Many Task Computing for Real-Time Uncertainty Prediction and Data Assimilation in the Ocean

Evangelinos, C., P.F.J. Lermusiaux, J.Xu, P.J. Haley, and C.N. Hill, 2010. Many Task Computing for Real-Time Uncertainty Prediction and Data Assimilation in the Ocean. IEEE Transactions on Parallel and Distributed Systems, Special Issue on Many-Task Computing, I. Foster, I. Raicu and Y. Zhao (Guest Eds.), In press.

Many Task Computing for Real-Time Uncertainty Prediction and Data Assimilation in the Ocean

Constantinos Evangelinos, *Member, IEEE,* Pierre F. J. Lermusiaux, *Member, IEEE,* Jinshan Xu, Patrick J. Haley Jr., and Chris N. Hill

Abstract—Uncertainty prediction for ocean and climate predictions is essential for multiple applications today. Many-Task Computing can play a significant role in making such predictions feasible. In this manuscript, we focus on ocean uncertainty prediction using the Error Subspace Statistical Estimation (ESSE) approach. In ESSE, uncertainties are represented by an error subspace of variable size. To predict these uncertainties, we perturb an initial state based on the initial error subspace and integrate the corresponding ensemble of initial conditions forward in time, including stochastic forcing during each simulation. The dominant error covariance (generated via SVD of the ensemble) is used for data assimilation. The resulting ocean fields are used as inputs for predictions of underwater sound propagation.
ESSE is a classic case of Many Task Computing: It uses dynamic heterogeneous workflows and ESSE ensembles are data intensive applications. We first study the execution characteristics of a distributed ESSE workflow on a medium size dedicated cluster, examine in more detail the I/O patterns exhibited and throughputs achieved by its components as well as the overall ensemble performance seen in practice. We then study the performance/usability challenges of employing Amazon EC2 and the Teragrid to augment our ESSE ensembles and provide better solutions faster.

Keywords—MTC; assimilation; data-intensive; ensemble;

1 Introduction

Our initial motivation was speeding up the execution of our stochastic ocean data assimilation ensembles via distributed computations and thereby allowing the evaluation of larger ensembles by the same hard deadline we are operating under. Our approach resulted in a clear case of a Many Task Computing (MTC) [1] application. This approach was extended by further augmenting ESSE ensemble size by employing remote resources on the Grid and the Amazon EC2 public Cloud.

Uncertainty prediction and data assimilation (the combination of observations and model results aiming for an optimal future state estimate with an error bound) is becoming ever more important a discipline, not only in the ocean sciences (as well as weather forecasting/climate prediction) but in other scientific/engineering fields where (field) measurements can be used to enhance the fidelity and usefulness of the computed solutions. It is clear that being able to use a larger ensemble size within the same time constraints can be very helpful and thus an MTC–style parallelization of ESSE is important, especially if it can be spread to such remote resources such as Grids and Clouds. More closely coupled parallelization approaches (whole-program like) to ensemble Kalman filter related techniques have already been attempted (e.g. [2], [3], [4]) but they have less opportunities for massive parallelism as they require a more tightly integrated parallel platform.

In what follows, Section 2 describes the application area of ocean data assimilation and provides details about the timeline of real-time data assimilation and ocean-acoustic modeling. Section 3 describes ESSE [5], [6], the data assimilation and error estimation approach used. Section 4 describes the ESSE implementation as a MTC application ([7], [8], [9]) and the options we face in terms of optimizing I/O issues. This is followed in Section 5 by a discussion of the practical MTC use of ESSE on local clusters, Grids and Amazon EC2. We discuss future work in Section 6. Conclusions are in Section 7.

2 Ocean Data Assimilation

Data Assimilation (DA) is a quantitative approach to optimally combine models and observations that is consistent with model and data uncertainties. Ocean DA can extract maximum knowledge from the sparse and expensive measurements of highly variable ocean dynamics. The ultimate goal is to better understand and predict these dynamics on multiple spatial and temporal scales. There are many applications that involve DA or build on its results, including: coastal, regional, seasonal, and inter-annual ocean and climate dynamics; carbon and biogeochemical cycles; ecosystem dynamics; ocean engineering; observing-system design; coastal management; fisheries; pollution control; naval operations; and

- *C. Evangelinos and C.N. Hill are with the Earth, Atmospheric and Planetary Sciences Department at the Massachusetts Institute of Technology, Cambridge, MA 02139*
  E-mail: ce107@mit.edu
- *P.F.J. Lermusiaux, J. Xu and P.J. Haley Jr. with the Mechanical Engineering Department at the Massachusetts Institute of Technology, Cambridge, MA 02139.*

defense and security. These applications have different requirements that lead to variations in the DA schemes utilized.

The ocean physics involves a multitude of phenomena occurring on multiple scales, from molecular and turbulent processes to decadal variations and climate dynamics. Life takes place in the ocean, from bacteria and plankton cells to fish and mammals. The range of space scales is from about 1 mm to 10,000 km, and of time scales, from about 1 s to 100 years and more. Features and properties in the ocean interact over these scales but significant interactions occur predominantly over certain ranges of scales, which are usually referred to as scale windows [10], [11]. For example, the internal weather of the sea, the so-called oceanic mesoscale, mainly consists of phenomena occurring over a day to months and over kilometers to hundreds of kilometers. This is one of the most energetic scale windows in the ocean and the present MTC study focuses on this window of processes.

A comprehensive prediction should include the reliability of estimated quantities. This allows an adequate use of these estimates in a scientific or operational application. In a prediction with a model integrating either in time and in space, errors in the initial data (initial conditions), boundary conditions and models themselves impact accuracy. Predicted uncertainties then contain the integrated effects of the initial error and of the errors introduced continuously during model integration. Mathematically, uncertainty can be defined here by the probability density function (PDF) of the error in the estimate. Since ocean fields are four-dimensional, uncertainty representations are here also fields, with structures in time and space.

Realistic simulations of four-dimensional ocean fields are carried out over broad numerical domains, e.g. O(10-1000) km for O(10-1000) days. The number of grid points and thus of discretized state variables are very large, usually of O(10^5-10^7). On the other hand, ocean data are limited in temporal and spatial coverage. Commonly, the number of data points for an at-sea sampling campaign is of O(10^4-10^5). For substantial scientific advances and to reduce uncertainties, the sources of information, the various data and dynamical models, are combined by data assimilation [12]. This combination is challenging and expensive to carry out, but optimal in the sense that each type of information is weighted in accord with its uncertainty.

2.1 Real Time Assimilation

An important clarification needs to be made regarding the different times involved in ocean forecasting: the observation time, forecaster time and simulation time (Fig. 1). New observations are made available in batches (Fig. 1, first row) during periods T_k, from the start of the experiment (T_0) up to the final time (T_f). During the experiment, for each prediction k (Fig. 1, zoom in middle row), the forecaster repeats a set of tasks (from τ_0^k to τ_f^k). These tasks include the processing of the currently available data and model (from τ_0^k to τ_0^i), the computation of $r+1$ data-driven forecast simulations (from t_0^i to t_f^{i+r}), and the study, selection and web-distribution of the best forecasts (from t_f^{i+r} to τ_f^k). Within these forecast computations, a specific forecast simulation i (Fig. 1), zoom in bottom row) is executed during t_0^i to t_f^i and associated to a "simulation time". For example, the ith simulation starts with the assimilation and adaptive modeling based on observations T_0, then integrates the dynamic model with data assimilation and adaptive modeling based on observations T_1, etc., up to the last observation period T_k which corresponds to the nowcast. After T_k, there are no new data available and the simulation enters the forecasting period proper, up to the last prediction time T_{k+n}.

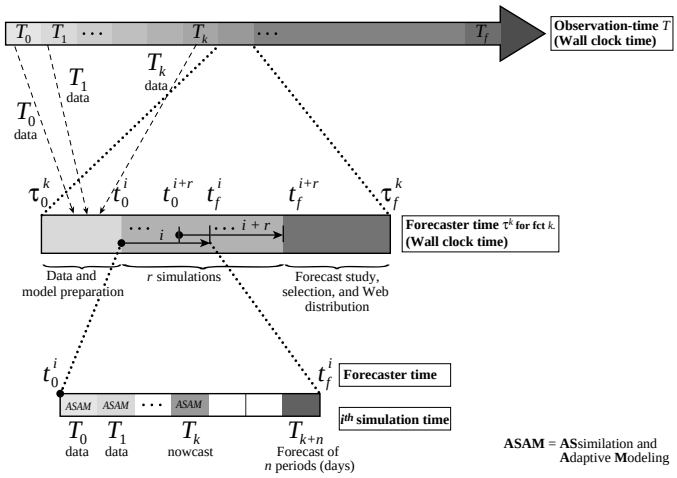


Fig. 1. Forecasting timelines. Top row: "Observation" or "ocean" time T during which measurements are made and the real phenomena occur. Middle row: "Forecaster" time τ^k during which the kth forecasting procedure and tasks are started and finished. Bottom row: "ith simulation" time t^i which covers portions of the real "ocean" time for each simulation. Multiple simulations are usually distributed on several computers, including ensembles of forecasts for uncertainty predictions (ESSE).

