizontal width of patch S2 and the vertical height of patch S3 must be at least $2R$. The rest of the physical domain is assigned to patches S1 and S4 as in Fig. 1, and their horizontal width and height also must be at least $2R$. Thus, the width of the

physical subdomain must be larger than $4R$ for the algorithm to work. All other subdomains in Fig. 1 are divided following the same patch pattern. Such a patch division assures that patches with the same label in adjacent subdomains are at least $2R$ apart, so observations in any one patch do not affect grid points in the same patch on other PUs and thus, they can be analyzed in parallel. In other words, no two observations that are being analyzed in parallel will influence the same grid point. In practice, we want to make patch S1 as large as possible, increasing the chance that any two observations can be processed independently (see below). Thus, the width of S2 and the height of S3 are assigned the minimum possible size of $2R$ (see Fig. 1), which leaves the majority of the subdomain to patch S1.

The EnKF DA over the analysis domain is performed in 4 sequential steps for observations within S1, S2, S3 and S4. In the first step, only observations within S1 on all PUs are assimilated in parallel while observations on each S1 patch are assimilated sequentially. Let P be the number of PUs. Then, there can be at most P observations being assimilated in parallel at any time. After all observations located within S1 are assimilated, MPI communications are required to properly update state variables at grid points within the extended boundary zones that are shared with neighboring PUs. The same procedure is then repeated for observations within S2, S3 and S4 in steps 2, 3, and 4.

The assimilation of observations within the same-labeled patches from all PUs can be done in parallel because: 1) the grid points influenced by the observations analyzed in parallel are separated far enough without overlap, and 2) the ensemble state arrays are extended beyond the physical subdomain, so that the influence on state grids by observations within each subdomain can be passed to its neighbor PUs with MPI communications. Best load balancing is realized if the same-labeled patches contain the same number of observations so that all PUs can

complete each analysis step in approximately the same time. In practice, however, the number of observations on each subdomain is usually different due to uneven spatial distribution of observations (and of observation types). One way to improve parallelism is to make one patch (S1 in our system) as large as possible, which increases the number of observations that can be processed independently and improves the load balance. Assimilation of observations on S2, S3 and S4 may not be well balanced. However, because they tend to be smaller and contain fewer observations, their effect on the assimilation time tends to be small.

Since high-density observations, such as radar data, usually assume relatively small localization radii, the constraint that the width of the physical subdomain should be at least $4R$ in each direction usually does not become a major problem, especially when the DA domain is large. When a hybrid MPI-OpenMP parallelization strategy is used this problem can be further alleviated (see later). While the proposed algorithm is valid for most meteorological observations that can assume a small localization radius, certain 'integral observations' such as radar reflectivity with path-integrated attenuation effect (e.g., Xue et al. 2009) and GPS slant-path water vapor (e.g., Liu and Xue 2006) pose special challenge for the serial EnSRF algorithm in general since their observation operators are non-local (Campbell et al. 2010).

b. The parallel algorithm for conventional observations with large covariance localization radii

Currently supported conventional observations in the ARPS EnKF system include surface station, upper air sounding, wind profiler, and aircraft observations. Since the covariance localization radii applied to these observations are usually large, the width of the extended boundary zones described in section 3a would be impractical for these data, unless the decomposed subdomains are much larger than the localization radii. This is usually only true when a small number of subdomains is used. Therefore, we design and implement an alternative

301 algorithm for this type of observations. Because the number of conventional (or any other
302 coarse-resolution) observations is typically much smaller than the number of (dense) radar
303 observations, we can afford to process the observations serially while trying to achieve
304 parallelism at the state variable level, similar to the strategy taken by AC07.

305 In our current implementation, conventional observations within the entire analysis
306 domain are broadcast to all PUs and assimilated one by one. Only the PU containing the
307 observation to be analyzed computes the observation prior; it then broadcasts the observation
308 prior ensemble, $H(\mathbf{x}_i)$, to all other PUs. The state variables within the covariance localization
309 radius of this observation are updated simultaneously on each PU that carries the state variables
310 (Fig. 2). Since we do not need extra boundary zones, state variable updating occurs within the
311 computational subdomains of the original NWP model. However, a set of MPI communications
312 between PUs is still needed right after the analysis of each observation to update the state
313 variables within the halo zone to facilitate the spatial interpolation involved in observation
314 operators. These steps are repeated until all observations are assimilated.

315 Our current implementation does not pre-calculate $H(\mathbf{x})$ or update $H(\mathbf{x})$ as part of the
316 extended state vector as AC07 does, and we use a regular domain decomposition strategy to
317 distribute the state variables across the PUs. This implementation will have load balance issues
318 for conventional observations, especially when the covariance localization radii of these
319 observations are small relative to the size of the entire model domain. AC07 mitigates this
320 problem by distributing the state variables across PUs as heterogeneously as possible, i.e., by
321 distributing neighboring grid points across as many PUs as possible. Such an irregular
322 distribution of state variables makes it difficult to implement grid-point-based treatments within
323 the EnKF algorithms. The $H(\mathbf{x})$ pre-calculation and update strategy employed by AC07 allows

simultaneous calculation of observation priors. This can be an option in a future implementation; in fact, the 4D EnSRF algorithm implemented by Wang et al. (2013) employs this strategy.

c. Hybrid MPI-OpenMP parallelization

All current supercomputers use compute nodes with multiple shared-memory cores. The original ARPS EnSRF code supports OpenMP parallelization via explicit loop-level directives at the state-variable-update level (Xue et al. 2006). Thus, it is straightforward to employ a hybrid technique, using SMP among cores on the same node and DMP via MPI across nodes. Doing so can reduce explicit data communication within nodes and allow for larger S1 patches within the decomposed domains on each PU (see Fig. 1). Our hybrid implementation is designed such that each MPI process spawns multiple threads. Since message passing calls are outside of the OpenMP parallel sections, they are parallel thread safe, i.e., only the master thread in a process makes calls to MPI routines. The final program is flexible enough to run in MPI only, OpenMP only, or in MPI/OpenMP hybrid modes, on a single-node workstation or supercomputers made up of multiple nodes.

d. Parallel strategy for assimilating data from multiple radars

In the ARPS EnKF system, full-resolution radar observations in the radar coordinates are usually mapped horizontally to the model grid columns during preprocessing (Brewster et al. 2005). The original ARPS EnSRF implementation processes data from one radar at a time, sequentially. This is convenient because the data are stored in arrays for individual radars on elevation levels (Xue et al. 2006). For data from the same radar, only a few parameters are needed to describe the radar characteristics. However, because each radar typically covers only a portion of the model domain, this procedure severely limits the scalability of the analysis system

346 due to load imbalances (see Fig. 3). Figure 3a illustrates a domain that contains six radars labeled
347 A through F. If this domain is decomposed into four subdomains, all PUs, except P1, will be idle
348 when data from radar A are assimilated. The same is true for radars B through F. To mitigate this
349 problem, we develop a procedure that merges radar data into composite sets or batches so that
350 data from multiple radars can be processed at the same time.

351 In the analysis program, all vertical levels of radar observations at each horizontal grid
352 location are stored continuously as a vector column. The most general approach is to store all
353 columns of radar data in a single dynamically allocated storage array or data structure while
354 keeping track of the radar characteristics associated with each column. Each column may contain
355 different numbers of available radar elevations. When overlapping coverage exists, the grid
356 columns covered by multiple radars will have multiple columns of data (see Fig. 3a). To keep
357 track of data in reference to the analysis grid, it is convenient to define arrays that have the same
358 dimensions as the model grid in the horizontal directions, but such arrays will only be able to
359 store no more than one column of data at each grid location unless the last dimension is defined
360 dynamically or pre-defined to be large enough. While for optimally tuned EnKF, the order in
361 which observations are assimilated should not matter, in practice, because the ensemble spread
362 can be reduced too much by observations processed earlier before covariance inflation is applied,
363 the order of observation processing sometimes do matter somewhat. For this reason, we group
364 the radar data into several batches, the number of which is no bigger than the maximum number
365 of radars covering the same spot anywhere in the analysis domain. For a radar network that is
366 designed to maximize spatial coverage, such as the WSR-88D radar network, this maximum is
367 usually a single digit number; i.e., anywhere in the network, less than 10 radars observe the same
368 column.

Fig. 3 shows the spatial coverage of three batches of data that add up to all columns of data available; those three batches of observations will be processed in sequence. Within regions having multiple radar coverage, the radar from which data will be first picked can be chosen randomly or based on the order the data were input to the program. Alternatively, the data columns from the closest radar can be picked first. The last option is more desirable, as it removes the randomness of the algorithm. Finally, because the radar data are no longer organized according to radar, additional two-dimensional arrays are needed to store parameters for each data column. When only a few elevations within a radar volume scan are analyzed using short (e.g., 1 to 2 minutes) assimilation cycles, the vertical dimension of the arrays storing the composite data sets need only to be a few.

