E-Infrastructures
H2020-INFRAEDI-2018-1

INFRAEDI-2-2018: Centres of Excellence on HPC

EoCoE-II

Energy oriented Center of Excellence :

toward exascale for energy

Grant Agreement Number: INFRAEDI-824158

D5.1 - Revision 5/1/2021

Melissa-DA: Architecture Specification for Large Scale Data Assimilation

Project and Deliverable Information Sheet

EoCoE	Project Ref:	INFRAEDI-824158
	Project Title:	Energy oriented Centre of Excellence
	Project Web Site:	http://www.eocoe.eu
	Deliverable ID:	D5.1 - Revision 5/1/2021
	Lead Beneficiary:	INRIA
	Contact:	Bruno Raffin
	Contact's e-mail:	Bruno.Raffin@inria.fr
	Deliverable Nature:	Report
	Dissemination Level:	PU*
	Contractual Date of Delivery:	M18 30/06/2020
	Actual Date of Delivery:	M19 17/07/2020
	EC Project Officer:	Evangelia Markidou

* - The dissemination level are indicated as follows: PU – Public, CO – Confidential, only for members of the consortium (including the Commission Services) CL – Classified, as referred to in Commission Decision 2991/844/EC.

Document Control Sheet

Document	Title :	Melissa-DA: Architecture Specification for Large Scale Data Assimilation
	ID :	D5.1 - Revision 5/1/2021
	Available at:	http://www.eocoe.eu
	Software tool:	LaTeX
Authorship	Written by:	Bruno Raffin, Sebastian Friedemann, Yen-Sen Lu, Harrie-Jan Hendricks Franssen
	Contributors:	Bibi Naz, Kai Keller, Anna Sekula, Bartłomiej Pogodziński
	Reviewed by:	Sebastian Lührs, Mathieu Lobet

Contents

List of Figures

1. Overview

The WP5 is one of EoCoE technical challenges called *Ensemble Runs*. The goal is to efficiently run large simulation ensembles on coming pre-exascale and exascale systems, in particular for data assimilation. The objective pursued here is to provide a flexible and maintainable way of executing simulation ensembles in multiple jobs on a given supercomputer while enabling communication between ensemble members to allow ensemble management and data assimilation processes. For that purpose WP5 targets developing an elastic exascale-ready framework for ensemble runs extending the Melissa approach developed at INRIA [15].

Two out of five EoCoE-II Scientific Challenges (Weather and Hydrology) integrate some support for ensemble runs for data assimilation or sensitivity analysis, usually relying on one monolithic big MPI job, like ESIAS developed during EoCoE I. WP5 relies on a novel developed framework, called Melissa-DA, to empower the Weather, and Hydrology EoCoE-II applications, enabling them to take benefit of next generation exascale machines for large ensemble runs. WP5 builds on top of WP4 (I/O) work re-using the PDI interface for data access and the FTI library for fault tolerance.

This document is the D5.1 deliverable from the EoCOE-II project from WP5. This document was initially expecting to contain an architecture specification with an early prototype code, but we actually progressed faster than expected with a sound code base and tests of data assimilation with Parflow scaling up to 1024 members on supercomputers.

WP5 developments are done by INRIA (lead and Melissa-DA core development), PSNC (core Melissa-DA development), FZJ (use cases) and CEA (PDI and Melissa-DA integration). BSC is contributing to Melissa-DA fault-tolerance mechanism with FTI (work attached to WP4).

The WP5 contribution includes as of today:

- A novel architecture specification, called Melissa-DA, that deeply modifies the original Melissa architecture to enable data assimilation at scale. The architecture retains the elastic parallel client/server model of Melissa. But the client has been revisited to become *runners* capable of handling several member executions in a time sharing mode, enabling to extend the architecture with load balancing capabilities. The server has also been redesigned to comply with the requirements of statistical data assimilation. The fault tolerance protocol has also been adapted to this novel context.
- A first functional code implementing this prototype, leveraging FTI for enabling multi-level checkpointing and accelerating restart in case of server crash.
- A toy use case based on Lorentz equation, as well as an advanced use case based on the Parflow hydrology simulation code.
- Integration of the PDAF data assimilation engine into Melissa server, enabling to support different assimilation algorithms.
- Experiments on supercomputers with Parflow simulations, EnKF assimilation, running up to 1024 members (target ensemble size for the final stage of the project).

- Identification of ambitious use cases with code and datasets for the Weather and Hydrology applications, that we will rely on for the next development and experimentation steps of WP5.

An overview of the WP5 organization follows this introduction (Sec. 2). After a quick reminder about data assimilation (Sec. 3), the developed architecture of the Melissa-DA framework for data assimilation is presented (Sec. 4), including initial integration work with developments from WP4 (PDI and FTI). This time period was also focused on identifying use cases (code and data sets) for future data assimilation tests at scale for the Hydrology and Weather applications (Sec 5 and 6). Section 7 presents results from experimentation with the hydrology use case. Pointers to code and documentation are detailed in section 8.1 before a conclusion (Sec. 9).

2. WorkPackage Organization and Management

The WP5 is structured in 3 tasks:

- Task 5.1: Ensemble Run Framework Development, M1-M30, Task leader INRIA,
- Task 5.2: Ensemble Runs for Meteo. M6-M36, Task leader: FZJ,
- Task 5.3: Ensemble Runs for Water, M6-M36, Task leader: FZJ,

with 3 deliverables:

- D5.1 - Architecture Specification and prototype codes for the framework and the 2 applications with initial documentation as well as an early usability and performance evaluation. (M18) (Report + code DEM, PU) (INRIA),
- D5.2 - M24 - First stable version of the framework code and the two applications adapted to the framework. Report will include documentation as well as performance evaluation. (M24) (Report + code DEM, PU) (FZJ),
- D5.3 - M34 - Final code release for the framework and the three applications, with the associated documentation. Report on testing at large scale. (M36) (Report + code DEM, PU) (CEA),

The work is progressing as expected. Task 5.3 is suffering a small delay as the task responsible quit and it took a few months to hire and train a new scientist (Sec 6) . But this initial delay should eventually be absorbed by the end of the project without significant impact on the outcomes. As mitigation action we focused the first experiments on the use case of Task 5.2, so that Task 5.1 and 5.2 could progress as expected. It would have been anyway difficult to address the 2 uses cases in parallel during this early stage of the prototype development.