2.2 Ocean Acoustics

As one of the major application of underwater acoustics, sonar performance prediction requires the modeling of the acoustic field evolution. The parameters include the four-dimensional ocean and seabed fields. They are complex to predict and can have significant uncertainties. Methods and systems that forecasts the ocean, the seabed and the acoustics in an integrated fashion have only been developed and utilized recently. Our approach is based on coupling data-assimilative environmental and acoustic propagation models with ensemble simulations, as developed by [13], [14].

Having an estimate of the ocean temperature and salinity fields (along with their respective uncertainties)

provides the required background information for calculating acoustic fields and their uncertainties. Sound-propagation studies often focus on vertical sections. ESSE ocean physics uncertainties are transferred to acoustical uncertainties along such a section. Time is fixed and an acoustic broadband transmission loss (TL) field is computed for each ocean realization. A sound source of specific frequency, location and depth is chosen. The coupled physical-acoustical covariance P for the section is computed and non-dimensionalized. Its dominant eigenvectors (uncertainty modes) can be used for coupled physical-acoustical assimilation of hydrographic and TL data. ESSE has also been extended to acoustic data assimilation. With enough compute power one can compute the whole "acoustic climate" in a three-dimensional region, providing TL for any source and receiver locations in the region as a function of time and frequency, by running multiple independent tasks for different sources/frequencies/slices at different times.

3 Error Subspace Statistical Estimation

3.1 Formalism

Using continuous-discrete Bayesian estimation [15] and the notation of [16], the spatially discretized version of the deterministic-stochastic ocean model and parameter equations are combined into a single equation for the augmented state vector $\mathbf{x}$, of large but finite dimensions. Observations are taken at discrete instants $t_k \geq t_0$ and are concatenated into a data vector $\mathbf{y}_k^o$. The dynamics, observations and DA criterion are then,

$$d\mathbf{x} = \mathcal{M}(\mathbf{x}, t) + d\boldsymbol{\eta} \tag{1}$$

$$\mathbf{y}_k^o = \mathcal{H}(\mathbf{x}_k, t_k) + \boldsymbol{\epsilon}_k \tag{2}$$

$$\min_{\mathbf{x}} \; J(\mathbf{x}\,,\, \mathbf{y}_k^o\,;\, d\eta\,,\, \epsilon_k\,,\, \mathbf{Q}(t)\,,\, \mathbf{R}_k) \tag{3}$$

where $\mathcal{M}$ and $\mathcal{H}$ are the model and measurement model operator, respectively, J the objective function, and $d\boldsymbol{\eta}$ Wiener processes (Brownian motion), i.e. $\boldsymbol{\eta} \sim \mathcal{N}(0, \mathbf{Q}(t))$ with $\mathcal{E}\{d\boldsymbol{\eta}(t) d\boldsymbol{\eta}^T(t)\} \doteq \mathbf{Q}(t)\, dt$. Note that the deterministic ocean dynamics and parameter equations are actually forced by noise processes correlated in time and space. State augmentation [15], [17], [18] is used to re-write equations in the form of Eq. 1 which are forced by intermediary processes $d\boldsymbol{\eta}$ white in time and space. Measurement model uncertainties $\boldsymbol{\epsilon}_k$ are assumed white Gaussian sequences, $\boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \mathbf{R}_k)$. The initial conditions have a prior PDF, $p(\mathbf{x}(t_0))$, i.e. $\mathbf{x}(t_0) = \widehat{\mathbf{x}}_0 + \mathbf{n}(0)$ with $\mathbf{n}(0)$ random.

Error Subspace Statistical Estimation (ESSE, [5], [19], [20]) intends to estimate and predict the largest uncertainties, and combine models and data accordingly. When the DA criterion (Eq. 3) guides the definition of the largest uncertainties or "error subspace", the suboptimal truncation of errors in the full space is optimal.

ESSE proceeds to generate an ensemble of model integrations whose initial conditions are perturbed with randomly weighted combinations of the error modes. A central (unperturbed) forecast is also generated. The matrix of differences between each perturbed model realization in the ensemble and the central forecast is then generated and an estimate of the conditional mean is produced. A singular value decomposition (SVD) of the resulting normalized matrix provides us with the dominant error modes (based on a comparison of the singular values). A convergence criterion compares error subspaces of different sizes. Hence the dimensions of the ensemble and error subspace vary in time in accord with data and dynamics. The whole procedure can be seen in Figure 2.

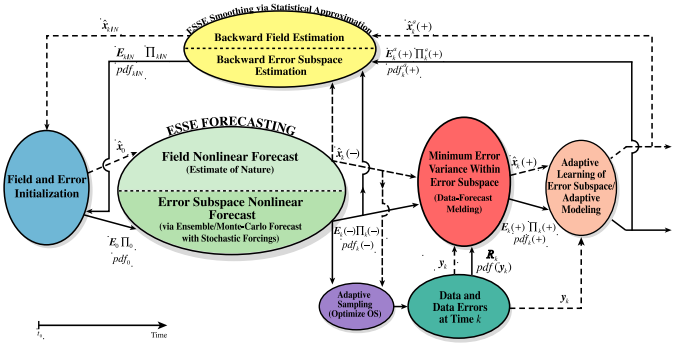


Fig. 2. The ESSE algorithm

The main component of the ESSE scheme that is used here is the uncertainty prediction. An initial condition for the dominant errors is assumed computed and available, using schemes given in [21], [22]. At t_k, $\widehat{\mathbf{x}}_k(+)$ is perturbed (Eq. 6) using a combination of error modes $\mathbf{E}_k(+)$ with random coefficients $\boldsymbol{\pi}_k^j(+)$. These coefficients are weighted by $\boldsymbol{\Pi}_k(+)$ and constrained by dynamics [19]. The truncated tail of the error spectrum is modeled by random white noise $\mathbf{n}_k^j$. For the evolution to t_{k+1}, a central forecast (Eq. 4) and an ensemble of $j = 1, ..., q$ stochastic ocean model integrations is run (Eq. 7), starting from the perturbed states $\mathbf{x}_k^j(+)$. The forcings $d\boldsymbol{\eta}(t)$ are defined in [5]. The ES forecast (Eq. 9) is computed from the ensemble. The matrix $\mathbf{M}_{k+1}(-) = [\widehat{\mathbf{x}}_{k+1}^j(-) - \widehat{\mathbf{x}}_{k+1}(-)]$ of differences between q realizations and an estimate of the conditional mean, e.g. $\widehat{\mathbf{x}}_{k+1}^{\mathrm{em}}(-)$ in Eq. 5, is then computed. It is normalized and decomposed (Eq. 9) into $\boldsymbol{\Pi}_{k+1(-)} \doteq \frac{1}{q}\boldsymbol{\Sigma}_{k+1}^2(-)$ and $\mathbf{E}_{k+1}(-)$ of rank $p \leq q$ by singular value decomposition (the operator $\mathrm{SVD}_p(\cdot)$ selects the rank-p SVD). The ensemble size is limited by a convergence criterion (Eq. 10). The coefficient ρ used here measures the similarity between two subspaces of different sizes ([23], [24]). A "previous" estimate $(\mathbf{E}, \boldsymbol{\Pi})$ of rank p and "new" estimate $(\widetilde{\mathbf{E}}, \widetilde{\boldsymbol{\Pi}})$ of rank $\tilde{p} \geq p$ are compared, using singular values to weight singular vectors. The scalar limit α is chosen by the user $(1 - \epsilon \leq \alpha \leq 1)$. $\sigma_i(\cdot)$ selects the singular value number i and $k = \min(\tilde{p}, p)$. When ρ is close to one,

$(\widetilde{\mathbf{E}}, \widetilde{\boldsymbol{\Pi}})$ is the error forecast for t_{k+1}: $\boldsymbol{\Pi}_{k+1(-)}$, $\mathbf{E}_{k+1(-)}$. The dimensions of the ensemble (q) and ES (p) hence vary with time, in accord with data and dynamics.

$$Central fcst: \quad (4)$$
$$\widehat{\mathbf{x}}^{\mathrm{cf}}_{k+1(-)} \;\big|\; d\widehat{\mathbf{x}} = \mathcal{M}(\widehat{\mathbf{x}}, t)\, dt \;, \quad \text{with} \;\; \widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_{k(+)}$$
$$Ens.mean: \quad (5)$$
$$\widehat{\mathbf{x}}^{\mathrm{em}}_{k+1(-)} \doteq \mathcal{E}^q\{\widehat{\mathbf{x}}^j_{k+1(-)}\}$$
$$ESIn.\ Cond.: \quad (6)$$
$$\widehat{\mathbf{x}}^j_{k(+)} = \widehat{\mathbf{x}}_{k(+)} + \mathbf{E}_{k(+)}\,\boldsymbol{\pi}^j_{k(+)} + \mathbf{n}^j_k \;, \quad j = 1, ..., q \;.$$
$$Ens.\ Fcst: \quad (7)$$
$$\widehat{\mathbf{x}}^j_{k+1(-)} \;\big|\; d\widehat{\mathbf{x}}^j = \mathcal{M}(\widehat{\mathbf{x}}^j, t)\, dt + d\boldsymbol{\eta} \quad \text{with} \;\; \widehat{\mathbf{x}}^j_k = \widehat{\mathbf{x}}^j_{k(+)}$$
$$ESFcst: \quad (8)$$
$$\mathbf{M}_{k+1(-)} = \left[\widehat{\mathbf{x}}^j_{k+1(-)} - \widehat{\mathbf{x}}_{k+1(-)}\right]$$
$$\left\{\boldsymbol{\Sigma}_{k+1(-)}, \mathbf{E}_{k+1(-)} \;\middle|\; \mathrm{SVD}_p(\mathbf{M}_{k+1(-)}) = \mathbf{E}_{k+1(-)}\boldsymbol{\Sigma}_{k+1(-)}\mathbf{V}^T_{k+1(-)}\right\} \quad (9)$$
$$Conv.Crit.: \quad (10)$$
$$\rho = \frac{\sum_{i=1}^{k} \sigma_i(\boldsymbol{\Pi}^{\frac{1}{2}}\mathbf{E}^T\widetilde{\mathbf{E}}\widetilde{\boldsymbol{\Pi}}^{\frac{1}{2}})}{\sum_{i=1}^{\tilde{p}} \sigma_i(\widetilde{\boldsymbol{\Pi}})} \geq \alpha$$

Acoustic predictions are generated using acoustic propagation models and newly developed parallel software. With this new parallel acoustic software, we compute the whole "acoustic climate" in a three-dimensional region, providing transmission loss (TL) for any source and receiver locations in the region as a function of time and frequency.