With the above implementation, the load balance is significantly improved for the first composite data set. It should be noted that we usually assimilate reflectivity data even in precipitation-free regions, which has the benefit of suppressing spurious storms (Tong and Xue 2005). We note that load imbalance does still exist with radial velocity data in the first group since they are usually only available in precipitation regions; however, their numbers are usually much smaller than the total number of reflectivity data. In addition, load imbalances usually exist with the second group of data and above but again the volume of data in these groups is small since they only exist in overlapping regions, and these regions are usually spread over the assimilation domain.

4. Algorithm verification and performance analysis

a. Verification of the parallelized code

The domain partition and batch processing inevitably change the sequence of observations being assimilated in the EnKF system. Theoretically, the order in which the

observations are processed does not matter for observations with uncorrelated errors, to the extent that sampling error does not impact the results. In practice, the analysis results may differ significantly if the filter is not properly tuned, where the tuning typically includes covariance inflation and localization.

A set of experiments has been performed to investigate the effect of domain decomposition on the analysis of simulated radar observations in an observing system simulation experiment (OSSE) framework. Convective storms are triggered by five 4-K ellipsoidal thermal bubbles with a 60-km horizontal radius and 4-km vertical radius in an environment defined by 20 May 1977 Del City, Oklahoma, supercell sounding (Ray et al. 1981). The model domain is $300\times200\times16$ km^3 with horizontal and vertical grid spacings of 1 km and 500 m, respectively. Forty ensemble members are initiated at 3000 seconds of model time. The full state vector has 1.4×10^9 elements. Simulated radar observations from three radars are produced, using the standard WSR-88D VCP (Volume Coverage Pattern) 11, which contains 14 elevation levels. The total number of observations is approximately 6.7×10^5 from three volume scans spanning 5 minutes each. Radar DA is first performed at 5-minute intervals from 3300 to 5700 seconds, using the original serial ARPS EnSRF code to provide an ensemble for subsequent parallel assimilation tests. The Milbrandt and Yau (2005) double-moment microphysics scheme is used in both truth simulation and in DA. The environment and model configurations that are not described here can be found in Xue et al. (2010).

Three parallel DA experiments are then performed at 6000 seconds, one running in pure MPI mode, one in pure OpenMP mode, and one in pure OpenMP mode but processing observations serially in a reversed order. These experiments are referred as MPI, OMP_F, and OMP_B (F for forward and B for backward), respectively. For each experiment, average RMS

errors for the state variables are computed against the truth simulation at the grid points where truth reflectivity is greater than 10 dBZ. The RMS errors of MPI and OMP_B are normalized by the RMS errors of OMP_F and shown in Fig. 4 for individual state variables. Most of the normalized errors are very close to 1, and all of them are between 0.95 and 1.05 for MPI. Among the variables, the total number concentration for rain water shows the largest variability, probably due to the high sensitivity of reflectivity to the rain drop size distribution. In fact, the normalized error for rain water number concentration is an outlier for OMP_B, reaching close to 1.25, much larger than the normalized error of about 1.05 for MPI. These results suggest that the effect of the domain partition on the analysis is small, and the differences are within the range of sampling uncertainties of the ensemble system.

With respect to the parallel code implementation for conventional data analysis, domain decomposition does not change the sequence of the observation processing (see section 3b). Therefore, identical results from experiments OMP_F and MPI are guaranteed. The results from the experiments when simulated surface observations are also included are not shown here.

b. Performance evaluation with OSSE experiments

The performance of our parallel EnKF system is evaluated with radar DA benchmark experiments on a Cray XT5 system (called Kraken) at National Institute of Computational Science (NICS) at the University of Tennessee, which has 9408 total compute nodes with 12 cores each (6 cores per processor, 2 processors per node), giving a peak performance of 1.17 petaFLOPS. With Kraken, users can set the number of MPI processes per node (1-12), the number of MPI processes per processor (1-6), and the number of cores (OpenMP threads) per MPI process (1-12). A number of experiments with combinations of different numbers of MPI processes, OpenMP threads, cores per node, and cores per processor have been performed to

examine the timing performance of various configurations. The same case described in section 4a is used for benchmarking.

First, the scalability of the OpenMP implementation is investigated as a reference. Since each Kraken node contains only 12 cores, the maximum number of threads that can be used for an OpenMP job is 12. The OpenMP implementation shows scalability up to 8 cores (see Table 1), beyond which the reduction in wall clock time becomes minimal. One very likely reason is the contention accessing shared memory and cache by different cores of the Opteron processors used.

To evaluate the performance of our MPI implementation, we ran several OpenMP and MPI experiments on a single compute node. Table 1 lists the wall clock times and relative speedups for these experiments. The experiment names follow the convention o[*total cores used*] for OpenMP and m[*nproc_x*]×[*nproc_y*] for MPI experiments, where *nproc_x* and *nproc_y* denote the number of PUs corresponding to the decomposed domains in the *x* and *y* directions, respectively. Generally, the OpenMP jobs perform better than their MPI counterparts using the same number of cores when running on a single node due to the communication overhead of MPI processes and possibly better load balance with OpenMP. It is also noticed that the wall-clock time is heavily influenced by the domain partitioning configuration in the *x* and *y* directions. For example, m02×01 takes almost 1.4 times longer than m01×02, although both use the same number of cores. Since FORTRAN arrays are stored contiguously in the column-major order in the computer memory, a run that has a smaller partition number in the *x* direction than the *y* direction (e.g., m01×02) is better at taking advantage of the spatial locality of the data in memory. This can accelerate data loading from main memory into cache and improve cache reuse. Conversely, an inefficient partition can degrade the performance even when more system resources are used. For example, m03×02 using 6 cores has a much smaller speed improvement

over m01×01 than experiments using 4 cores or even some experiments using 2 cores. These results suggest that finding the optimal domain decomposition is important in achieving the best performance with the given system resources.

Table 2 shows performance data collected from pure MPI runs, and from hybrid MPI/OpenMP experiments that run on 4 Kraken nodes. All experiments are named as following: m(h)[*nproc_x*]×[*nproc_y*]_[*number of processes per node*]o[*number of threads per process*], where "m" denotes MPI runs and "h" denotes hybrid runs. For MPI runs, the number of threads per process is always 1. Thus, "o[*number of threads per process*]" is omitted from the notations for all MPI runs in Table 2. Since each Kraken node contains two processors, the processes on each node are distributed to those processors as evenly as possible in order to obtain the best possible performance.

It is found that the domain partitioning again plays an important role for the DA system performance. For example, experiments that use 20 cores in total on 4 compute nodes show large variability in the execution time. Among these experiments, m02×10_05 has the best performance, suggesting that m02×10_05 utilizes the system cache most efficiently and/or has the least message-passing overhead given 20 cores. Generally, the MPI experiments using more nodes perform better than those experiments with the same domain partitioning but using fewer nodes. For example, m01×04 in Table 1 that uses one compute node takes 2660 seconds to finish while m01×04_01 in Table 2 running on 4 compute nodes takes only 2343 seconds. This is consistent with the observation that performance is improved as available cache size increases. Adding more processes improves the performance on 4 compute nodes. As an example, m06×08_12 takes less time than those experiments using 40 cores or less. This is because more observations can be processed in parallel in the m06×08_12 experiment than the others, even

484 though MPI communication costs are higher than in the other experiments. However, as
485 observed before with OpenMP experiments, access contention for the memory bandwidth and
486 the cache sharing as more cores are used may impede the performance at some point. It suggests
487 that there is a tradeoff between the number of processes and available computing resources and,
488 therefore, finding optimal configurations for MPI runs may not be straightforward because it
489 depends on a number of hardware factors.

490 For the hybrid runs, the wall-clock times of m01×04_01 (i.e. h01×04_01o1),
491 h01×04_01o2, h01×04_01o4, h01×04_01o6, h01×04_01o8 and h01×04_01o12 decrease
492 monotonically, in that order. The decreasing trend of wall-clock time with increasing number of
493 threads is consistently found in other similar sets of experiments. It is also found that the hybrid
494 runs are as sensitive as the MPI runs to the domain partitioning, available cache, and other
495 hardware configuration factors. A hybrid experiment can outperform or underperform the
496 corresponding MPI experiments using the same resources (number of cores and number of nodes)
497 depending on the configuration (Table 2 and 3). For example, the minimum wall-clock time with
498 8 cores from 4 nodes in hybrid mode is 1471 seconds, which is smaller than the minimum time
499 required by a MPI run with 8 processes on 4 nodes (2169 seconds) in Table 3. On the other hand,
500 h01×04_01o12 takes 733 seconds, more than the 606 seconds of m06×08_12, which uses the
501 same resources. It is also observed that a larger improvement is achieved by the hybrid jobs with
502 a fewer number of threads. This is because observations are processed one by one with OpenMP
503 processes. By using more MPI processes rather than more OpenMP threads, we can assimilate
504 more observations simultaneously and, hence, improve the parallel efficiency (see section 4c for
505 more details). In addition, cache availability and memory access contention with a large number
506 of threads in the hybrid experiments also affect program performance.

c. Performance evaluation with a real data application

The parallel ARPS EnKF system is applied to the 10 May 2010 Oklahoma-Kansas tornado outbreak case. Over 60 tornadoes, with up to EF4 intensity, affected large parts of Oklahoma and adjacent parts of southern Kansas, southwestern Missouri, and western Arkansas on that day. This real data case is run on an SGI UV 1000 cc-NUMA shared-memory system at the Pittsburgh Supercomputing Center (PSC). The system, called Blacklight, is comprised of 256 nodes containing 2 eight-core Intel Xeon processors each; its theoretical peak performance is 37 teraFLOPS. The cc-NUMA architecture allows for SMP across nodes. Up to 16 terabytes (TB) of memory can be requested for a single shared memory job, while hybrid jobs can access the full 32 TB of system memory.