WP5 coordination is ensure through regular global WP5 conf calls, specific WP5 meetings and join meetings with WP4 during project face-to-face meetings (WP5 has a close connection with WP4 as we make an advance use of the PDI and FTI tools). Core developments are coordinated through a weekly conf calls where other actors are involved when relevant. In addition to EoCoe-II coordination tools, we use a Gitlab for sharing code developments (Sec. 8.1) and issue tracking as well as a dedicated Slack channel for quick discussions.

3. Statistical Data Assimilation: Reminder

Data Assimilation (DA) is the method of combining model outputs and observations to create a corrected state estimate for (chaotic) systems [2, 11].

Typically a model is run until observations are available. Then these observations are used to correct the model current state. The model resumes from that new state for the next assimilation cycle. A DA problem is defined as follow:

- the model operator $\mathcal{M}$,
- an observation operator $\mathcal{H}$ that transforms model output into the space of observations,
- initial states $(x_0, t_0, \dots)$ and,
- observations y for all assimilation cycles.

Then an improved state, the analysis state x_a can be estimated by assimilating observations into the model state outputs.

There are two main branches of DA methods: variational DA methods rely on the one most probable analysis state x_a whereas statistical DA methods permit to describe the probability density function or some further uncertainty quantification of the analysis state estimate x_a. A special case of the latter are ensemble based methods, like the Ensemble Kalman Filter, where the required statistical values are estimated by running a large sample of models.

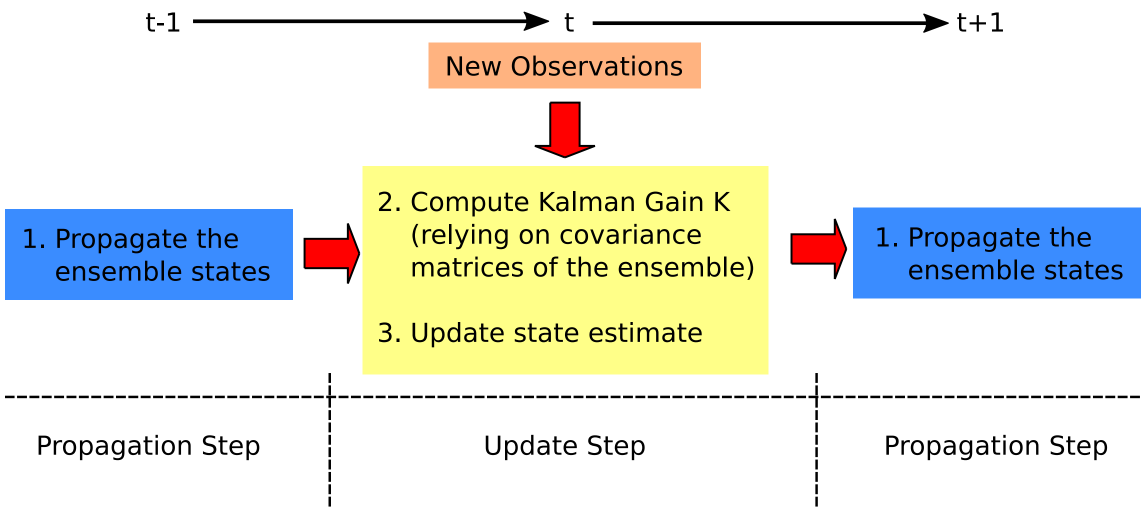
t-1
t
t+1
New Observations
1. Propagate the ensemble states
2. Compute Kalman Gain K (relying on covariance matrices of the ensemble)
3. Update state estimate
1. Propagate the ensemble states
Propagation Step
Update Step
Propagation Step

Figure 1: The Ensemble Kalman Filter

The Ensemble Kalman Filter (EnKF) is a statistical DA method that inherits the Kalman Filter [8], extending it by the possibility to deal with non linear models and non gaussian statistics [2]. To keep track of the uncertainty of the analysis state x_a an ensemble based statistical estimation is used. Calculating the covariance of the state ensemble allows to estimate model errors and to weight the last model propagation results against the recent observations y and their error R.

Figure 1 gives an overview on the EnKF algorithm:

1. An ensemble of M states x_a, statistically representing the assimilated system state, is propagated by the model $\mathcal{M}$. We denote the retained background states x_b. For the initial assimilation cycle x_a is an ensemble of perturbed states. Later it is obtained from the previous assimilation cycle.

2. The Kalman gain K is calculated using the ensemble covaraiance and the observation error R.

3. Multiply the Kalman gain K with the innovation $(y - \mathcal{H}(x_b))$ and add to the background states: $x_a = x_b + K \cdot (y - \mathcal{H}(x_b))$ to retain the new ensemble analysis states x_a.

4. Start over with the next assimilation cycle (step 1).

For a more detailed describtion of the Kalman Filter and the EnKF refer to [2].

WP5 focuses on ensemble based data assimilation. The EnKF is one of the most common DA assimilation filter, which we target in the context of the first WP5 use case (Hydrology). Target is to scale to ensembles $M = O(1000)$. The EnKF algorithm's propagation step scales linearly with the ensemble size M. Calculation of the covariance over the ensemble needs $O(M^2)$ operations governing the computational complexity of the EnKF update step [6].

4. Melissa-DA Architecture

4.1 Overview

Melissa-DA is an extension of the Melissa architecture designed for sensibility analysis [15], for enabling ensemble based data assimilation at scale. Instead of classically relying on a single application harnessing the different simulation members and the data assimilation process in a parallel monolithic MPI code, like PDAF [14], Melissa-DA relies on an elastic and fault tolerant parallel client/server architecture. Melissa-DA is the result of the work performed in the context of the EoCoE-II project.

Melissa-DA inherits from Melissa a three tier architecture (Fig. 2). The *server* gathers background states for all ensemble members. New observations are assimilated into these background states to compute a new analytical state for each member. These analytical states are distributed to the *runners* that take care of progressing the ensemble members up to the next assimilation step. The runners and the server are parallelized codes. Runners are independently launched, each one in and independent resource allocation given by the machine batch scheduler. Once started, each runner dynamically connects to the server, running in an other job. This enables for a very elastic resource usage, as the number of runners can dynamically evolve depending on the availability of compute resources. The third tier is the *launcher*. The launcher orchestrates the execution, interacting with the supercomputer batch scheduler to request resources for starting new jobs or kill jobs, monitoring job statuses, and triggering job restarts in case of failure.