4 ESSE WORKFLOW

The ESSE calculations require the calculation of a very large ensemble of ocean forecasts. This imposes significant demands on computational power and storage. ESSE ensembles, however, differ from typical high throughput applications such as parameter scans in more than one way:

1) there is a hard deadline associated with the execution of the ensemble, as a forecast needs to be timely;
2) the size of the ensemble is dynamically adjusted according to the convergence of the ESSE procedure;
3) individual ensemble members are not significant (and their results can be ignored if unavailable) - what is important is the statistical coverage of the ensemble;
4) the full resulting dataset of the ensemble member forecast is required, not just a small set of numbers;
5) individual forecasts within an ensemble, especially in the case of interdisciplinary interactions and nested meshes, can be parallel programs themselves.

Point (1) above hints towards the use of the any Advanced Reservation capabilities available; point (2) means that the actual computing and data requirements for the forecast are not known beforehand and change dynamically; point (3) suggests that failures (due to software or hardware problems) are not catastrophic and can be tolerated - moreover runs that have not finished (or even started) by the forecast deadline can be safely ignored provided they do not collectively represent a systematic hole in the statistical coverage. Point (4) means that relatively high data storage and network bandwidth constraints will be placed on the underlying infrastructure and point (5) means that the computing requirements will not be insignificant either.

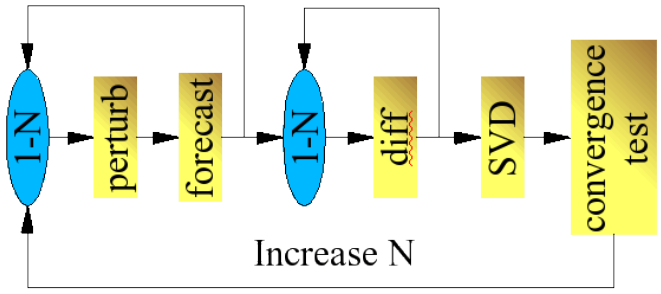


Fig. 3. The serial ESSE implementation

In the case of the ESSE approach to Data Assimilation, a central process acts as a job shepherd for the ensemble, as shown in Fig 3: A loop of N ensemble members is first calculated, each member consisting of a perturbation of the initial conditions/parameters and a forecast. After all members are calculated, the difference of the resulting forecast from a central forecast is calculated in a loop, creating a large file containing the uncertainty covariance matrix. A Singular Value Decomposition (SVD) of this matrix ensues followed by a convergence test with the result of the previous SVD. If convergence is not achieved, the process loops back to increase N to N_2, up to some maximal value N_{max} or until the time T_{max} available for the forecast expires. The process then restarts for the ensemble members $N+1$ to N_2. This approach suffers from several bottlenecks:

1) The perturb/forecast loop needs to finish for the diff loop to start (or the two loops can be fused (merged). Either way there is no exposed parallelism.
2) The diff loop has a serial bottleneck (the same file is written to). Depending on the variant of the perturbation type employed, it may also expect to add the perturbations to the uncertainty covariance matrix in the order they were generated.
3) The SVD/convergence test has to wait for the diff loop to finish.
4) The SVD and the convergence test are large calcula-

tions requiring a lot of memory and time, especially for large N.

4.1 Parallelized ESSE

We considered a natural transformation of the ESSE process to address these bottlenecks and increase the amount of exploitable parallelism, transforming the problem into one amenable to MTC techniques - see also Fig 4. Specifically we dealt with bottleneck 1 above by replacing the concept of the loop with that of a pool of ensemble calculations, of size $M \geq N$. These calculations can be done concurrently on different machines, as there is no actual serial dependence in the forecasting loop. They would in effect be the MTC element of the forecasting procedure. We then decouple the diff loop by having it run continuously, adding new elements to the uncertainty covariance matrix, as they become available from the forecast ensemble calculations. Furthermore, we relax our requirement that elements of the covariance matrix are in the order of the perturbation number (bottleneck 2) and instead keep track of which perturbation is added every time for bookkeeping purposes. Unfortunately we cannot easily do away with the single file bottleneck on the diff loop and that forces us to limit the diff calculation to a single machine with access to lots of disk space as the covariance matrix tends to be very large (O((N G V)2) where G is the number of 3D grid points and V the number of physical fields and biochemical/physical tracer variables).

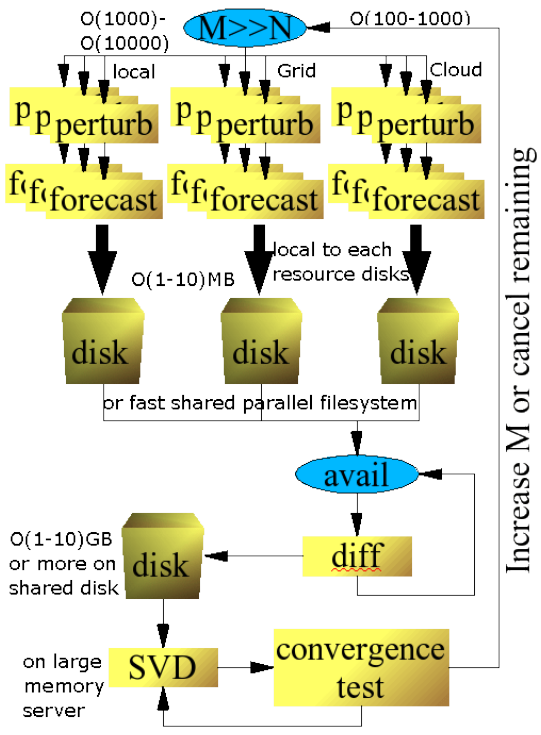


Fig. 4. The parallel ESSE implementation

The SVD calculation and the convergence test are also decoupled from the diff loop by running continuously on their own, using the latest result available from the diff loop. To fully decouple the loops without introducing a race condition on the covariance matrix file between its reading for the SVD and its writing by diff, we employ three files, a safe one for SVD to use and a live alternating pair for diff to write to, with the safe one being updated by the the appropriate member of the pair. The SVD calculation and the convergence test proceed on its own with the requirement of fast I/O access to the safe file and a machine with large memory and many processors for the parallel SVD calculation on a dense matrix (for the time being we are employing shared-memory parallel LAPACK calls though the use of SCALAPACK for distributed memory clusters may become necessary in the future if our ensembles get too large).

If the convergence test succeeds, the remaining ensemble members (queued for execution or running) are canceled, and depending on the time constraints (for forecast timeliness) and an associated policy, either the ensemble calculation concludes immediately or the remaining ensemble results already calculated are diffed, another SVD calculation is performed and all available results are used. In theory one could also spare any ensemble calculations close to finishing (according to performance estimates for the machines they are executing on and accumulated runtime), to further minimize the wasted cycles at the expense of further delays.

If the convergence test fails for a number of ensemble members sufficiently close to $M < N_{max}$, the ensemble pool can be enlarged (in stages) up to N_{max} (or even slightly more) in order to ensure convergence and at the same time make sure that there is no point during this process where the pipeline of results drains and the SVD calculation has to wait (aside from the startup wait).

4.2 Implementation specifics

The ESSE workflow is implemented as a shell script in variants targeting either Sun Grid Engine (SGE) [25] or Condor [26]. If the shell script catches the kill signal it proceeds to cancel all pending jobs and do some cleanup. This *master* script that runs on a central machine on the *home* cluster launches *singleton* jobs that implement the perturb/forecast ensemble calculations (*pert* and *pemodel* executables respectively). The differ, SVD and convergence check calculations proceed semi-independently, either on the same machine as the *master* script or on some other machine with access to the same filesystem and lots of memory. They wait to ascertain that a multiple of a set number of realizations has finished and then they run. We allow for variants where the perturb/forecast ensemble is split in two, first all the perturbations are generated and then the forecasts are run. This makes sense only in case that there are very few machines with good network connections to the storage hosting the large files that perturb needs to read (hundreds of MBs to GBs). In that case it makes

sense to restrict the execution of *pert* to those machines only and split the ensemble workflow. Dependencies are tracked using separate (per perturbation index) files containing the error codes of the singleton scripts (which are set on purpose to signify success or failure). These files reside in directories accessible directly or indirectly from all execution hosts so that state information can be readily shared. Moreover the perturbation index number is passed on to each singleton either by cleverly altering the name of each job submission to include it or by stripping it off the task array. The latter approach is more desirable (as it places less strain on the job scheduler) but it cannot be used if the ESSE execution gets stopped. Any ESSE restarts that avoid rerunning jobs have to switch to a one-job per perturbation index submission strategy.