The EnSRF analyses are performed on a grid with 4 km horizontal grid spacing, using 40 ensemble members. The domain consists of 443×483×53 grid points, and the model state includes three velocity components, potential temperature, pressure, and mixing ratios of water vapor, and 5 water and ice species. A single-moment microphysics scheme is used. The state vector has 4.9×10^9 elements. Observations of radar reflectivity and radial velocity from 35 radars are analyzed from 1705 UTC to 1800 UTC at 5-minute intervals. Fig. 5 presents a comparison between the radar observation mosaic at 1800 UTC on 10 May 2010 and the corresponding analysis results by the parallel ARPS EnSRF system. Overall, the analyzed reflectivity exhibits a good fit to the observations in shape, structure, and intensity. The exceptions are several echoes in Texas, southeast Montana, and northwest Colorado, which are due to the incomplete radar coverage over those areas. Several timing benchmark analyses at 1710 UTC are performed. There are about 1.3×10^6 observations from the 35 radars at this time (see Fig. 6), more than any of the other times in the analysis window.

530 Our parallel benchmark experiments are run in pure OpenMP, pure MPI and hybrid
531 MPI/OpenMP modes. In all cases, all cores on the compute nodes were fully utilized, either by
532 individual MPI processes or by OMP threats. The experiment names and their configurations are
533 listed in Table 4. Guided by the timing results on Kraken, experiments are designed to use the
534 most optimal configurations, i.e., with a larger number of PUs in the y direction than in the x
535 direction. Each experiment in Table 4 was repeated 7 times. Because the timing results on
536 Blacklight show up to 185% variability due to system load variations, the best timing results for
537 each case are selected and presented here. Fig. 7 shows the best timing results of each case as a
538 function of the number of cores used. Very large variations in run time were found to be
539 attributable to disk I/O on a large shared file system; I/O times are therefore excluded in Fig. 7 to
540 allow us to focus on the time spent on the analyses. The times with and without including
541 message passing are shown.

542 Both MPI and hybrid runs show good scalability according to Fig. 7, and they outperform
543 pure OpenMP runs by a large margin except for the case of 16 cores. Because each physical node
544 of Blacklight has only 16 cores, when more than 16 cores are used by OpenMP, the memory
545 access will be across different physical nodes; this clearly leads to reduced parallelization
546 efficiency with the OpenMP runs. Also, with pure OpenMP, the parallelization is limited to the
547 state variable level, meaning all observations have to be processed serially (i.e., not
548 parallelization at the observation level).

549 Fig. 7 also shows that, when using the same amount of total resources, the hybrid runs
550 generally outperform pure MPI runs when both analysis and message passing times are included.
551 For the same number of cores used, pure MPI runs implies more PUs, i.e., more message passing
552 requests. Even though the pure MPI mode may be able to parallelize more at the observation

553 level, the message passing overhead can reduce the benefit. Not surprisingly, the hybrid
554 OpenMP/MPI runs are better in terms of total computational time. Among the hybrid groups,
555 jobs with fewer threads hence more MPI processes seem to give better performances, in terms of
556 the analysis time. This suggests that assimilating observations in parallel via MPI processes gives
557 a greater benefit before the increased message passing overhead becomes overwhelming.

558 We have noticed that I/O can easily take 60% to 80% of the total wall-clock time with
559 experiments in which all data I/O were handled by a single MPI process or the master OpenMP
560 thread. This I/O time can be reduced by distributing I/O loads among the MPI processes (but not
561 among OpenMP threads). Therefore, our solution is to let each MPI process read and write data
562 within its own subdomain, in the form of "split files". This improves I/O parallelization and also
563 reduces time needed for communicating gridded information across PUs. With split files, only
564 data within the extended boundary zones need to be exchanged with neighboring PUs. Due to the
565 large variations in the I/O times collected on Blacklight, we ran another set of experiments on a
566 supercomputer with more consistent I/O performance between runs. It consists of 2.0 GHz quad-
567 core Pentium4 Xeon E5405 processors, with 2 processors on each node. Tests with split files on
568 this system, corresponding to h04×08_01o8 (see above naming conventions), reveal that the
569 times spent on I/O and message passing are reduced (the latter because of the reduced exchanges
570 of gridded information across MPI processes); the total wall-clock time for I/O and message
571 passing for one experiment was reduced from 1231 seconds to 188 seconds using split files.

572 **5. Summary and conclusions**

573 A parallel algorithm based on the domain decomposition strategy has been developed and
574 implemented within the ARPS EnKF framework. The algorithm takes advantage of the relatively
575 small spatial covariance localization radii typically used by high-resolution observations such as

those of radar. Assuming that the maximum horizontal covariance localization radius of the observations to be analyzed in parallel is R, the horizontal area of a decomposed physical subdomain should be at least $4R \times 4R$. An additional boundary zone of width R is added to each side of the physical subdomains to create enlarged computational subdomains, which facilitate information exchanges between neighboring subdomains. Each subdomain is assigned to one processing unit (PU), within which no MPI message passing is required. The subdomains are then further divided up into 4 sub-patches, denoted S1 through S4. The width and height of each patch are required to be at least $2R$ to ensure any two observations that may be processed in parallel are well separated. In practice, the size of S1 is made as large as possible within its subdomain to increase the probability that observations from different subdomains can be processed in parallel.

Observations within the 4 patches are processed sequentially, but data in the patches with the same label in different subdomains are processed simultaneously. Distributed-memory parallelization is therefore achieved at the observation level. The patch division ensures that most of the analysis work is done in parallel when processing observations within patches S1 of all PUs. To handle the load imbalance issue when assimilating observations from many radars, the observation arrays are organized into batches. The maximum number of batches is limited by the maximum number of radars covering the same location anywhere in the analysis domain. Such an observation organization improves the uniformity of observation distribution within the first observation batch and thereby improves load balance.

Conventional data that use larger covariance localization radii are still processed serially. State variables influenced by a particular observation are updated synchronously on the PUs carrying those state variables.

599 The algorithm supports three parallel modes: pure OpenMP, pure MPI, and
600 MPI/OpenMP hybrid. Within the PUs with multiple cores, shared-memory parallelization can be
601 achieved via OpenMP at the state variable update level. OpenMP parallelization reduces
602 message passing overhead and allows for larger decomposed domains, making the $4R$
603 requirement easier to satisfy.

604 It was first confirmed via OSSEs that changing the sequence of observation processing
605 due to domain decomposition has little impact on the analysis. Parallel DA benchmark
606 experiments are performed on a Cray XT5 machine. The OpenMP implementation shows
607 scalability up to 8 threads, beyond which memory and cache access contention limit further
608 improvement. MPI and OpenMP runs on a single compute node show that OpenMP
609 parallelization runs faster because of the lower communication overhead. MPI jobs with a
610 smaller number of partitions in the x direction than in the y direction exhibit better performance.
611 The same also applies to most of the hybrid jobs, although all hybrid jobs do not outperform the
612 corresponding MPI jobs.

613 A real data case involving 35 radars is tested on an SGI UV 1000 cc-NUMA system
614 capable of shared-memory programming across physical nodes. Poor scalability with pure
615 OpenMP is observed when more than one node is used, but both MPI and hybrid runs show good
616 scalability on this system. Excluding message passing time, pure MPI runs exhibit best
617 performance. When message-passing time is included, the hybrid runs generally outperform pure
618 MPI runs. For this real data case, the EnKF analysis excluding I/O can be completed within 4.5
619 minutes using 160 cores of the SGI UV 1000.