The benefits of the Melissa-DA framework include:

- **Elasticity:** Melissa-DA enables the dynamic adaptation of compute resource usage according to availability. Runners are independent and connect dynamically to the parallel server when they start. They are submitted as independent jobs to the batch scheduler. Thus, the number of concurrently running runners can vary during the course of a study to adapt to the availability of compute resources.

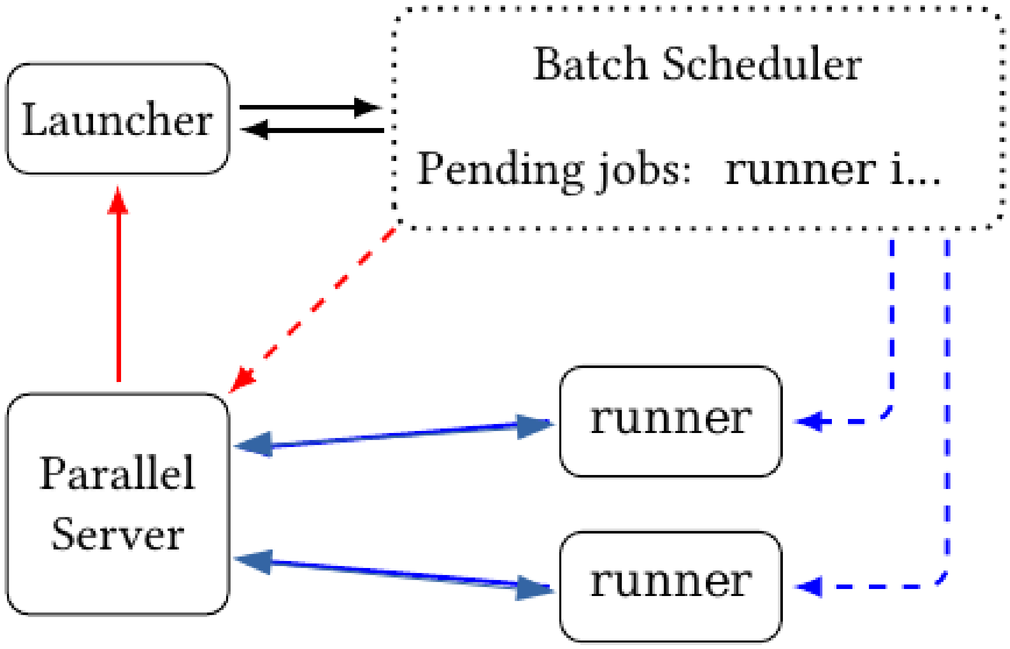

Figure 2: Melissa-DA three tier architecture overview.

- **Fault tolerance:** Melissa-DA asynchronous client/server architecture supports a simple yet robust fault tolerance mechanism. Only some lightweight bookkeeping and a few heartbeats as well as FTI multi-level checkpointing on the server side are required to detect issues and restart the server or the runners.
- **Load Balancing:** The distribution of member states to runners is controlled by the server and defined dynamically, enabling to adjust the load of each runner according to the time required to propagate each member.
- **Communication/Computation Overlaping:** communications between the server and the runners occur asynchronously, in parallel with computation, enabling an effective overlapping between computation and communication, improving the overall execution efficiency.

4.2 Unbalanced Propagation Time

The time taken for propagating a given analytical state may vary significantly [4] causing inefficiencies when performed in parallel. The server has to wait for the slowest member to return its background state before being able to proceed with the data assimilation process to compute the analytical states. Worst case occurs when state propagation is fully parallel, i.e. when each runner is in charge of a single member. In that case runners idle time is the sum of the differences between each propagation time and the slowest one. As we target large number of members, each member potentially being a large scale parallel simulation, this can account for significant resource under utilization. To reduce this source of inefficiency Melissa-DA enables 1) to control the propagation concurrency level independently from the number of members 2) to distribute dynamically members to runners.

4.3 Members' States

We split a simulation state into three sub-categories: the *static state*, the *moving state* and the *assimilated state*. The static state encompasses all simulation variables that are defined at start-time, do not change during the execution, and are common to all possible members. This may be the mesh topology used to discretize the simulation space, given that this mesh topology does not change during the simulation execution and that all members use the same mesh. The union of the static and moving state define the full state of an application at a given point in time. The assimilated state is the sub-part of the moving state the data assimilation process is concerned with. Given a running simulation, switching from one member state to an other one, it only requires to overwrite the current moving state with the new moving state, including the assimilated state.

4.4 Server

At every assimilation cycle the server collects all the ensemble member moving states from the runners (Fig. 3). The member assimilated states are used by the server to compute the next analytical states. Because the server also holds the full moving states, the server can dynamically control the distribution of members to runners, enabling load balancing, adaptation to potentially varying number of runners. Checkpointing the server is also sufficient to ensure a proper restart in case of runner or server crash.

The server is parallel (based on MPI) and runs on several nodes. The number of nodes required for the server is first guided by memory needs. The amount of memory needed is in the order of the sum of the member's moving states. The current server embeds PDAF as parallel assimilation engine. The server parallelization can be chosen independently from the member parallelization. A N $\times$ M data redistribution takes place between each runner and the server to account for different levels of parallelism on the server and runner side. This redistribution scheme is implemented on top of ZeroMQ, a connection library extending sockets. This library supports a client/server connection scheme allowing to dynamically add or remove runners.

Care must be taken to store coherent state parts as they might not be received by all server ranks in the same order: runners are not synchronized together. For instance server rank 0 could receive a part of the member 3 moving state, while rank 1 receives a part of member 4 moving state. Even more importantly the state parts that are sent back must be synchronized so that the ranks of one runner receive the parts of the same ensemble member's moving state from all the connected server ranks. For that purpose all received state parts are labeled with the ensemble member id they belong to, enabling the server to assemble coherently distributed member states. State propagation is ensured by the server rank 0, the only one making decisions on which runner propagates which ensemble member. This decision is next shared amongst all the server ranks using non blocking MPI broadcasts.

Notice that the server actually only needs the assimilated states for computing the new analysis states. Gathering the full moving states on the server is a commodity for load balancing and fault tolerance. Future work will investigate more advanced strategies to reduce the transfers of moving states.