5 ESSE AS AN MTC APPLICATION IN PRACTICE

In this section we shall discuss how ESSE performs as a MTC application in a qualitative as well as quantitative manner: Its specialized requirements, its I/O characterization and performance in our local cluster environment and its performance on some of the various platforms available in a distributed Grid and public Cloud setting.

5.1 Special ESSE needs

ESSE and other similar ensemble-based ocean forecasting methodologies are used several times a year in a real-time setting during live ocean experiments lasting weeks to months. In the past, any calculations that was more involved than a simple serial forecast (possibly employing objective analysis based data assimilation which could still be handled by a powerful on-board workstation) had to be performed back on land. Remote computer clusters at participating academic/commercial or military institutions were used, connected via slow links to the ship-borne measurement apparatus. Advances in computer system and networking technology have now resulted in the availability of a ship-borne computing infrastructure (of a rack or even deskside form factor) to handle pretty large basic ensemble calculations. At the same time the constant drive for higher resolution, better (and more usually than not - more complex) models and comprehensive error subspace representations have resulted in considerably larger increases of the computational demands. In practice this means that for the "real-time" requirements to be satisfied, the use of land-based clusters is still required for the more involved ESSE analyses.

This suggests that use of a dedicated *home* cluster resource is definitely worthwhile as such a system is under the complete control of the PIs and can be devoted entirely to the needs of the real-time experiment. Such systems are also necessary because a lot of other incubating computations are required, either to prepare such experiments and develop new methods and software for it, or to carry out other independent research work.

Importantly, the local *home* cluster resources should be augmented by remote machines that are not under the direct control of the user. Such resources can be provided in the form of batch-controlled allocations on (in the case of the USA and depending on the sponsoring agency) NSF, DoE, DoD, NOAA or NASA shared compute resources or more generally via use of cloud computing based virtual clusters, such as Amazon's EC2. Such systems can be utilized on demand, as a function of the real-time needs over limited periods.

5.2 ESSE on a local cluster

Our local cluster is composed of 114 dual socket Opteron 250 (2.4GHz) nodes (1 with 16G RAM, 2 with 8GB and the rest with 4GB), 3 dual socket Opteron 285 (dual core 2.6GHz) nodes, all with 4GB RAM (replacement nodes), and a dual socket Opteron 2380 (Shanghai generation, quad core 2.5GHz) head node with 24GB RAM. The dual socket Opteron fileserver serves over 18TB of shared RAID6 disk (ext3 filesystem, LSI 8888 ELP controller) over NFS, using a 10Gbit/s connection to a 200Gbit/s switch backbone. All nodes have a Gigabit Ethernet connection to switches arranged in a star formation, feeding into the central switch. The cluster has both SGE and Condor installed and active (at the same time). Condor is setup to consider nodes used by SGE as claimed by their "owner" so the two systems can coexist (with Condor giving precedence to SGE). All users tend to use only one of the two systems at the time.

5.2.1 I/O analysis using strace

Beyond the compute side needs of the major components of ESSE we have to consider the I/O stresses that they put on our storage infrastructure. Both *pert* and *pemodel* (as well as the other executables) employ NetCDF [27] for reasons of portability - this however tends to make I/O requests rather opaque as they are hidden under the hood of the NetCDF library. We chose to use a tool that looks at I/O performance without requiring recompilation or other type of instrumentation of executables: strace_analyzer [28] employs low level hooks in the operating system to capture read, write and other I/O requests. In the parts that follow, KB, MB etc. refer to powers of 10 instead of powers of 2, e.g. 1KB=1000 bytes etc.

5.2.2 I/O needs of the perturbation generators

When *pert* is run on the local filesystem I/O time is about 27% of the total time. As can be seen in Table 1, there are a great many I/O calls of a small size.

The total number of bytes written was 4.5MB in 546 total calls. The average (mean) bytes per write call were 8,197 bytes, with a standard deviation of 649 bytes. The median bytes per call were 2 pages (8192 bytes).

The total number of bytes read was 435.5MB, in 53,072 total calls. The average (mean) bytes per read call was

TABLE 1
pert write/read sizes

I/O size range	# of requests	type
0-1KB	11	read
1-8KB	3	read
8-32KB	53603	read
0-1KB	1	write
1-8KB	1	write
8-32KB	544	write

8,206 bytes, with a standard deviation of 373 bytes. The median bytes per call were as before 2 pages.

A total of 4,363,806 bytes were read from the central forecast file, 426,639,520 were read from the error subspace matrix and 4,494,987 bytes were read from with 4,472,832 written to the perturbed ICs.

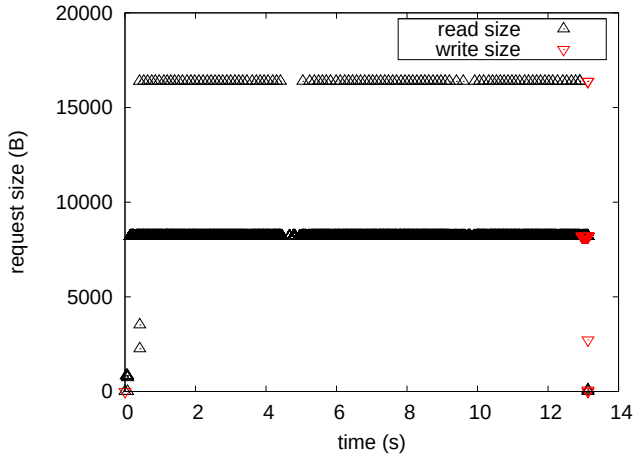


Fig. 5. Read/write I/O request sizes for *pert* as a function of time

Looking at a breakdown in time of the I/O requests in Figure 5 we see that there is a continuous set of reads throughout the run in two sizes (8 & 16KB); a write happens at the very beginning and the rest at the end.

Looking at a breakdown in throughput of the I/O requests in Figure 6 we see that there is a wide spread of read bandwidth as seen by the application (ie. including filesystem cache effects) throughout the run. Similarly for the writes at the end of the run, one sees a range of performance values.

Overall we can state that *pert* is partly I/O bound (but finishes quickly on local disk) - the main I/O involves reading part of the error subspace matrix. The individual I/O calls by the NetCDF library are both numerous and small: The calls at a granularity that is smaller than a page are insignificant but such a pattern is not well suited for large parallel filesystems (such as PVFS2 [29]) which are tuned for large streaming stores. Moreover the internal structure of the NetCDF files appears to necessitate a lot of reading and then writing of the output file which in many cases can cause trouble with NFS.

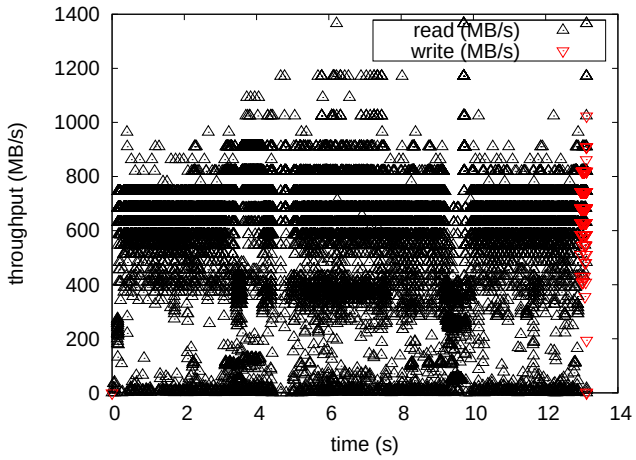


Fig. 6. Read/write I/O throughput (in MB/s) for *pert* as a function of time - filesystem cache is enabled

TABLE 2
pemodel write/read sizes

I/O size range	# of requests	type
0-1KB	3290	read
1-8KB	4227	read
8-32KB	137133	read
0-1KB	2019	write
1-8KB	148	write
8-32KB	59532	write

5.2.3 I/O needs of the ocean model

When *pemodel* is run on the local filesystem I/O time is only about 0.24% of the total time (ie. the code, as currently setup is not I/O limited). As can be seen in Table 2, there are a great many I/O calls of a small size.

The total number of bytes written was 492MB in 61,699 total calls. The average (mean) bytes per write call were 7,976 bytes, with a standard deviation of 1,584 bytes. The median bytes per call were again 2 pages (8,192 bytes)

The total number of bytes read was 679MB, in 82,952 total calls. The average (mean) bytes per read call was 8,187 bytes, with a standard deviation of 2,121 bytes. The median bytes per call were as before 2 pages.

A total of 173,283,024 bytes were read from the forcing file, 8,541,726 were read from the IC and 480,431,123 were read from, with 490,541,996 being written to the main output file - another 468,564/721,57 bytes are read from/written to a secondary output file, while $12.4MB$ and $3.9MB$ are read from the pressure bias and shapiro filter input files respectively.

Looking at a breakdown in time of the I/O requests in Figure 7 we see that there is a flurry of activity at the very beginning and end of the run; reads continue in two main sizes for the first 3rd of the run or so. Writes are very small and spread throughout.