620 Given a fixed amount of resources, the hybrid jobs improve more over pure MPI jobs
621 with fewer numbers of threads. Because MPI processes realize parallelization at the observation

622 level, they are more efficient than OpenMP threads. However, there is a tradeoff between a
623 performance improvement due to the parallel processing of observations and degradation due to
624 increased message passing overhead. On the other hand, a pure OpenMP strategy for EnKF
625 shows good scalability on symmetric shared-memory systems but is limited by the number of
626 cores available on the individual node and by the physical memory available on the node. With
627 pure OpenMP, data I/O can only be handled by a single process, reducing the overall scalability.
628 The MPI/OpenMP hybrid strategy combines the strengths of both methods. However,
629 care must be taken when partitioning the domain, because the configuration of MPI domain
630 partitioning has a significant impact on the system performance. Given the same resources, jobs
631 with smaller numbers of partitions in the *x* direction tend to run faster because FORTRAN arrays
632 are stored in the column-major order in memory. Timing experiments have also shown that
633 determining the optimal decomposition configuration on a specific computing system is not
634 straightforward because the performance depends on factors such as the subdomain size in the *x*
635 and *y* directions, the number of cores on each node, the cache sizes and memory bandwidth
636 available to each core, and the networking topology across the nodes.
637 In all configurations, data I/O constituted a large portion of the execution time.
638 Experiments on a small dedicated Linux cluster show that the time spent on I/O and message
639 passing are reduced significantly by distributing I/O loads among the MPI processes with
640 MPI/OpenMP hybrid or pure MPI runs.
641 Although a data batching strategy is developed to reduce the load imbalance issue, further
642 improvement could be obtained through dynamic load balancing. Another problem is the low
643 resource utilization during inter-node communications because all threads are idle except one:
644 the master thread. The development of runtime management algorithms, for example the

Scalable Adaptive Computational Toolkit (SACT) (Li and Parashar 2007; Parashar et al. 2008), are expected to decrease runtime of the application automatically with reduced efforts from developers. Finally we point out that our parallel algorithm can be easily applied to other serial ensemble-based algorithm such as EAKF and the classic EnKF.

Acknowledgements: This work was primarily supported by NSF grant OCI-0905040, AGS-0802888, and NOAA Warn-on-Forecast Project under NA080AR4320904. Partial support was also provided by NSF AGS-0750790, AGS-0941491, AGS-1046171, and AGS-1046081. We acknowledge David O'Neal of Pittsburgh Supercomputing Center (PSC) for assistance with the use of Tuning and Analysis Utilities (TAU) on a PSC Altix cluster early in the work. Computations were performed at the Pittsburgh Supercomputing Center (PSC), and the National Institute for Computational Sciences (NICS), and the OU Supercomputing Center for Education and Research (OSCER).

References

Anderson, J. L., 2001: An ensemble adjustment Kalman filter for data assimilation. *Mon. Wea. Rev.*, **129,** 2884-2903.

Anderson, J. L., 2003: A local least square framework for ensemble filtering. *Mon. Wea. Rev.*, **131,** 634-642.

Anderson, J. L. and N. Collins, 2007: Scalable implementations of ensemble filter algorithms for data assimilation. *J. Atmos. Ocean. Technol.* , **24,** 1452-1463.

Bishop, C. H., B. J. Etherton, and S. J. Majumdar, 2001: Adaptive sampling with the ensemble transform Kalman filter. Part I: Theoretical aspects. *Mon. Wea. Rev.*, **129,** 420.

Brewster, K., M. Hu, M. Xue, and J. Gao, 2005: Efficient assimilation of radar data at high resolution for short-range numerical weather prediction. *WWRP Int. Symp. Nowcasting Very Short Range Forecasting*, CDROM 3.06.

Buehner, M., P. L. Houtekamer, C. Charette, H. L. Mitchell, and B. He, 2010: Intercomparison of variational data assimilation and the ensemble Kalman filter for global deterministic NWP. Part II: One-month experiments with real observations. *Mon. Wea. Rev.*, **138,** 1567-1586.

Burgers, G., P. J. v. Leeuwen, and G. Evensen, 1998: Analysis scheme in the ensemble Kalman filter. *Mon. Wea. Rev.*, **126,** 1719-1724.

Campbell, W. F., C. H. Bishop, and D. Hodyss, 2010: Vertical covariance localization for satellite radiances in ensemble Kalman filters. *Mon. Wea. Rev.*, **138,** 282-290.

Caya, A., J. Sun, and C. Snyder, 2005: A comparison between the 4D-VAR and the ensemble Kalman filter techniques for radar data assimilation. *Mon. Wea. Rev.*, **133,** 3081–3094.

Courtier, P. and O. Talagrand, 1987: Variational assimilation of meteorological observations with the adjoint equation. Part II: Numerical results. *Quart. J. Roy. Meteor. Soc.*, **113,** 1329-1347.

Dong, J., M. Xue, and K. K. Droegemeier, 2011: The analysis and impact of simulated high-resolution surface observations in addition to radar data for convective storms with an ensemble Kalman filter. *Meteor. Atmos. Phy.*, **112,** 41-61.

Dowell, D., F. Zhang, L. J. Wicker, C. Snyder, and N. A. Crook, 2004: Wind and temperature retrievals in the 17 May 1981 Arcadia, Oklahoma supercell: Ensemble Kalman filter experiments. *Mon. Wea. Rev.*, **132,** 1982-2005.

Dowell, D. C., L. J. Wicker, and C. Snyder, 2011: Ensemble Kalman filter assimilation of radar observations of the 8 May 2003 Oklahoma City supercell: Influence of reflectivity observations on storm-scale analysis. *Mon. Wea. Rev.*, **138,** 1152-1171.

Evensen, G., 1994: Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *J. Geophys. Res.*, **99,** 10143-10162.

Evensen, G., 2003: The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean Dynamics*, **53,** 343-367.

Evensen, G. and P. J. v. Leeuwen, 1996: Assimilation of geosat altimeter data for the Agulhas current using the ensemble Kalman filter with a quasigeostrophic model. *Mon. Wea. Rev.*, **124,** 85-96.

Gao, J., M. Xue, Z. Wang, and K. K. Droegemeier, 1998: The initial condition and explicit prediction of convection using ARPS adjoint and other retrievals methods with WSR-88D data. *12th Conf. Num. Wea. Pred.*, Phoenix AZ, Amer. Meteor. Soc., 176-178.

703 Hamill, T. M., J. S. Whitaker, M. Fiorino, and S. G. Benjamin, 2011: Global ensemble
704 predictions of 2009's tropical cyclones initialized with an ensemble Kalman filter. *Mon.*
705 *Wea. Rev.*, **139,** 668-688.
706 Houtekamer, P. L. and H. L. Mitchell, 1998: Data assimilation using an ensemble Kalman filter
707 technique. *Mon. Wea. Rev.*, **126,** 796-811.
708 Jung, Y., M. Xue, and M. Tong, 2012: Ensemble Kalman filter analyses of the 29-30 May 2004
709 Oklahoma tornadic thunderstorm using one- and two-moment bulk microphysics
710 schemes, with verification against polarimetric data. *Mon. Wea. Rev.*, **140,** 1457-1475.
711 Jung, Y., M. Xue, G. Zhang, and J. Straka, 2008: Assimilation of simulated polarimetric radar
712 data for a convective storm using ensemble Kalman filter. Part II: Impact of polarimetric
713 data on storm analysis. *Mon. Wea. Rev.*, **136,** 2246-2260.
714 Keppenne, C. L. and M. M. Rienecker, 2002: Initial testing of a massively parallel ensemble
715 Kalman filter with the Poseidon isopycnal ocean general circulation model. *Mon. Wea.*
716 *Rev.*, **130,** 2951-2965.
717 Le Dimet, F. X. and O. Talagrand, 1986: Variational algorithms for analysis and assimilation of
718 meteorological observations: Theoretical aspects. *Tellus*, **38A,** 97-110.
719 Li, X. and M. Parashar, 2007: Hybrid Runtime Management of Space-Time Heterogeneity for
720 Parallel Structured Adaptive Applications. *IEEE Transactions on Parallel and*
721 *Distributed Systems (TPDS)*, **18,** 1202-1214.
722 Liu, H. and M. Xue, 2006: Retrieval of moisture from slant-path water vapor observations of a
723 hypothetical GPS network using a three-dimensional variational scheme with anisotropic
724 background error. *Mon. Wea. Rev.*, **134,** 933–949.

Michalakes, J., J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, and W. Wang, 2004: The Weather Research and Forecast Model: Software Architecture and Performance. . *Proceedings, 11th ECMWF Workshop on the Use of High Performance Computing In Meteorology*, Reading U.K.

Milbrandt, J. A. and M. K. Yau, 2005: A multi-moment bulk microphysics parameterization. Part I: Aanlysis of the role of the spectral shape parameter. *J. Atmos. Sci.*, **62,** 3051-3064.

Parashar, M., X. Li, and S. Chandra, 2008: *Advanced Computational Infrastructure for Parallel and Distributed Adaptive Applications*. Vol. To appear, John Wiley & Sons.

Ray, P. S., B. Johnson, K. W. Johnson, J. S. Bradberry, J. J. Stephens, K. K. Wagner, R. B. Wilhelmson, and J. B. Klemp, 1981: The morphology of severe tornadic storms on 20 May 1977. *J. Atmos. Sci.*, **38,** 1643-1663.