Depending on the complexity of the simulation, it may be difficult to properly save the moving state. In that case Melissa-DA loses some flexibility but can still operate. Each

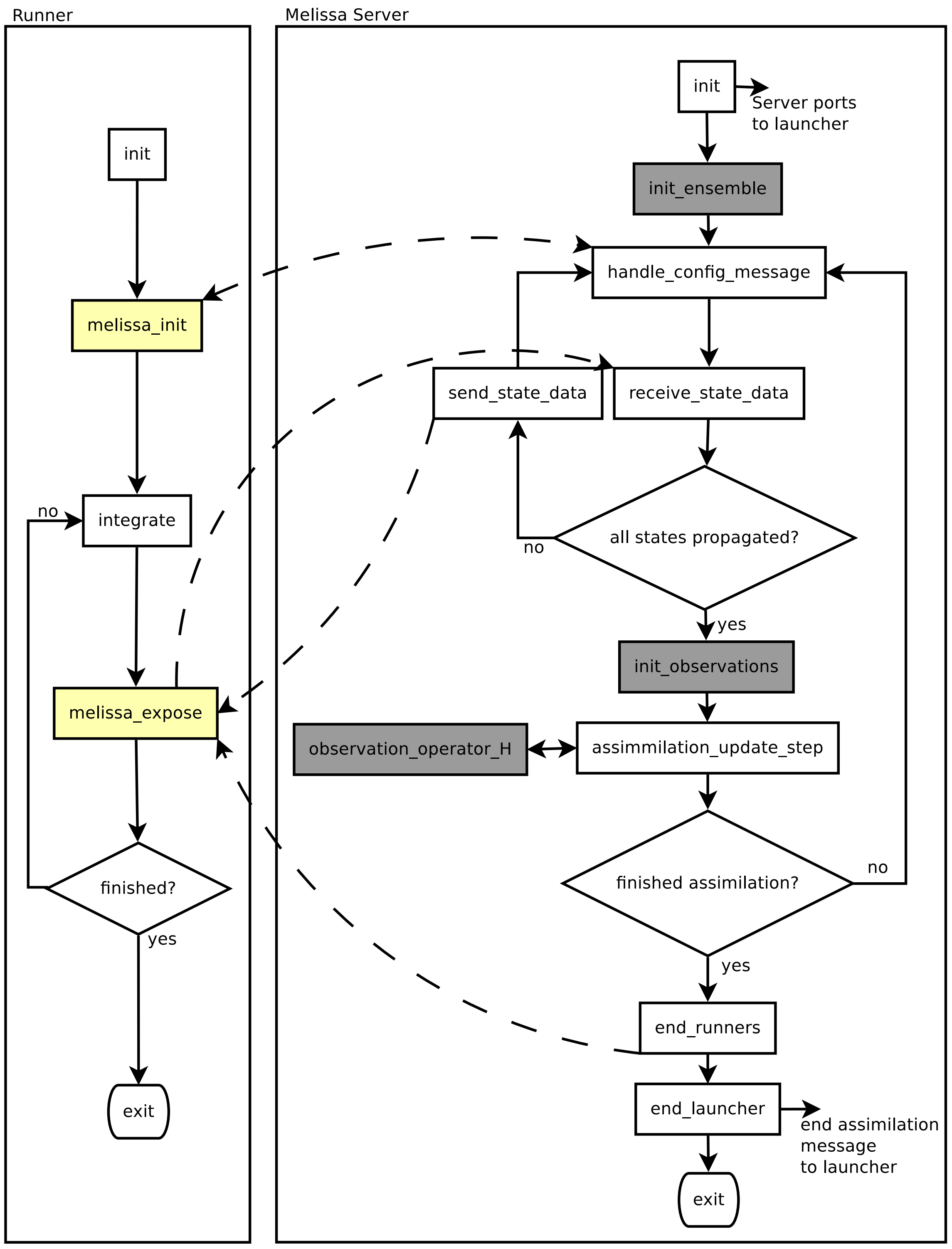

Figure 3: Melissa-DA runner and server interactions (fault tolerance part omitted for sake of clarity)

runner is assigned a unique member and only the assimilated states are exchanged.

4.5 Runners

A Melissa-DA runner is based on the simulation code and maintains in memory only the state of one given member. An existing code is turned into a Melissa-DA runner using the minimalistic Melissa-DA API. This API consists only of two functions: `melissa_init` and `melissa_expose`. `melissa_init` must be called once at the beginning to define the size of the moving state per simulation rank. Then `melissa_init` exchanges this information with the server, retrieving the server parallelization level. Thus the $N \times M$ communication scheme is computed on the server and the runner side. Eventually `melissa_init` opens all the necessary connections to the different server ranks. `melissa_expose` needs to be inserted into the simulation code to enable the update of the runners state to represent the member's analysis state as received from the server. When called, this function is given a pointer to the moving state data that is sent to the server, and next waits to receive from the server the assimilated state to be used to update the current simulation state. The function `melissa_expose` returns the amount of timesteps the received state needs to be propagated or a stop signal.

4.6 Launcher

The launcher orchestrates the full application. The Launcher typically runs on the supercomputer front node. The launcher requests the batch scheduler to start the server and the clients. It first submits to the batch scheduler a job for the server. Then, the launcher retrieves the server node addresses and submits the runners jobs.

The user only interacts with the launcher to start, stop the application, retrieve data about the progress. A Jupyter notebook can connect to the launcher for easing control and monitoring.

The launcher is the main actor of Melissa fault tolerance mechanism. It monitors job status, receives heartbeats from the server, and is able to resubmit the server or the simulation jobs if needed. The launcher Is the single point of failure If the launcher crashes or is killed by the user, the full application stops.

The launcher is in charge of adapting resource usage, by adding or removing runners. When the server notifies that the assimilation finished, the launcher takes care of killing all runner jobs.

4.7 Fault Tolerance

Melissa-DA supports detection and recovery from failures (including straggler issues) of the server and runners, through timeouts, heartbeats and server check-pointing. The server is checkpointed using the multi-level checkpointing library FTI (part of WP4). The integration of FTI in the Melissa-DA server is detailed in Section 6.2 of Deliverable D4.1. Checkpointing is performed once per assimilation cycle. Because the server stores the moving state of the different members, no checkpointing is required on the runners, lowering the needs on the simulation side. If the simulation comes with checkpointing capabilities, it can be leveraged to speed-up runner restart.