Looking at a breakdown in throughput of the I/O requests in Figure 8 we see that there is a wide spread of bandwidth as seen by the application (ie. including

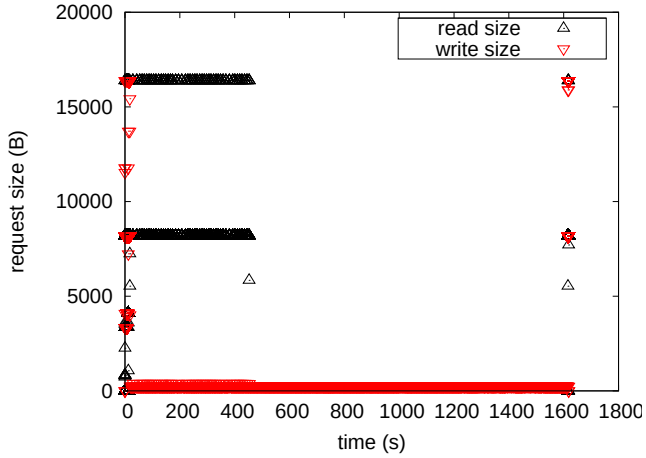


Fig. 7. Read/write I/O request sizes for *pemodel* as a function of time

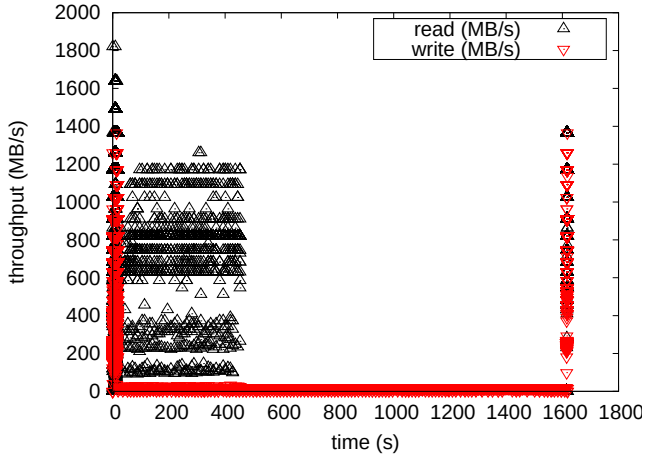


Fig. 8. Read/write throughput (in MB/s) for *pemodel* as a function of time - filesystem cache is enabled

filesystem cache effects) throughout the run. A range of low to high performance is seen at the beginning and the end; the small writes in the rest of the run cannot achieve high performance.

Overall we can state that *pemodel* is not I/O bound when working out of local disk - most of the I/O involves reading the forcing and reading/writing the output files. As before the nature of NetCDF necessitates a lot of rereading of the output file.

5.2.4 Performance implications over NFS

If we are to look at the performance of a single *pert* with input/output files stored over NFS (Gigabit Ethernet high performance network used by the Steele Cluster at Purdue University) we discover that the total runtime can increase by a very large amount - from a few seconds to 2.5 minutes or more (274+ secs in Figure 9. In fact in the past, over more constrained LANs (100Mbit mix of switches and hubs) we have seen times as high as 700+ seconds, making *pert* very expensive.

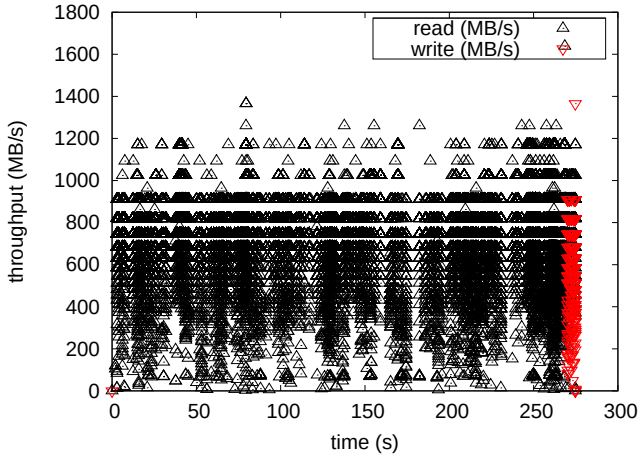


Fig. 9. Read/write throughput (in MB/s) for *pert* over NFS as a function of time - filesystem cache is enabled

The performance drop may not show in the percentage of time spent doing I/O, indeed that may decrease, as the extra time appears simply as wallclock time - the process idles as the kernel waits on NFS activity. Thus the individual bandwidth figures in Figure 9 appear to cover a decent range of values - it is the total wallclock time that suffers. If this is a problem with a single *pert* reading/writing over NFS - multiple ones (as would be the case in a production ensemble of hundreds or thousands) can only make the problem worse. The load on the NFS server increases to very high values and the fileserver may become unresponsive. This suggests that use of diskless clusters is not well suited to this application.

5.2.5 The multicore picture

In the case of a multicore (or even older unicore)/multisocket node of N processor cores (the expected norm for most systems nowadays) the resulting read/write activity as seen at the local filesystem level would be that of N superimposed copies of Figures 5 and 7 respectively, slightly shifted in time (allowing for varied start times due to the queue/OS scheduler) as well as stretched in time (due to contention issues). The basic picture however does not change unless the shifting becomes very regular and the result is a more uniform pattern (which would a very high core count).

If one were to try and devise a better suited distributed filesystem for ESSE workloads one should consider a filesystem that:

1) adaptively (or based on hints) propagates replicas of large files (especially those larger than the filesystem cache) that everyone is reading to more locations to spread the load
2) has a distributed metadata server mechanism (or no metadata server at all as in the case of GlusterFS [30]) to be able to handle the large number of concurrent metadata operations.

3) allows local disk to be used in a unified namespace for output (similar to what GlusterFS calls NUFA - non-uniform file access).

5.2.6 Timings

For the timings discussed below about 210 of the 240 cores were available - the rest were in use by other users. We tested two scenarios: one that uses NFS for the large input files and another that prestages (to every local disk) all input files so that all input is local. We did not test the case where both input and output files live on the NFS server for the duration of the execution of the singletons as it places too much stress on the NFS server and is disruptive to other users. Therefore in all cases the useful output files are copied back to the NFS server at the end of their job. In all cases the differencing, SVD and convergence check calculations were happening on the master node.

This I/O optimization made more of a difference for the perturbation part of the algorithm, where CPU utilization jumped from $\approx 20\%$ to $\approx 100\%$. The initial conditions generated thus and used for the ensemble model runs are stored on the local directory anyway and therefore this (more expensive) part of the ESSE procedure does not offer as much of a performance boost. 600 ensemble members pass through the ESSE workflow in $\approx 77mins$ in the all local I/O case and in $\approx 86mins$ in the mixed locality case. As all nodes were equally close to the fileserver we did not deem it necessary to test the ESSE variant where the perturbation calculation is done in a separate job submission from that of the PE model. For both SGE and Condor we used job arrays to lessen the load on the scheduler.

Timings under Condor were between $10-20\%$ slower. Essentially the difference could be seen in the time it took for the queuing system to reassign a new job to a node that just finished one. In the case of SGE the transition was immediate - Condor appeared to want to wait. We tweaked the configuration files to diminish this difference in throughput which is probably due to the effort put in Condor to function as a very successful cycle harvester and the resulting care it takes not to disrupt everyday desktop usage.

The ESSE calculation was followed by more than 6000 ocean acoustics realizations - each of which executed for approximately 3 minutes - in this case no job arrays were used and the system handled all 6000+ jobs without any problem whatsoever.

5.3 ESSE on the Grid

The task at hand is to augment the ESSE ensemble size by employing remote resources (usually but not always Grid-enabled). That could be either a departmental cluster within the same overall organization, a partner institution Grid or the large-scale national and international Grid infrastructures such as the Teragrid, Open Science Grid, EGEE etc.

The disadvantage of dealing with Grid resources is that they come with a wide variety of rather heavyweight middleware (such as Globus, gLite, Unicore5/6, OMII-UK, ARC, GRIA) that are not very easy to install and require maintenance over time. In this manner they represent an additional burden on both the users and the administrators.

5.3.1 Scheduling ensembles

The easiest (while at the same time least flexible) way to add Grid resources for the execution of our ensembles was remote submission/cancellation of jobs (using (gsi)ssh + the local job manager commands) either individually or as a job array. Essentially a small part of the ESSE *master* script dealing with job submission/cancellation is replicated on the remote resource. *singleton* scripts particular to the remote system in question are submitted and no complicated logic is needed to make them work as they are not generic. The directories that keep track of job submissions/completions etc. on the *home* cluster are either mounted on the remote system using XUFS [31], SSHFS [32] etc. or they are updated using passwordless SCP connections (to avoid requiring setting up Globus or other Grid infrastructure servers on the *home* cluster end. This approach gives no easy way for the user to monitor the progress of one's jobs (other than to try to monitor the contents of the submission/completion directories). One needs to take care to assign a clearly separated block of ensemble members to these external Grid execution hosts to avoid overlaps.