Sakov, P., G. Evensen, and L. Bertino, 2010: Asynchronous data assimilation with the EnKF. *Tellus*, **62,** 24-29.

Sathye, A., M. Xue, G. Bassett, and K. K. Droegemeier, 1997: Parallel weather modeling with the Advanced Regional Prediction System. *Parallel Computing*, **23,** 2243-2256.

Snook, N., M. Xue, and J. Jung, 2011: Analysis of a tornadic meoscale convective vortex based on ensemble Kalman filter assimilation of CASA X-band and WSR-88D radar data. *Mon. Wea. Rev.*, **139,** 3446-3468.

Snyder, C. and F. Zhang, 2003: Assimilation of simulated Doppler radar observations with an ensemble Kalman filter. *Mon. Wea. Rev.*, **131,** 1663-1677.

Sun, J. and N. A. Crook, 1997: Dynamical and microphysical retrieval from Doppler radar observations using a cloud model and its adjoint. Part I: Model development and simulated data experiments. *J. Atmos. Sci.*, **54,** 1642-1661.

748 Szunyogh, I., E. J. Kostelich, G. Gyarmati, E. Kalnay, B. R. Hunt, E. Ott, and E. Satterfield,
749 2008: A local ensemble tranform Kalman filter data assimilation system for the NCEP
750 global model. *Tellus*, **60A,** 113-130.
751 Tippett, M. K., J. L. Anderson, C. H. Bishop, T. M. Hamill, and J. S. Whitaker, 2003: Ensemble
752 square root filters. *Mon. Wea. Rev.*, **131,** 1485-1490.
753 Tong, M. and M. Xue, 2005: Ensemble Kalman filter assimilation of Doppler radar data with a
754 compressible nonhydrostatic model: OSS Experiments. *Mon. Wea. Rev.*, **133,** 1789-1807.
755 Tong, M. and M. Xue, 2008: Simultaneous estimation of microphysical parameters and
756 atmospheric state with radar data and ensemble square-root Kalman filter. Part II:
757 Parameter estimation experiments. *Mon. Wea. Rev.*, **136,** 1649–1668.
758 Wang, S., M. Xue, and J. Min, 2013: A four-dimensional asynchronous ensemble square-root
759 filter (4DEnSRF) and tests with simulated radar data. *Tellus***,** In press.
760 Whitaker, J. S. and T. M. Hamill, 2002: Ensemble data assimilation without perturbed
761 observations. *Mon. Wea. Rev.*, **130,** 1913-1924.
762 Wu, B., J. Verlinde, and J. Sun, 2000: Dynamical and microphysical retrievals from Doppler
763 radar observations of a deep convective cloud. *J. Atmos. Sci.*, **57,** 262-283.
764 Xue, M., K. K. Droegemeier, and V. Wong, 2000: The Advanced Regional Prediction System
765 (ARPS) - A multiscale nonhydrostatic atmospheric simulation and prediction tool. Part I:
766 Model dynamics and verification. *Meteor. Atmos. Phy.*, **75,** 161-193.
767 Xue, M., M. Tong, and K. K. Droegemeier, 2006: An OSSE framework based on the ensemble
768 square-root Kalman filter for evaluating impact of data from radar networks on
769 thunderstorm analysis and forecast. *J. Atmos. Ocean Tech.*, **23,** 46–66.

Xue, M., K. K. Droegemeier, and D. Weber, 2007: Numerical prediction of high-impact local weather: A driver for petascale computing. *Petascale Computing: Algorithms and Applications*, Taylor & Francis Group, LLC, 103-124.

Xue, M., M. Tong, and G. Zhang, 2009: Simultaneous state estimation and attenuation correction for thunderstorms with radar data using an ensemble Kalman filter: Tests with simulated data. *Q. J. Roy. Meteor. Soc.*, **135,** 1409-1423.

Xue, M., Y. Jung, and G. Zhang, 2010: State estimation of convective storms with a two-moment microphysics scheme and an ensemble Kalman filter: Experiments with simulated radar data *Q. J. Roy. Meteor. Soc*, **136,** 685-700.

Xue, M., D.-H. Wang, J.-D. Gao, K. Brewster, and K. K. Droegemeier, 2003: The Advanced Regional Prediction System (ARPS), storm-scale numerical weather prediction and data assimilation. *Meteor. Atmos. Phy.*, **82,** 139-170.

Xue, M., K. K. Droegemeier, V. Wong, A. Shapiro, K. Brewster, F. Carr, D. Weber, Y. Liu, and D. Wang, 2001: The Advanced Regional Prediction System (ARPS) - A multi-scale nonhydrostatic atmospheric simulation and prediction tool. Part II: Model physics and applications. *Meteor. Atmos. Phys.*, **76,** 143-165.

Xue, M., K. Brewster, K. K. Droegemeier, V. Wong, D. H. Wang, F. Carr, A. Shapiro, L. M. Zhao, S. Weygandt, D. Andra, and P. Janish, 1996: The 1996 CAPS spring operational forecasting period: Realtime storm-scale NWP, Part II: Operational summary and examples. *Preprint, 11th Conf. Num. Wea. Pred.*, Norfolk, VA, Amer. Meteor. Soc., 297-300.

Xue, M., F. Kong, K. W. Thomas, Y. Wang, K. Brewster, J. Gao, X. Wang, S. J. Weiss, A. J. Clark, J. S. Kain, M. C. Coniglio, J. Du, T. L. Jensen, and Y. H. Kuo, 2011: CAPS

Realtime Storm Scale Ensemble and High Resolution Forecasts for the NOAA Hazardous Weather Testbed 2010 Spring Experiment. *24th Conf. Wea. Forecasting/20th Conf. Num. Wea. Pred.*, Amer. Meteor. Soc., Paper 9A.2.

Zhang, S., M. J. Harrison, A. T. Wittenberg, A. Rosati, J. L. Anderson, and V. Balaji, 2005: Initialization of an ENSO forecast system using a parallelized ensemble filter. *Mon. Wea. Rev.*, **133,** 3176-3201.

List of Figures

Fig. 4. RMS errors averaged over the grid points where truth reflectivity is greater than 10 dBZ and normalized by the errors of experiment OMP_F. The state variables are the 16 ARPS prognostic variables: three velocity components (u, v, and w), potential temperature (pt), pressure (p), mixing ratios of water vapor (q_v), cloud water (q_c), rain water (q_r), cloud ice (q_i), snow aggregate (q_s), hail (q_h) and their respective number concentrations (N_{tc}, N_{tr}, N_{ti}, N_{ts}, and N_{th}, associated with a two-moment microphysics scheme used).

Fig. 5. (a) The observed radar reflectivity mosaic and (b) the reflectivity field analyzed by the parallel EnKF algorithm, at model grid level 20 at 1800 UTC 10 May 2010.

Fig. 6. The model domain and coverage of 35 WSR-88D radars with 230 km range rings for the 10 May 2010 real data test case.

Fig. 7. Wall clock times of the EnKF analyses as a function of the total number of compute cores used, for the 10 May 2010 real data case in the analysis domain shown in Fig. 6, obtained on the PSC Blacklight (an SGI UV 1000). OMP denotes pure OpenMP runs, MPI denotes pure MPI runs, and H_o4, H_o8, and H_o16 denote hybrid runs with 4, 8, and 16 OpenMP threads within each MPI process, respectively. In all cases, all cores on the compute nodes were fully utilized, either by individual MPI processes or by OMP threats. Solid lines denote the total time excluding message passing, and the dashed lines show the total times including message passing. Data I/O times are excluded from all statistics.

Table 1. Timing comparisons of OpenMP experiments with MPI experiments on one compute node. Speedup for OpenMP and MPI experiments are computed relative to o1 and m01×01, respectively.

Experiment	Total number of cores used	Wall Clock Time (s)	Speedup
o1	1	6310	1.00
o2	2	3617	1.75
o4	4	2597	2.43
o6	6	1919	3.29
o8	8	1597	3.95
o12	12	1607	3.93
m01×01	1	6815	1.00
m01×02	2	3994	1.71
m02×01	2	5698	1.20
m01×04	4	2660	2.56
m02×02	4	3690	1.85
m04×01	4	2896	2.35
m03×02	6	4177	1.63
m02×04	8	2100	3.25
m04×02	8	2413	2.82

Table 2. Timing comparisons of pure MPI experiments with hybrid MPI/OpenMP experiments on 4 compute nodes. Speedup is computed relative to experiment m01×01 (6815 seconds in Table 1).