The server sends heartbeats to the launcher. If missing the launcher kills the runners and server jobs, restarts the server from the last checkpoint and next runners.

The server is in charge of tracking runners activity based on timeouts. If a runner is detected as failing, the server re-assigns state propagation to other active runners. More precisely, if one of the server ranks detects a timeout from a runner, it notifies the server rank 0 who then reschedules this ensemble member to a different runner, informing all server ranks to discard information already received by the crashed runner. Further the server sends stop messages to all other ranks of the failing runner. So the launcher recognizes that the runner job crashed and restarts a new one that will connect to the server as soon as ready.

The most common faults we experienced are jobs being canceled by the batch scheduler once reaching the limit wall-time and numerical errors. In these cases the fault tolerance protocol enables to keep the application running smoothly. A pitfall that is for the moment not fully addressed is the case of errors that cannot be solved by a restart, typically numerical errors. The current fault protocol is not able to distinguish such situation and take corrective actions. This can lead to multiple restarts and an application that never ends causing useless resource consumption. We will investigate mitigation actions in the next steps.

Testing the fault-tolerance protocol is tricky, in particular as it requires a distributed execution. We are currently developing a test suite integrated in the CI process relying on a virtual cluster built from containers (developed by the OAR team (`https://oar.imag.fr/`). We currently support tests with the OAR and Slurm batch schedulers.

4.8 PDI Support

To further simplify the integration of existing simulations in Melissa-DA, we plan to develop a Melissa-DA plugin for PDI (WP4). Application developers will then only have to instrument their code using the PDI interface that they already use for regular I/Os, all Melissa-DA specific code being hidden in the PDI plugin. PDI will also be used for the server I/Os. A first prototype is under development.

4.9 Code

The code of Melissa-DA (server and API) is written in C++ relying on features introduced with cpp14. This is especially handy regarding smart pointers to avoid memory leaks and having access to different containers (sets, lists, maps) used to store scheduling mappings. The assimilation update step is contained in its own class, which accesses the received moving states and creates a new set of analyzed states. The assimilation process on the server side is currently implemented using the parallel code from the PDAF assimilation engine. This enables to Melissa-DA to support the various assimilation algorithms from PDAF (EnKF, LETKF… See `http://pdaf.awi.de/trac/wiki/FeaturesofPdaf` for a full list).

5. Hydrology Scientific Challenge

5.1 Motivation

Predictions with terrestrial system models which include the land surface and subsurface are strongly affected by uncertainty, which is related to the strong heterogeneity of the land surface and the subsurface and the lack of high quality data. In order to reduce uncertainty with terrestrial system models and improve predictions which include groundwater, soil, vegetation and surface water, the integration of model data with data assimilation is important. Data assimilation is compute intensive, because a large number of model runs is needed to characterize model uncertainty. For applications with integrated terrestrial system models like TSMP, typically O(100) ensemble members are processed using Ensemble based Kalman filter (EnKF) algorithms. However, from earlier studies it is known that the performance of the EnKF is still suboptimal if 100 ensemble members are used, especially if together with the states also parameters are updated. This is related to the fact that smaller ensemble sizes result in less accurate estimation of model covariances, and this not only results in a suboptimal weighting of simulated values versus measured values, but also in an artificial reduction of model uncertainty. This on turn affects subsequent data assimilation steps and in particular the weighting of simulated values versus measured values, giving too much weight to simulated values so that measured values have too little corrective influence on the model runs. Numerical experiments suggest that O(500) and maybe even O(1000) members are needed for EnKF to achieve a satisfying performance. As we also need to use a high spatial resolution to adequately model hydrological processes, and in case of large spatial domains, it is not feasible to use such a large number of ensemble members. A specific problem, which reduces computation efficiency, is load imbalance. This is related to the strongly heterogeneous spatial distribution of parameters like saturated hydraulic conductivity. The differences in parameter values between individual ensemble runs in combination with the non-linearity of the equations (especially for drying out of the upper soil layer and generation of overland flow) result in very different compute times for individual ensemble members. As a result, the progress of the data assimilation system can be strongly reduced by the slowest ensemble members. Tackling the load imbalance issue is therefore paramount for increasing the computation efficiency for data assimilation studies.

5.2 Target Use Case and Progress

Data assimilation with integrated terrestrial system models is very compute intensive because it is necessary to calculate at a very high spatial resolution to represent processes like infiltration and surface runoff, and perform a large number of model runs. Load imbalance is a serious problem because individual ensemble members require very different amounts of compute time.

TSMP [10] is an advanced coupled code relying on ParFlow (subsurface), CLM (land surface) and COSMO-DE (atmosphere). Current Melissa-DA tests rely on ParFlow only (Sec. 7) as handling a coupled code will require significant effort, in particular to properly identify a coherent moving state split in between the 3 codes. This work is part of next step. The developed integration of Melissa-DA into TSMP-PDAF [10] will be finally tested for a very compute intensive test case. This is the land surface-subsurface model for the Neckar catchment and surroundings in Southwestern Germany. Currently, even for a relatively coarse model of 800m resolution, available compute time on supercomputers does not allow for a larger ensemble size than 64 members and runs of one year. It is expected that this affects the performance of the data assimilation, especially if also parameters are estimated. We aim to increase the ensemble size for this problem to 1024 members and demonstrate the increased efficiency of the TSMP-PDAF-Melissa-DA framework. Such an increase, and also a smaller increase, would be of tremendous advantage for data assimilation applications in combination with terrestrial system models and Earth system models. This would also allow to do ensemble runs with a higher spatial resolution of for example 400m instead of 800m, but this is not planned in the context of the target use case.

6. Weather Scientific Challenge

6.1 Motivation

The renewable energy sources to the electricity grids of European countries highly depend on the variable weather that challenges to the existing energy systems in many ways. Therefore, the resilient power grid and power plant management together with available trading at power stock exchanges are two known concerns, which rely on the predictability of wind and solar energy. The challenge will be delivering continuous probabilistic short term forecasts with ultra-large ensemble size with $O(1000)$ model runs to yield probability density functions (pdfs) for wind and clouds. Besides the large ensemble runs, the sufficient spatial resolution (i.e. 1km) also requires exascale capability. The weather use case employs the flagship codes Ensemble for Stochastic Interpolation of Atmospheric Simulations (ESIAS), including two satellite codes, the ensemble version of Weather Research and Forecast (WRF) model [4] and the EURopean Air pollution Dispersion-Inverse Model (EURAD-IM) [5] to predict winds at 100m of rotor hub heights, cloud optical thickness (COT), and the aerosol induced turbidity (aerosol optical thickness, AOT). These tools will be applied to the power management systems for selected site location and cloud tracking systems. Based on developments in the EoCoE project, the stochastic integration of an ultra-large ensemble by ESIAS can produce pdfs to provide reliable forecasting for the electricity system.