A different path is offered by the wide availability of the Condor software. The existing Condor implementation of ESSE needs to be slightly adjusted to allow for use of remote clusters either via flocking, Condor-C or Condor-Glidein. Unfortunately all of these approaches entail modification of the configuration of the *home* Condor cluster and sometimes even of the remote cluster - something we are able to do locally but in general a non-privileged user cannot do. Further issues (which can be avoided with careful configuration choices) can arise when other users' jobs (also submitted to the local Condor queues) end up on remote Grid resources they cannot be executed on. The remaining alternative, Condor-G, on the other hand is not as capable of handling so many jobs as we are envisioning.

One other possibility (which circumvents these problems) is the use of Personal Condor (in which case all local configuration files are owned by the user), connecting via Condor-Glidein to both the local Condor pool and the remote clusters. A related effort which we plan to investigate further is the use of the MyCluster [33] software that makes a collection of remote and local resources appear as one large Condor or SGE controlled cluster. This way we we are not limited to Condor but we can use our SGE-based setup instead.

TABLE 3
pert/*pemodel* performance (time to completion in seconds) on a few Teragrid platforms

site	processor type	pert	pemodel
ORNL	Pentium4 3.06MHz	67.83	1823.99
Purdue	Core2 2.33MHz	6.25	1107.40
local	Opteron 250 2.4GHz	6.21	1531.33

5.3.2 I/O issues

There are significant I/O issues that need to be addressed when considering the use of remote resources for ESSE ensembles. As a minimum requirement the shared input files can be read remotely from OpenDAP servers at the *home* institution (using the NetCDF-OpenDAP library) allowing the immediate opportunistic use of a remote resource that is discovered to be idling. The performance implications of such an approach however (hundreds of requests to a central OpenDAP server make it a less desirable solution. Therefore one is more likely to employ manual prestaging of the input files - use of shared filesystems over a WAN can help speed up such operations (e.g. one copy from *home* to gpfs-wan and then a fast distribution from gpfs-wan to local fast disks. Use of data staging engines such as Stork are another possibility, provided they work with our scheduler.

When it comes to the output files, one has the choice of either a *push* model (from the remote execution hosts back to the *home* cluster) or a *pull* model (a pull-agent on the *home* cluster fetching files from a central repository for each of the remote clusters). The former method is the simplest one requiring the least book-keeping - at the same time it requires nodes that can talk to the outside world and the batch nature of the runs results in a very large number of concurrent remote transfer attempts followed by no network activity whatsoever. This can seriously slow down the gateway nodes of the *home* cluster. The *pull* model requires more work (a separate agent, notifications that files have been copied so they can be safely deleted etc.) but can pace the file transfers so that they happen more or less continuously and perform much better. A third alternative introduces a two-stage *put* strategy - with nodes storing their output on a shared filesystem and an independent agent transferring them over to the *home* cluster.

5.3.3 Computational issues

An idea of the speeds of Teragrid hosts running *pemodel* and *pert* vs. the speeds seen on our local *home* cluster is shown in Table 3.

As one can see speeds vary appreciably. The slow *pert* performance for ORNL appears to be mainly related to the PVFS2 filesystem used - the Purdue runs employ a local fast filesystem. In practice this means that the more dissimilar the hosts used to augment the local compute facilities, the more uneven the progress of the various remote clusters will be and perturbation 900 may very well finish well before number 700.

5.3.4 Evaluating ESSE on the Grid

There are many advantages of using the Grid to augment local compute resources for ESSE:

- There are a great many computational resources available on the Grid. Teragrid's Condor pool is claimed to be almost 27,000 cores but at the time of the writing of this paper only about 1828 appeared to be available for use, with around 100 at a time free to run a user job.
- Many Grid-enabled systems have been designed with massive I/O requirements in mind, allowing for fast access from many nodes to a shared filesystem. Unfortunately the tuning of such parallel filesystems is usually generic (or targeted towards specific grand-challenge workloads) and the numerous small size requests that our ESSE workload generates can prove to be a bad fit.
- Similarly large shared Grid-enabled systems usually have excellent connectivity to the fastest Internet backbone and allow for fast file transfers to and from the *home* system.

At the same time there are significant disadvantages of using the Grid:

1) Each remote resource is slightly to very different in hardware, software (O/S, compilers and libraries) and filesystem configuration. This means that the user is faced not only with having to rebuilt and redeploy the code binaries every time but also with modifying variables in the *singleton* execution scripts to match the particulars of the filesystem/operating system setup at hand.
2) Due to the shared nature of resources on large external centers one cannot be sure that there will be enough nodes on a single resource to reach the capacity needed. In the absence of advance reservation the jobs submitted may very well end up running on the following day (or in any case outside the useful time window for ocean forecasts to be issued). So many different Grid resources at the same time would have to be employed (with the resulting increase in complexity).
3) A careful estimate of the duration of the jobs can help in case backfilling is employed on the queuing system of the Grid resource but even in that case commonly used limitations of active jobs (irrespective of total core count) per user can throttle back performance expectations.
4) Moreover in many cases the queuing system scheduler has been tuned to prioritize large core count parallel jobs and thereby penalize massive task parallelism workloads. In that case one needs to refactor *singleton* jobs to batches of *singletons* packaged as a single job (with all the extra trouble this refactoring can introduce).

Advance reservations (which are not yet widely available if at all possible) will be necessary to ensure that a sufficient number of cpu power will be available. Experiments are planned ahead of time to allow for such reservations to be made but their daily time boundaries cannot be very tight.

Another issue with the MPP platforms available on the Grid that offer massive numbers of processors for high throughput/massive task parallelism type of workloads is that their I/O configuration and support for running scripts can be limited. Case in point are the IBM Blue Gene/L systems (like NCAR's Frost on the Teragrid) which share one I/O node for a number of compute nodes and does not offer a complete O/S environment on the compute node to support running a script. Full support for running shell scripts on MPP compute nodes unfortunately may go against the general philosophy of having them run a minimized O/S in order to better perform when running closely coupled parallel codes.

5.4 ESSE in the Cloud

The emerging Cloud Computing infrastructure offers us a different avenue we can pursue to augment the ESSE ensemble size. Given our needs we are interested in the IaaS (Infrastructure as a Service) form of Cloud Computing services. In particular we have experimented with what is currently the most easy to use IaaS system, Amazon's EC2.

5.4.1 Scheduling ensembles

EC2 offers a set of tools that allow the provisioning and booting of various Linux, Solaris and Windows Xen virtual machine images (called AMIs) and allows the remote user to login to them as an administrator and control them accordingly. There is also control over which ports each live instance has open to the internal EC2 network as well as the outside world. This level of complete control allows us a wide variety of options on how to use EC2 provisioned nodes for ESSE calculations:

- Creation of an independent on-demand cluster, with its own master node and queuing system and remote submission of jobs in the same way as for a generic remote cluster/Grid environment.
- Addition of the EC2 nodes to the home cluster as extra compute nodes. This has already been demonstrated for GridEngine and we have been able to replicate it. Condor also offers the ability to launch jobs on Amazon EC2 nodes but the way that they are provisioned (essentially as a job) and controlled is too restrictive for our needs.
- Creation of a personal (Condor or SGE) private cluster using MyCluster mixing local and EC2 resources.
- Dynamic addition of EC2 nodes to an existing cluster - offered in product form by Univa (UniCloud) and Sun (Cloud Adapter in Hedeby/SDM).

This last option automates the booting/termination of EC2 nodes based on queuing system demand, further minimizing costs. Most of the options allow for minimal changes to the generic SGE setup.

TABLE 4
pert/*pemodel* performance (time to completion in seconds) on various EC2 instance types - Opt stands for Opteron

site	processor type	pert	pemodel	cores
m1.small	Opt DC 2.6GHz	13.53	2850.14	0.5
m1.large	Opt DC 2.0GHz	9.33	1817.13	2
m1.xlarge	Opt DC 2.0GHz	9.14	1860.81	4
c1.medium	Core2 2.33GHz	9.80	1008.11	2
c1.xlarge	Core2 2.33GHz	6.67	1030.42	8
m2.2xlarge	Nehalem 2.67GHz	3.39	779.77	4
m2.4xlarge	Nehalem 2.67GHz	3.35	790.86	8

5.4.2 I/O and computational issues

The I/O issues of the Amazon EC2 option are similar to the Grid ones but compounded by the fact that neither the networking nor the disk hardware are geared towards high performance computing. Similar solutions can be adopted, with an emphasis on avoiding issues resulting from the relatively low network bandwidth of EC2 to the outside world. Any common staging areas can be provided either via NFS exporting a persistent EBS volume or populating an on-demand created parallel filesystem with data from EBS. In the latter case extra work needs to be made to ensure that the AMIs can function as clients for the parallel filesystem. Given the issues with NFS performance discussed in Section 5, NFS staging areas should not be (ab)used for massive concurrent I/O from remote nodes, however appealing the simplicity of such an approach may appear.

An idea of the times of several EC2 instances running *pemodel* and *pert* for various instance types is shown in Table 4. As of the last few months of 2009 the new Amazon datacenters have introduced new varieties of actual hardware for the various EC2 instance types - hence the m1.small (standard) instance can be either AMD Opteron - 2.0GHz or 2.6GHz dual core (DC) based - or Intel (Core2) Xeon 54xx series (Penryn generation) quad core (QC) based. The latter type of hardware can offer even better performance that the one shown in Table 4. Amazon recently introduced Intel Nehalem-based instance types with increased memory capacity, effectively allowing all of the ESSE calculations to be conducted in the Cloud, even for very large ensemble sizes as the nodes have enough memory to handle the SVD of extremely large matrices.