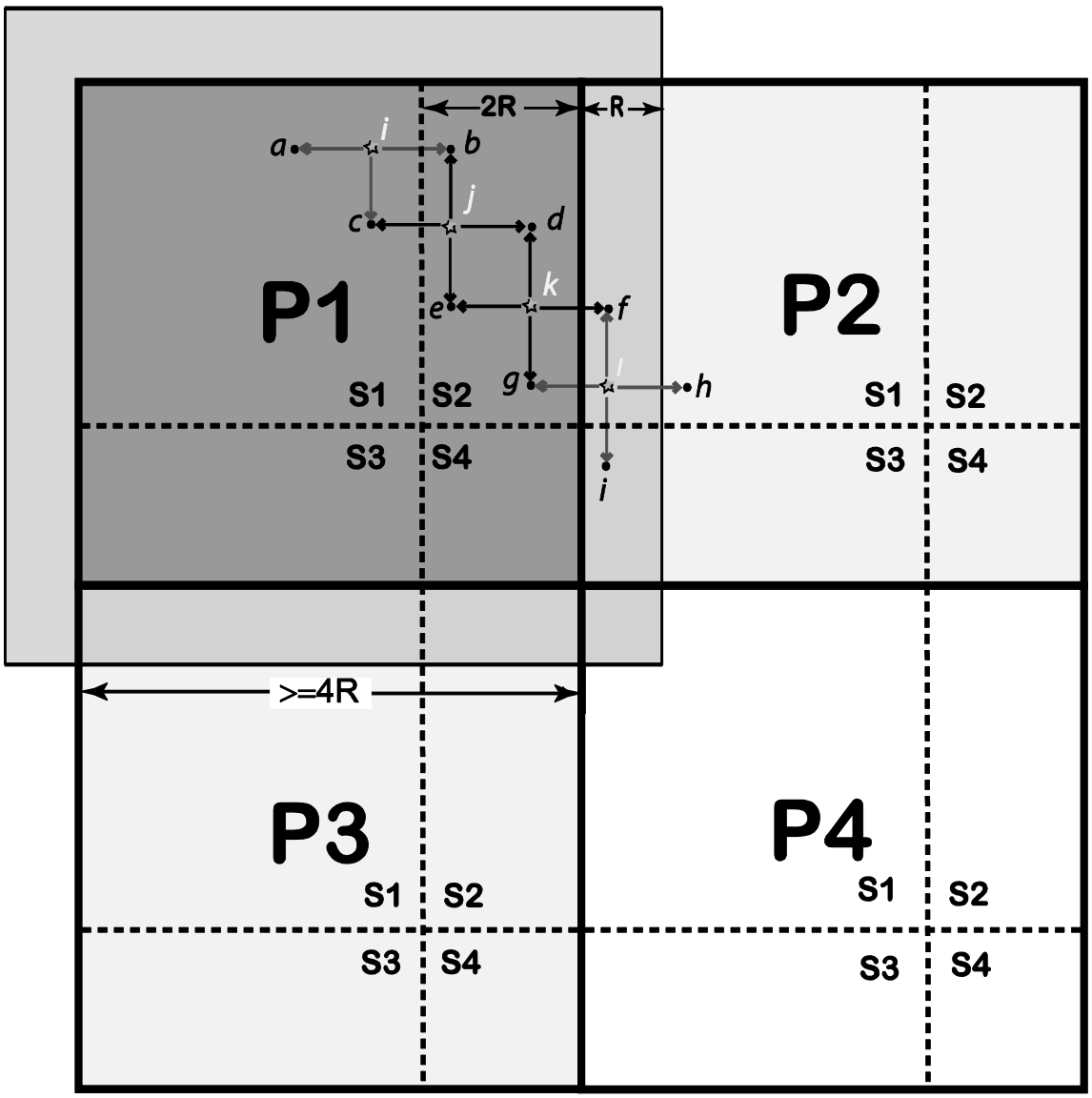
Experiment	Total number of cores used	Wall Clock Time (s)	Speedup
m01×04_01	4	2343	2.91
m02×02_01		3577	1.91
m04×01_01		2750	2.48
m02×04_02	8	2169	3.14
m04×02_02		2330	2.92
m03×04_03	12	1575	4.33
m06×02_03		1699	4.01
m04×04_04	16	1327	5.14
m02×10_05	20	915	7.45
m10×02_05		1357	5.02
m04×05_05		1082	6.30
m05×04_05		1085	6.28
m03×08_06	24	880	7.74
m06×04_06		1049	6.50
m04×10_10	40	637	10.70
m10×04_10		720	9.47
m06×08_12	48	606	11.25
h01×04_01o2	8	1471	4.63
h01×04_01o4	16	1129	6.04
h01×04_01o6	24	831	8.20
h01×04_01o8	32	772	8.83
h01×04_01o12	48	733	9.30
h02×04_02o2	16	1200	5.68
h02×04_02o4	32	908	7.51
h02×04_02o6	48	709	9.61

Table 3. Comparison of the minimum time taken in hybrid mode with that in MPI mode using the same number of cores on 4 compute nodes

Number of cores	Hybrid case	Minimum time (s)	MPI case	Minimum time (s)	Difference (s)
8	h01×04_01o2	1471	m02×04_02	2169	698
16	h01×04_01o4	1129	m04×04_04	1327	198
24	h01×04_01o6	831	m03×08_06	880	49
40	h02×10_05o2	635	m04×10_10	637	2
48	h03×08_06o2	604	m06×08_12	606	2

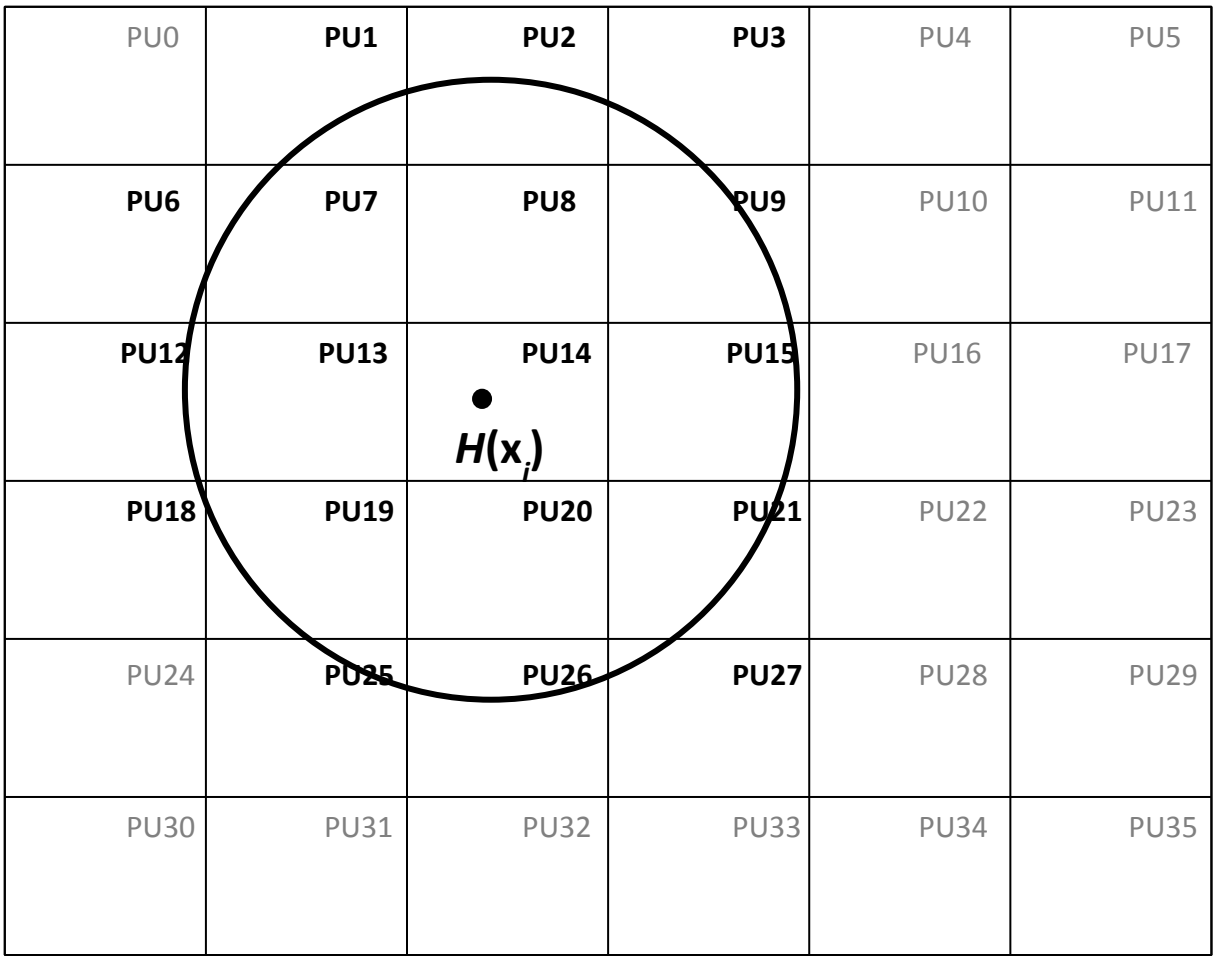
Table 4. The names and configurations of real data experiments.