However, the current ESIAS framework is based on hardwired modification of codes, and therefore the difficulty increases when upgrading those satellite softwares. Integrating Melissa-DA framework becomes essential to lessen the effort on software engineering, and also to increase the elasticity on ensemble forecasting. Moreover, a particle filter data assimilation scheme will be employed in the ESIAS framework to ensure the weighting of ensemble member by importance sampling.

6.2 Target Use Case and Progress

The target use case will focus on the European domain and on the stochastic weather forecasting for wind at 100m, COT, and AOT. As shown in Figure 4, the most course spatial resolution is 20 km for the European domain, with 4 km and 1 km for the nested domains that focus on Germany and Nordrhein-Westfalen State of Germany, respectively. These nested domains can perform not only the finer spatial resolution for greater precision but also the ability for large ensemble simulation. An integration of particle filter scheme will be used for weighting the resulting ensemble members and to select ensemble members by importance sampling.

The integration of Melissa-DA and ESIAS is highly related to WP1 and WP4 for science challenge and Parallel Data Interface (PDI) integration, respectively. In WP1, the integration of Melissa-DA and ESIAS can aid to support elastic exascale-ready framework for large ensemble runs. Melissa-DA will leverage the integration of PDI into ESIAS to minimize code intrusion as it will enable to avoid the presence of any Melissa-DA specific API calls in ESIAS. The Melissa-DA API calls will be hidden in the Melissa-DA plugin for PDI, which will be developed in cooperation between WP4 and 5.

Due to the delay on hiring process in M1-M12 and the lockdowns during the COVID-19 crisis, the preparation of integration work of ESIAS and Melissa-DA has been postponed and started recently only. This should not affect the schedule as the current Melissa-DA prototype is anyway not yet ready for supporting ESIAS. Currently the framework of ESIAS-met reanalysis and the Python codes for executing ESIAS-met are prepared to support Melissa-DA [1]

[1]the preprocessing Python codes can be found on `https://jugit.fz-juelich.de/ye.lu/esias-met/-/tree/master/Preprocessing_Tools`

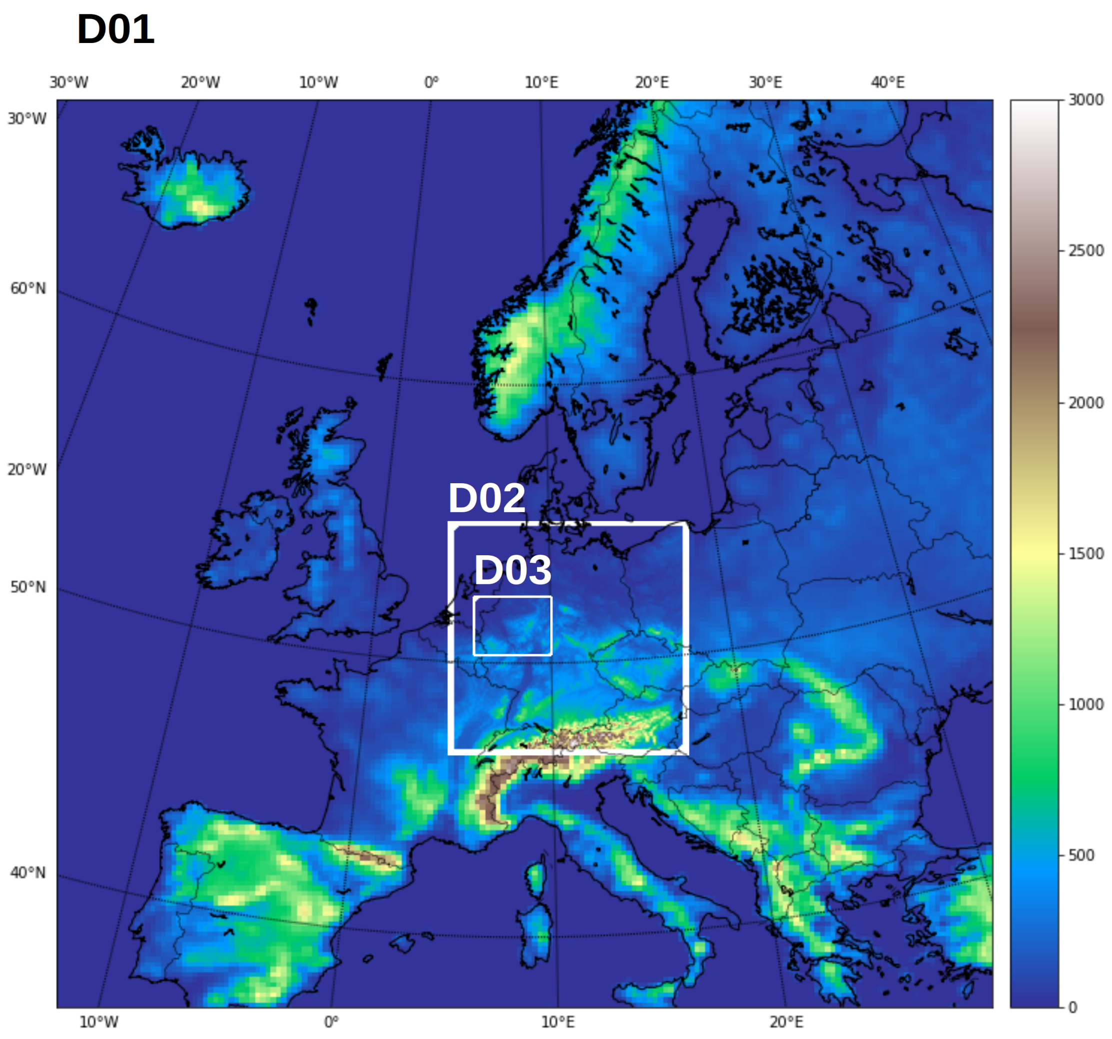

Figure 4: The model domain for ESIAS-met. D01, D02, and D03 are the European domain, German Domain, and Nordrhein-Westfalen domain, respectively

7. Early Experiments

7.1 Setup

First experiments with Melissa-DA were performed on the Jean-Zay supercomputer (France). Jean-Zay is a HPE-SGI8600 machine of 1528 compute nodes, dual-processor Intel Cascade Lake 6248 with 20 cores running at 2,5 GHz with 192GB DDR4-2667 memory per node. The compute nodes are connected through an Omni-Path interconnection network with a bandwidth of 100 Gb/s. Hyperthreading was used on this machine, pinning 40 MPI ranks to each node.