In all the cases shown the instance type was fully utilized (ie. 8 copies of *pert*/*pemodel* were run concurrently on a c1.xlarge instance). The m1.small instance appears as a 1 core but is in fact limited to a maximum of 50% (or even 40% for newer cpu types) cpu utilization, hence appearing as a half-core. The executables (and software environment) were identical to those on the *home* cluster. In each case the worst time of the batch is being reported. Despite the faster Nehalem generation

processors available on the m2 series instances, the most cost-effective approach balancing price and performance appears to be c1.medium (or c1.xlarge if one wants to maximize the number of execution cores for a given number of instances).

Cost-wise for example an ESSE calculation with 1.5GB input data, 960 ensemble members each sending back 11MB (for a total of 6.6GB) would cost: $1.5(GB) \times 0.1 + 10.56(GB) \times 0.17 + 2(hr) * 20 * 0.68 = \$29.15$ Use of reserved or spot instances would drop pricing for the cpu usage by a bit less than a factor of 3.

5.4.3 Evaluating ESSE on EC2

There are quite a few clear advantages of using EC2 for larger ESSE ensembles:

- For all intents and purposes the response is immediate. EC2's capacity is large enough that a request for a virtual EC2 cluster gets satisfied on-demand, without having to worry about queue times and backfill slots.
- The use of virtual machines allows for deploying the same environment as the *home* cluster. This provides for a very clean integration of the two clusters.
- Having the same software environment also results in no need to rebuild (and in most cases having to revalidate) executables. This means that last minute changes (because of model build-time parameter tuning) can be used ASAP instead of having to go through a build-test-deploy cycle on each remote platform. As the executables are copied over to the EC2 cluster (a cheap operation) there is no associated cost of rebundling virtual images.
- EC2 allows our virtual clusters to scale at will: There is a default 20 instance limit (which correspond to a maximum configuration of 160 cores) but if needed it can be increased upon request.
- Since the remote machines are under our complete control, scheduling software and policies etc. can be tuned exactly to our needs.

At the same time use of EC2 is not without its problems:

- Unlike the case of shared state or national resources that come out of research grant related allocations, EC2 usage needs to be directly paid to Amazon.
- Amazon charges by the hour - much like cell-phone charges usage of 1 hour 1 sec. counts as 2 hours. Moreover Amazon charges for data movement in and out of EC2.
- The performance of virtual machines is less than that of "bare metal", the difference being more pronounced when it comes to I/O.
- Unlike purpose-build parallel clusters, EC2 does not offer a persistent large parallel filesystem. One can be constructed on demand (just like the virtual clusters) but the Gigabit Ethernet connectivity used throughout Amazon EC2 alongside the randomization of instance placement mean that parallel performance of the filesystem is not up to par.
- Moreover, unlike national and state supercomputing facilities, Amazon's connections to the *home* cluster are bound to be slower and result in file transfer delays.

The substandard (single port Gigabit Ethernet based) interconnect that an EC2 virtual cluster provides should not be so much of an issue for future ESSE ensembles employing nested calculations: Two-way nesting would be run on 2-core instances (or two.four of them could be "packed" on 4/8-core instances), utilizing shared memory for fast communications between the nested models. Even the far less commonly used three-way nesting could run faster on oversubscribed 2-core instances than spread over multiple nodes.

6 Future work

We plan to fine tune our ESSE workflows for production using the Teragrid as well as test them on the Open Science Grid. We would like to investigate the efficacy of a data scheduler such as Stork to help us with prestaging input data. We also plan to test the feasibility of a mixed local/Grid/EC2 run employing MyCluster. Future more involved experiments are expected to scale from 1000 to 10000 or more ESSE ensemble members (and even more acoustic calculations). We are interested in seeing how queuing systems and resource managers handle such a workload in a short time interval. Furthermore more realistic model setups are expected to require the use of the new nested and unstructured MSEAS calculations [34], [35] which are executed in parallel - thereby introducing the concept of massive ensembles of small (2-3 task) MPI jobs. We plan to simplify the use of such setups via the use of an XML driven validating graphical user interface [36].

Another area where MTC would be most valuable is the intelligent coordination of autonomous ocean sampling networks. To achieve optimal and adaptive sampling [37], [38], [39], [40], [41], large-dimensional nonlinear stochastic optimizations, artificial intelligence and advance Markovian estimation systems can be required. Such complex systems are prime examples of MTC problems that can be combined with our uncertainty estimations [42] and feature model initialization [43].

7 Conclusion

We described a new type of Many-Task Computing application that is very relevant to Earth and Environmental Science applications (and prototypical of a general class of ensemble-based forecasting and estimation methods). We introduced the concept of ocean data assimilation, discussed the ESSE algorithm and described its MTC implementation (and its variations along with their justification). Results on a local cluster were presented along with a discussion of the challenges of scaling out and solutions for doing so employing

Grids and Clouds. I/O locality issues are among our main concern. We believe that this type of ensemble based forecast workflows can in the future represent an important new class of MTC applications.

ACKNOWLEDGMENTS

PFJL, PJH and JX thank the ONR for partial support under grant N00014-08-1-1097, N00014-07-1-0501 and N00014-08-1-0586. CE and CNH are grateful to the NSF for support and Amazon for an educational grant.

REFERENCES

[1] I. Raicu, I. Foster, , and Z. Y., "Many-task computing for grids and supercomputers," in *Proceedings of "IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08)"*. IEEE, 2008.

[2] C. L. Keppenne, "Data assimilation into a primitive-equation model with a parallel ensemble kalman filter," *Mon. Wea. Rev.*, vol. 128, pp. 1971–1981, June 2000.

[3] C. L. Keppenne and M. M. Rienecker, "Initial testing of a massively parallel ensemble kalman filter with the poseidon isopycnal ocean general circulation model." *Mon. Wea. Rev.*, vol. 130, pp. 2951–2965, 2002.

[4] J. Anderson and N. Collins, "Scalable implementations of ensemble filter algorithms for data assimilation," *J. of Atmos. Oceanic Tech.*, vol. 24, no. 8, pp. 1452–1463, August 2007.

[5] P. J. F. Lermusiaux, "Uncertainty estimation and prediction for interdisciplinary ocean dynamics," *Journal of Computational Physics*, pp. 176–199, 2006, in special issue on *"Uncertainty Quantification"*. J. Glimm and G. Karniadakis, Eds.

[6] P. F. J. Lermusiaux, "Adaptive modeling, adaptive data assimilation and adaptive sampling," *Physica D*, vol. 230, pp. 172–196, 2007, in special issue on *"Mathematical Issues and Challenges in Data Assimilation for Geophysical Systems: Interdisciplinary Perspectives"*, C. K. R. T. Jones and K. Ide, Eds.

[7] Y. Gu and R. Grossman, "Exploring data parallelism and locality in wide area networks," in *1st Workshop on Many-Task Computing on Grids and Supercomputers*, 2008.

[8] D. Abramson, B. Bethwaite, C. Enticott, S. Garic, and T. Peachley, "Robust workflows for science and engineering," in *2nd Workshop on Many-Task Computing on Grids and Supercomputers*, 2009.

[9] E. Ogasawara, D. De Oliveira, F. Seabra, C. Barbosa, R. Elias, A. Coutinho, and M. Mattoso, "Exploring many task computing in scientific workflows," in *2nd Workshop on Many-Task Computing on Grids and Supercomputers*, 2009.

[10] X. S. Liang and A. R. Robinson, "A study of the Iceland-Faeroe frontal variability with the multiscale energy and vorticity analysis," *J. Phys. Oceanogr.*, vol. 34, pp. 2571–2591, 2004.

[11] X. S. Liang and D. G. Anderson, "Multiscale window transform," *SIAM J. Multiscale Model. Simul.*, vol. 6, pp. 437–467, 2007.

[12] P. F. J. Lermusiaux, C.-S. Chiu, G. G. Gawarkiewicz, P. Abbot, A. R. Robinson, R. N. Miller, P. J. Haley, Jr., W. G. Leslie, S. J. Majumdar, A. Pang, and F. Lekien, "Quantifying uncertainties in ocean predictions," *Oceanography*, vol. 19, no. 1, pp. 92–105, 2006, in special issue on *"Advances in Computational Oceanography"*, T. Paluszkiewicz and S. Harper, Eds.

[13] P. F. J. Lermusiaux and C.-S. Chiu, "Four-dimensional data assimilation for coupled physical-acoustical fields," in *Acoustic Variability*, ser. SACLANTCEN, N. Pace and F. Jensen, Eds. Kluwer Academic Press, 2002, pp. 417–424.

[14] J. Xu, P. F. J. Lermusiaux, P. J. Haley, Jr., W. G. Leslie, and O. G. Logutov, "Spatial and temporal variations in acoustic propagation during the plusnet07 exercise in dabob bay," in *Proceedings of Meetings on Acoustics (POMA), 4, 155th Meeting*. Acoustical Society of America, 2008.

[15] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[16] I. K., P. Courtier, M. Ghil, and A. C. Lorenc, "Unified notation for data assimilation: operational, sequential and variational," *J. of the Meteorol. Soc. of Japan*, vol. 75, no. 1B, pp. 181–189, 1997.