Experiment		Number of PUs in *x* direction	Number of PUs in *y* direction	Number of threads per PU	Total number of Cores used
OpenMP	o16			16	16
	o32			32	32
	o64			64	64
	o80			80	80
	o160			160	160
MPI	m16	1	16		16
	m32	2	16		32
	m64	3	16		64
	m80	5	16		80
	m160	10	16		160
Hybrid Group 1	h4o4	1	4	4	16
	h8o4	1	8		32
	h16o4	2	8		64
	h20o4	2	10		80
	h40o4	4	10		160
Hybrid Group 2	h2o8	1	2	8	16
	h4o8	1	4		32
	h8o8	1	8		64
	h10o8	2	5		80
	h20o8	4	5		160
Hybrid Group 3	h2o16	1	2	16	32
	h4o16	1	4		64
	h5o16	1	5		80
	h10o16	2	5		160



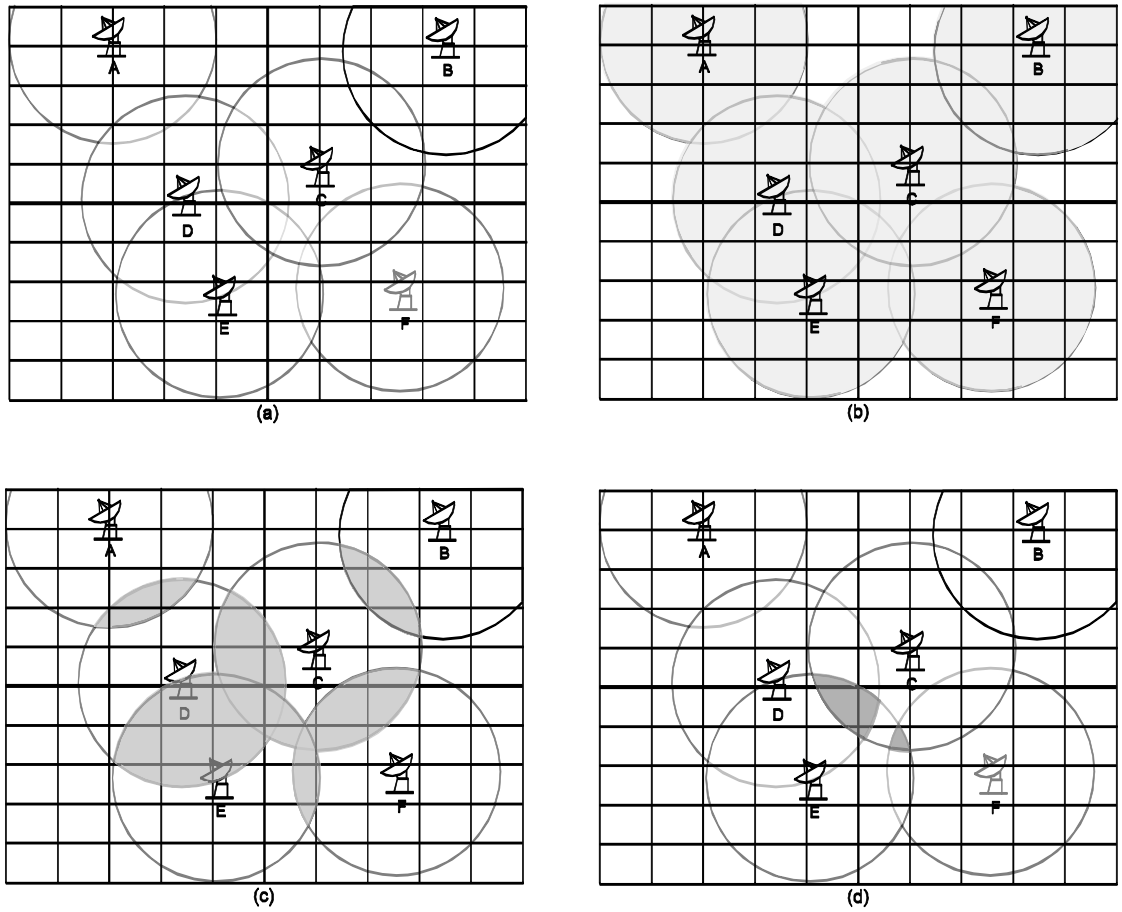
865

866 Fig. 1. A schematic of the domain decomposition strategy for the analysis of high-
867 density observations, illustrated with 4 processing units (PUs, denoted by P1
868 through P4). Letters *i*-*l* denote observations that are assumed to be equally spaced
869 and letters *a*-*h* indicate the influence limits (as determined by the covariance
870 localization radii of EnKF) of those observations. In this example, observations *i*
871 and *l* are far enough apart that they will not influence any of the same state
872 variables; they are among the observations that are analyzed simultaneously in the
873 first step of the procedure. Observations *j* and *k* are analyzed in the second step,
874 but they must be analyzed sequentially. Note that in practice, there will be many
875 more observations within patches S1 and S2 of subdomains P1 to P4 than shown
876 in the figure.



877

878 Fig. 2. A schematic for analyzing conventional data. Three steps are involved
879 when analyzing one observation whose location is denoted by a black dot in the
880 figure: 1) PU14 computes $\boldsymbol{H}(\boldsymbol{x}_i)$ (where i is the ensemble index); 2) $\boldsymbol{H}(\boldsymbol{x}_i)$ are
881 broadcasted to all PUs; 3) state variables $\boldsymbol{x}_i$ within the influence range of this
882 observation (within the large circle) are updated in parallel by the PUs that carry
883 the state variables.



884

885 Fig. 3. Composite radar data batches organized such that within each batch, no
886 more than one column of data exists for each grid column. (a) Observations from
887 six radars (A-F) with their coverage indicated by the maximum range circles are
888 remapped onto the model grid. (b) Observations of the first batch, (c) observations
889 of the second batch, and (d) observations of the third batch. If there are more
890 observations unaccounted for, additional data batch(es) will be formed.
891

892

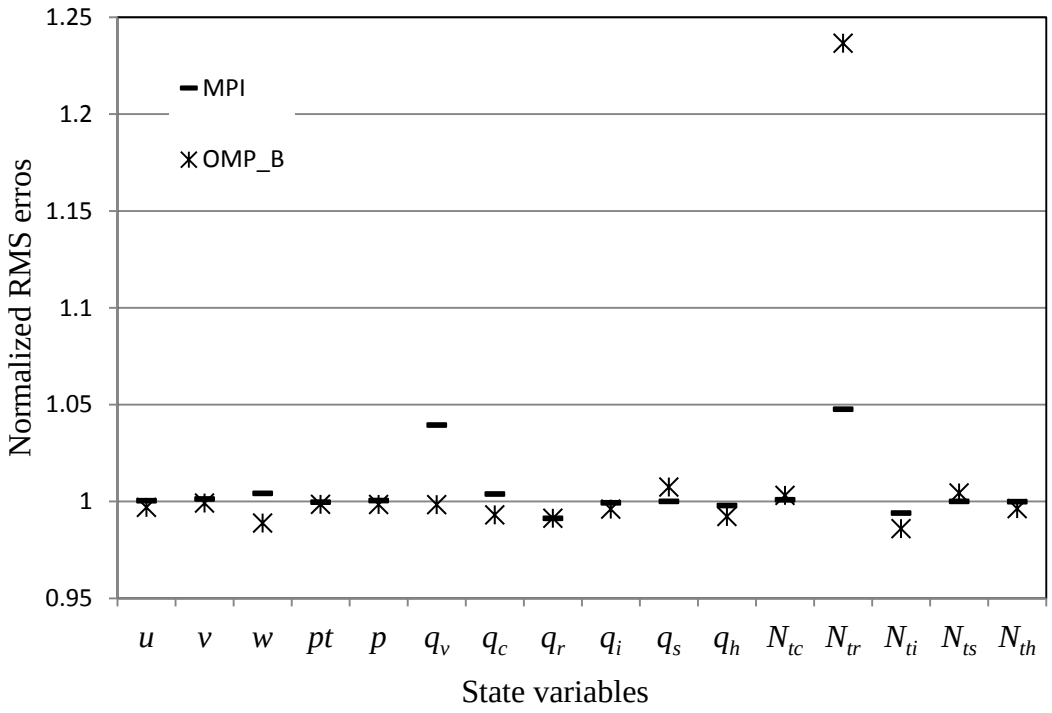
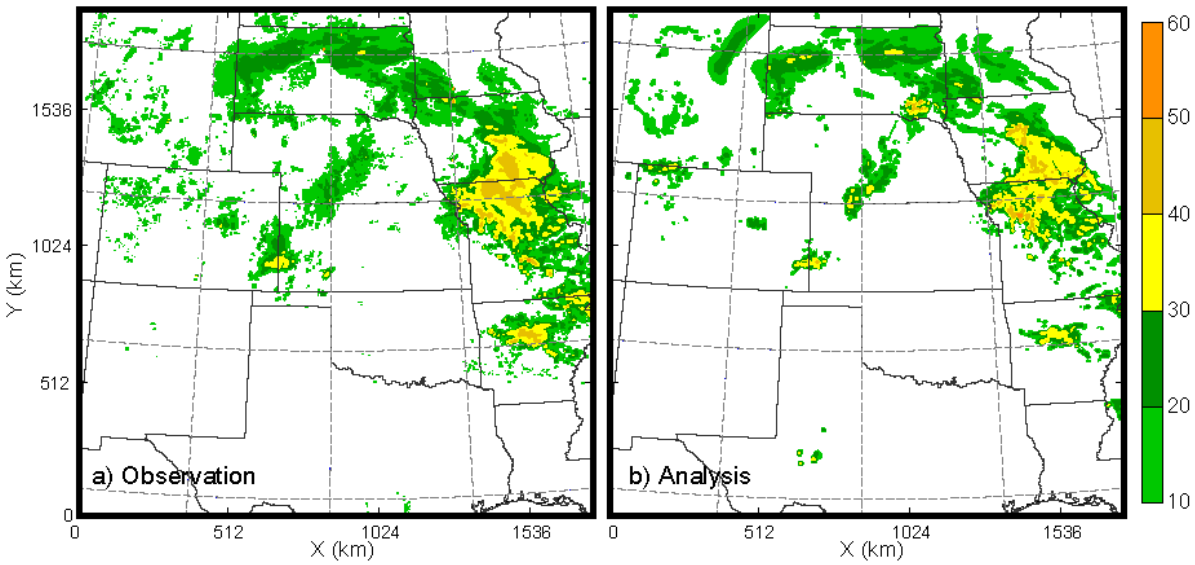