For the sake of simplicity and minimal intrusion, the code was instrumented by calls to the STL-chrono [1] library allowing a precision of a nanosecond. This minimal instrumentation was successfully validated against automatic score-P instrumentation and scalasca.

7.2 Use Case

Tests rely on the Hydrology use case (Sec 5), using the Parflow simulation code, a physically based, fully coupled water transfer model for the critical zone that relies on an iterative Krylov-Newton solver [3, 7, 13, 9, 12]. This solver performs a changing amount of iterations until a defined convergence tolerance is reached each time step, making the time for state propagation very variable [4].

Experiments assimilating Parflow simulations use an assimilated state of 4031700 cells (pressure). The whole moving state three times as big containing saturation and density vectors in addition.

7.3 Melissa-DA Performance

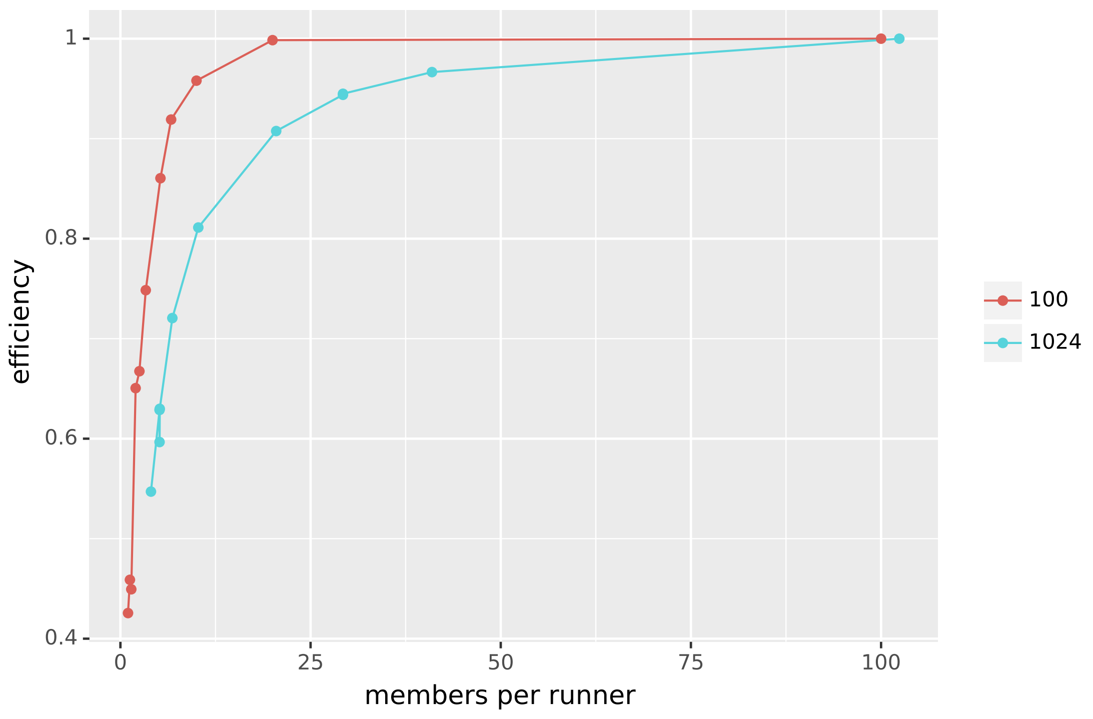

Figure 5: Scaling efficiency (compared to the case with ≈ 100 members per runner) regarding runners while assimilating 25 observations into about 4 M grid cells with 100 or 1024 ensemble members

Early experiments focused on getting insight on the behavior of Melissa-DA depending on the member per runner ratio. The server runs on 1 node or 3 nodes to have enough memory to store the 100 or 1024 ensemble member's moving states respectively. Each runner runs on the 40 threads of 1 node. We tested different member/runner ratios for 100 and 1024 members. Members are distributed to runners dynamically one-by-one each time the server receives a request for a member from a runner. We plot the efficiency for each case (Figure 5). Efficiency is maximum when a single runner executes all members as in that case runner idle time is reduced to a minimum (basically communication time and assimilation time). We only ran this configuration for 100 members, as it would take too many hours for 1024 members. Efficiency is worst when each runner gets one member, as it maximizes runner idle time. The 90% efficiency limit is reached when having 15 runners for 100 members (av. of $6\frac{2}{3}$ members per runner) and 20 runners for 1024 members (av. of 51.2 members per runner). The speed-up is respectively about $14\times$ and $18\times$. The efficiency then quickly drops as the member to runner ratio decreases (or number of runners increases). Maintaining a 90% efficiency at 1024 members strongly limits the parallelization. We have here a significant margin for improvements. The server is very likely the bottleneck. Indeed, the server can become the bottleneck as the number of members to proceed increases, requiring to allocate more nodes to the server. On-going experiments are focusing on identifying the sources of this bottleneck (communications, co-variance matrix or Kalman gain computation, etc.) to better understand how to calibrate the server size. These are very early results that need to be complemented with more experiments and more detailed performance analysis.

8. Code Availability and Scalability Targets (Added as of Revision 1/1/2021)

This section was added following the request for revision of the 3/12/2020. Section 8.1 details the code management process. Section 8.2 details the scalability targets for WP5, complementing the Table 1 in D7.2.