[17] A. Gelb, Ed., *Applied Optimal Estimation*. Cambridge, MA.: MIT Press, 1974.

[18] C. Gardiner, *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*. Springer-Verlag, 1983, 442 pages.

[19] P. F. J. Lermusiaux and A. R. Robinson, "Data assimilation via error subspace statistical estimation, part i: theory and schemes." *Mon. Wea. Rev.*, vol. 127, no. 7, pp. 1385–1407, 1999.

[20] P. F. J. Lermusiaux, A. Robinson, P. J. Haley, Jr., and W. G. Leslie, "Advanced interdisciplinary data assimilation: Filtering and smoothing via error subspace statistical estimation," in *Proceedings of "The OCEANS 2002 MTS/IEEE" conference*, IEEE. Holland Publications, 2002, pp. 795–802.

[21] P. F. J. Lermusiaux, D. G. M. Anderson, and C. J. Lozano, "On the mapping of multivariate geophysical fields: Error and variability subspace estimates," *Q.J.R. Meteorol. Soc*, vol. 126, pp. 1387–1429, 2000.

[22] P. F. J. Lermusiaux, "On the mapping of multivariate geophysical fields: sensitivity to size, scales and dynamics," *J. of Atmos. Oceanic Tech.*, vol. 19, pp. 1602–1637, 2002.

[23] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1985, 561 pages.

[24] ——, *Topics in Matrix Analysis*. Cambridge University Press, 1985, 607 pages.

[25] W. Gentzsch, "Sun grid engine: Towards creating a compute power grid," in *Proceedings of 1st IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*. Washington, DC: IEEE Computer Society, 2001, p. 35, (May 15 - 18, 2001).

[26] T. D., T. T., and L. M., "Distributed computing in practice: the condor experience," *Concurr. Comput. : Pract. Exper.*, vol. 17, no. 2–4, pp. 323–356, February 2005, research Articles.

[27] H. L. Jenter and R. P. Signell, "Netcdf: A public-domain-software solution to data-access problems for numerical modelers," 1992.

[28] J. Layton, "Strace analyzer." [Online]. Available: http://clusterbuffer.wetpaint.com/page/Strace+Analyzer

[29] R. Ross and R. Latham, "Pvfs: a parallel file system," in *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*. New York, NY, USA: ACM, 2006, p. 34.

[30] G. Developers, "The Gluster web site," 2008, http://www.gluster.org.

[31] E. Walker, "A distributed file system for a wide-area high performance computing infrastructure," in *Proceedings of the 3rd conference on USENIX Workshop on Real, Large Distributed Systems, Seattle, WA*, vol. 3. Berkeley, CA: USENIX Association, 2006, pp. 9++.

[32] M. E. Hoskins, "Sshfs: super easy file access over ssh," *Linux Journal*, vol. 146, p. 4, June 2006.

[33] E. Walker, P. J. Gardner, V. Litvin, and E. Turner, "Personal adaptive clusters as containers for scientific jobs," *Cluster Computing*, vol. 10, no. 3, September 2007.

[34] P. J. Haley, Jr. and P. F. J. Lermusiaux, "Multiscale two-way embedding schemes for free-surface primitive-equations in the Multidisciplinary Simulation, Estimation and Assimilation System (MSEAS)," *Ocean Dynamics*, 2010, under review.

[35] M. P. Ueckermann and P. F. J. Lermusiaux, "High order schemes for 2D unsteady biogeochemical ocean models," *Ocean Dynamics*, 2010, under review.

[36] C. Evangelinos, P. F. J. Lermusiaux, S. Geiger, R. C. Chang, and N. M. Patrikalakis, "Web-enabled configuration and control of legacy codes: An application to ocean modeling," *Ocean Modeling*, vol. 13, pp. 197–220, 2006.

[37] P. F. J. Lermusiaux, P. J. Haley, Jr., and N. K. Yilmaz, "Environmental prediction, path planning and adaptive sampling: Sensing and modeling for efficient ocean monitoring, management and pollution control," *Sea Technology*, vol. 48, no. 9, pp. 35–38, 2007.

[38] K. D. Heaney, G. Gawarkiewicz, T. F. Duda, and P. F. J. Lermusiaux, "Non-linear optimization of autonomous undersea vehicle sampling strategies for oceanographic data-assimilation," *Journal of Field Robotics*, vol. 24, no. 6, pp. 437–448, 2007, in special issue on *"Underwater Robotics"*.

[39] Y. N. K., C. Evangelinos, P. F. J. Lermusiaux, and N. M. Patrikalakis, "Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 4, pp. 522–537, 2008.

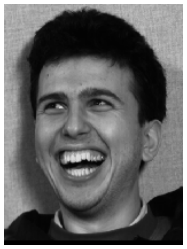
[40] D. Wang, P. F. J. Lermusiaux, P. J. Haley, Jr., D. Eickstedt, W. G. Leslie, and H. Schmidt, "Acoustically focused adaptive sampling and on-board routing for marine rapid environmental assessment," *Journal of Marine Systems*, 2009, in special issue on *"Coastal*

Processes: Challenges for Monitoring and Prediction", J.W. Book, M. Orlic and M. Rixen, Eds.

[41] F. P. Lam, P. J. Haley, Jr., J. Janmaat, P. F. J. Lermusiaux, W. G. Leslie, M. W. Schouten, L. A. te Raa, and M. Rixen, "At-sea real-time coupled four-dimensional oceanographic and acoustic forecasts during battlespace preparation 2007," *Journal of Marine Systems*, 2009, in special issue on *"Coastal processes: Challenges for Monitoring and Prediction"*, J. W. Book, M. Orlic and M. Rixen, Eds.

[42] T. P. Sapsis and P. F. J. Lermusiaux, "Dynamically orthogonal field equations for continuous stochastic dynamical systems," *Physica D*, 2009.

[43] A. Gangopadhyay, A. R. Robinson, P. J. Haley, Jr., W. G. Leslie, C. J. Lozano, J. J. Bisagni, and Y. Z. T., "Feature-oriented regional modeling and simulations in the Gulf of Maine and Georges Bank," *Continental Shelf Research*, vol. 23, no. 3–4, pp. 317–353, 2003.



Constantinos Evangelinos received the B.A. degree (with honors) in mathematics from Cambridge University, Cambridge, U.K., in 1993 and the Sc.M. and Ph.D. degrees in applied mathematics from Brown University, Providence, RI, in 1994 and 1999, respectively. Currently, he is a Research Scientist in the Earth, Atmospheric and Planetary Sciences Department, Massachusetts Institute of Technology, Cambridge, where he works on ocean state estimation. His current research interests are in computational methods for variational as well as sequential data assimilation, adjoint techniques and automatic differentiation, parallel/grid computing approaches and performance modeling applied to grand challenge applications in the ocean sciences. Dr. Evangelinos is a member of the IEEE Computer Society and the Association for Computing Machinery (ACM).

Pierre F.J. Lermusiaux received B.&M. in Mech. Eng. degrees (highest honors and Jurys congratulations) from Liege University, Liege, Belgium, in 1992 and the S.M. and Ph.D. degrees in engineering sciences from Harvard University, Cambridge, MA, in 1993 and 1997. Currently, he is an Associate Professor of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge. His current research interests include physical and interdisciplinary ocean dynamics, from submesoscales to interannual scales. They involve physicalbiogeochemicalacoustical ocean modeling, optimal estimation and data assimilation, uncertainty and error modeling, and the optimization of observing systems. Dr. Lermusiaux has held Fulbright Foundation Fellowships, was awarded the Wallace Prize at Harvard in 1993, presented the Ogilvie Young Investigator Lecture in Ocean Engineering at MIT in 1998 and was awarded a Doherty Associate Professorship in Ocean Utilization by MIT in 2009. He is a member of the Association of Engineers of Liege University, Friends of the University of Liege, Royal Meteorological Society, American Geophysical Union, Oceanography Society, American Association for the Advancement of Science and Society for Industrial and Applied Mathematics.

Jinshan Xu received the B.A. degree in electrical engineering and the Sc.M. in physical oceanography from University of Qingdao, Qingdao, China, in 1994 and 1999, and the Ph.D. degrees in mechanical and oceanographic engineering from Massachusetts Institute of Technology/Woods Hole Oceanographic Institute Joint Program in Oceanography, in September 2007. Currently, he is a Postdoctoral Research Associate in the Mechanical Engineering Department, Massachusetts Institute of Technology, Cambridge. His research interests are in the areas of underwater acoustic propagation modeling and signal processing, wave propagation through random media. Dr. Xu is a member of the Acoustical Society of America.

Patrick J. Haley Jr. received the B.S. degree in physics from Siena College, Loudonville, NY. in 1984 and the Ph.D. degree in applied mathematics from Northwestern University, Evanston, IL. in 1991. Currently he is a research scientist in the Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge. His current research interests are in multiscale, interdisciplinary ocean forecasting and modeling, and the development of ocean observing and prediction systems.

PLACE PHOTO HERE

Chris N. Hill received his B.Sc. degree in Physics from Imperial College, London and spent several years studying network flow in electrical transmission grids before joining MIT in 1993. Currently, he is a Principal Research Scientist in the Earth, Atmospheric and Planetary Sciences Department, Massachusetts Institute of Technology, Cambridge, where he works on the application of parallel computing to ocean and atmospheric modeling.