Fig. 4. RMS errors averaged over the grid points where truth reflectivity is greater than 10 dBZ and normalized by the errors of experiment OMP_F. The state variables are the 16 ARPS prognostic variables: three velocity components (u, v, and w), potential temperature (pt), pressure (p), mixing ratios of water vapor (q_v), cloud water (q_c), rain water (q_r), cloud ice (q_i), snow aggregate (q_s), hail (q_h) and their respective number concentrations (N_{tc}, N_{tr}, N_{ti}, N_{ts}, and N_{th}, associated with a two-moment microphysics scheme used).

902

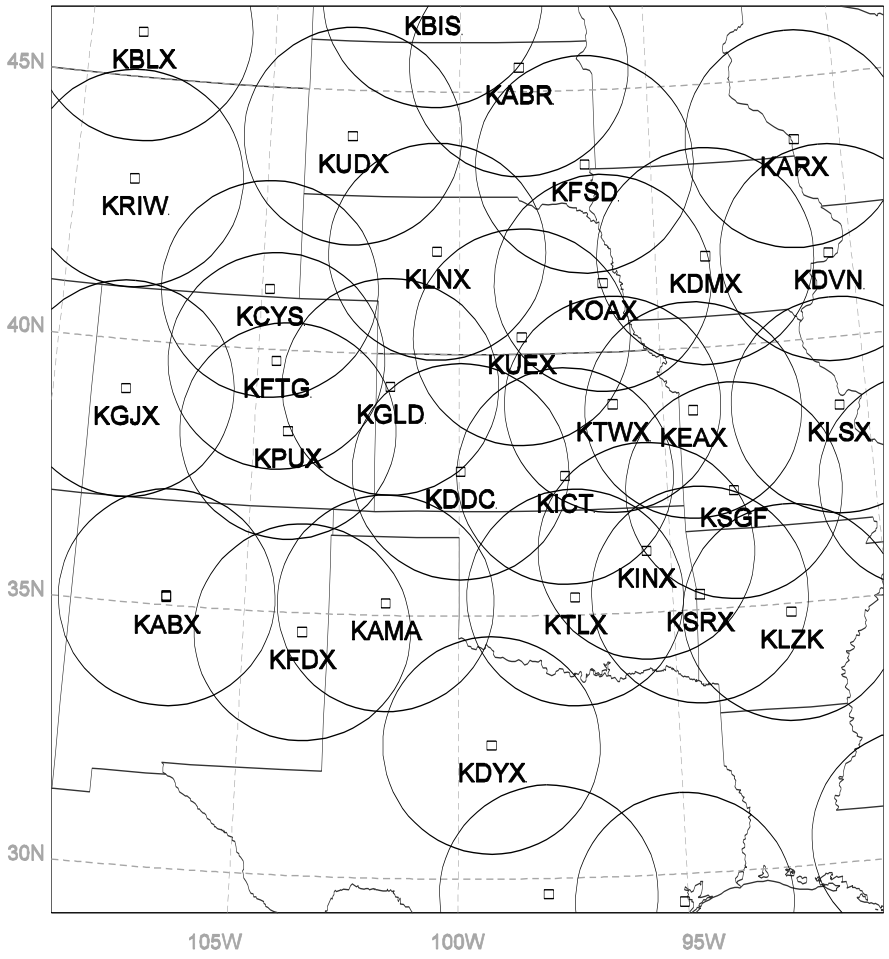


903

904 Fig. 5. (a) The observed radar reflectivity mosaic and (b) the reflectivity field
905 analyzed by the parallel EnKF algorithm, at model grid level 20 at 1800 UTC 10
906 May 2010.

907

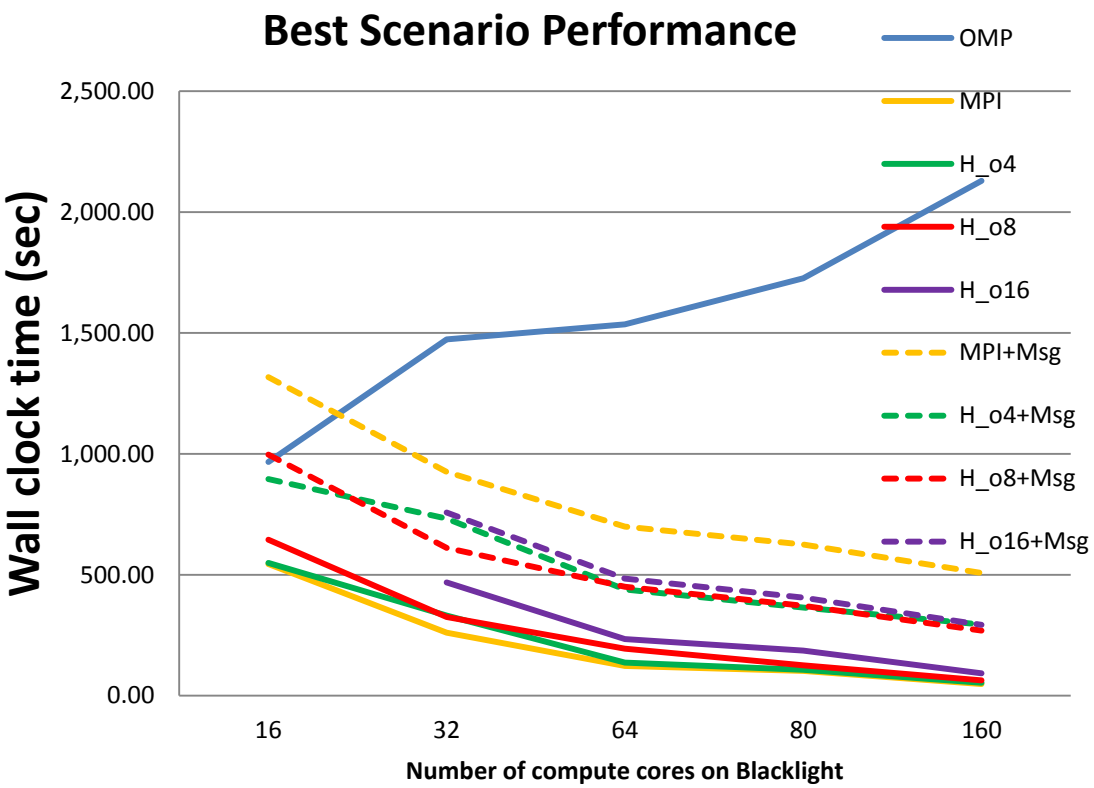
908



909

910 Fig. 6. The model domain and coverage of 35 WSR-88D radars with 230 km
911 range rings for the 10 May 2010 real data test case.

912



913

914 Fig. 7. Wall clock times of the EnKF analyses as a function of the total number of
915 compute cores used, for the 10 May 2010 real data case in the analysis domain
916 shown in Fig. 6, obtained on the PSC Blacklight (an SGI UV 1000). OMP denotes
917 pure OpenMP runs, MPI denotes pure MPI runs, and H_o4, H_o8, and H_o16
918 denote hybrid runs with 4, 8, and 16 OpenMP threads within each MPI process,
919 respectively. In all cases, all cores on the compute nodes were fully utilized, either
920 by individual MPI processes or by OMP threats. Solid lines denote the total time
921 excluding message passing, and the dashed lines show the total times including
922 message passing. Data I/O times are excluded from all statistics.