8.1 Code Management and Availability

- Melissa is a framework for in-transit data processing for large scale ensemble runs. Former work focused on sensibility analysis. EoCoE-II work extends Melissa to data assimilation.
- Licence: BSD-3 (enable copying, code modification, inclusion in other code without contamination, and commercial exploitation)
- Melissa public repository (accessible to all):
  - Code repository: `https://github.com/melissa-sa/melissa`
  - Web Page: `https://melissa-sa.github.io`
- Melissa work repository (access on invitation/request only):
  - `https://gitlab.inria.fr/melissa/melissa`
  - This work directory contains various prototype developments not yet ready to be public. As soon as these developments are properly validated, backed with a publication for major contributions, and documented, they are pushed to the public repository on GitHub.
  - Current prototype of Melissa for data assimilation is part of this repository. As planed in the work program, a first version of the code will be made available on the public repository at M24 and a second one at M36. These releases will come with proper support for the TerrSysMP and ESIAS use cases.

8.2 Scability Targets

Note that a one-to-one direct comparison between Pre-EoCoE-II and Melissa based EoCoE-II target runs should be made very carefully as the numbers given bellow are focused on scalability only and do not reflect the gains in efficiency and resilience. *Hero run* denotes a large scale run pushing the scalability in number of members of the system. *Production run* denote a realistic run involving coupled codes and full observation datasets.

Meteo - ESIAS (WRF+EURAD-IM)

- Pre-EoCoE-II status: production runs with 4096 ensemble with 262 144 CPU cores
- **EoCoE-II targets:**
  - Hero run (WRF): 15 000 members on 40 000 CPU cores. **Status: planned**
  - Production run :
    - Target: analysis of a mineral dust event limiting the solar power production in a water-cloud free sky coupling WRF and EURAD-IM within the Melissa
    - 512 members, 65 000 CPU cores, JUWELS machine
    - **Status: planned**

Water - TerrSysMP (Parflow/CLM)

- Pre-EoCOE-II status:
  - Hero run (Parflow, CLM and TSMP-PDAF): 256 members on 132 768 CPU cores
  - Production runs :
    - Neckar use-case (Parflow, CLM and TSMP-PDAF): 64 ensemble members on 4 608 CPU cores.
    - Europe use-case (CLM and TSMP-PDAF): 20 ensemble members on 1 920 CPU cores.
- **EoCoE-II targets**:
  - Hero run (Parflow, Melissa): 15 000 members on 40 000 CPU cores. **Status: in progress**
  - Production runs:
    - Neckar use-case (Parflow, CLM and Melissa)
      - 512 members on 36 864 CPU cores, JUWELS machine
      - **Status: planned**
    - Europe use-case (Parflow, CLM and Melissa):
      - 512 members on 36 864 CPU cores, JUWELS machine
      - **Status: planned**

9. Summary and Perspectives

We full-filled a first design of Melissa-DA architecture and have a running prototype that integrates the PDAF data assimilation engine and a fault-tolerance protocol leveraging the FTI library. The current prototype supports the PDAF hydrology code and the EnKF assimilation method. Early results already demonstrate the benefits of having a modular, elastic and fault tolerant architecture for large scale data assimilation. It has been tested on two supercomputers (Juwels and Jean-Zay). Initial experiments enabled us to scale to 1024 member with the Parflow application, a target that we expected to reach at the end of the project only.

Future work will focus on:

- Extensive testing and performance evaluation of current implementation ;
- Refining the implementation (fault tolerance protocol, PDI support) ;
- Publication presenting the current Melissa-DA architecture and initial results ;
- Documentation, tutorials and examples ;
- Further performance improvement strategies (load balancing algorithms, state migration minimization) ;
- Extend the Parflow use case with the TSMP coupled code ;
- Integration of the ESIAS application with an assimilation process based on a particle filter algorithm.

10. Acknowledgments

This work was granted access to the HPC resources of IDRIS under the allocation 2020-A8 A0080610366 attributed by GENCI (Grand Equipement National de Calcul Intensif).

References

[1] Date and time utilities - cppreference.com.

[2] Mark Asch, Marc Bocquet, and Maëlle Nodet. *Data assimilation: methods, algorithms, and applications*, volume 11. SIAM, 2016.

[3] Steven F. Ashby and Robert D. Falgout. A Parallel Multigrid Preconditioned Conjugate Gradient Algorithm for Groundwater Flow Simulations. *Nuclear Science and Engineering*, 124(1):145–159, September 1996.

[4] Jonas Berndt. *On the predictability of exceptional error events in wind power forecasting—an ultra large ensemble approach—*. PhD thesis, Universität zu Köln, 2018.

[5] Philipp Franke. *Quantitative estimation of unexpected emissions in the atmosphere by stochastic inversion techniques*. PhD thesis, Universität zu Köln, 2018.

[6] Hojat Ghorbanidehno, Amalia Kokkinaki, Judith Yue Li, Eric Darve, and Peter K. Kitanidis. Real-time data assimilation for large-scale systems: The spectral Kalman filter. *Advances in Water Resources*, 86:260–272, December 2015.

[7] Jim E. Jones and Carol S. Woodward. Newton–Krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems. *Advances in Water Resources*, 24(7):763–774, July 2001.

[8] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[9] Stefan J. Kollet and Reed M. Maxwell. Capturing the influence of groundwater dynamics on land surface processes using an integrated, distributed watershed model. *Water Resources Research*, 44(2):W02402, February 2008.

[10] Wolfgang Kurtz, Guowei He, Stefan J Kollet, Reed M Maxwell, Harry Vereecken, and Harrie-Jan Hendricks Franssen. Terrsysmp-pdaf (version 1.0): a modular high-performance data assimilation framework for an integrated land surface-subsurface model. *Geoscientific Model Development*, 9(4), 2016.

[11] Edward N. Lorenz. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, March 1963. Publisher: American Meteorological Society.

[12] Reed M. Maxwell. A terrain-following grid transform and preconditioner for parallel, large-scale, integrated hydrologic modeling. *Advances in Water Resources*, 53:109–117, March 2013.

[13] Reed M. Maxwell and Norman L. Miller. Development of a Coupled Land Surface and Groundwater Model. *Journal of Hydrometeorology*, 6(3):233–247, June 2005.

[14] Lars Nerger and Wolfgang Hiller. Software for ensemble-based data assimilation systems—implementation strategies and scalability. *Computers & Geosciences*, 55:110–118, 2013.

[15] Théophile Terraz, Alejandro Ribes, Yvan Fournier, Bertrand Iooss, and Bruno Raffin. Melissa: Large scale in transit sensitivity analysis avoiding intermediate files. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'17)*, Denver, 2017.