



## 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 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. 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.

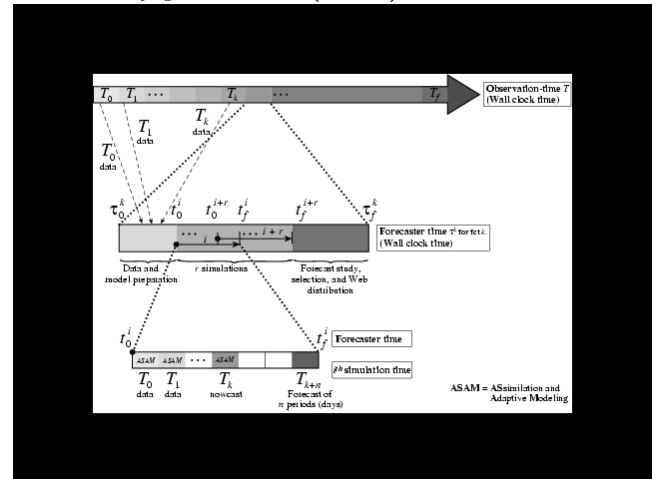
Realistic simulations of four-dimensional ocean fields are carried out over broad numerical domains, e.g.  $O(10-1000)$  km for  $O(10-1000)$  days. The number of grid points and thus of discretized state variables are very large, usually of  $O(10^5 - 10^7)$ . On the other hand, ocean data are limited 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 [13]. 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  $\tau_0^k$  to  $\tau_f^k$ ). These tasks include the processing of the currently available data and model (from  $\tau_0^k$  to  $\tau_0^i$ ), 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  $\tau_f^k$ ). 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  $i$ th 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  $T_{k+n}$ .

**Figure 1: Forecasting timelines.** Top row: “Observation” or “ocean” time  $T$  during which measurements are made and the real phenomena occur. Middle row: “Forecaster” time  $\tau^k$  during which the  $k$ th forecasting procedure and tasks are started and finished. Bottom row: “ $i$ th 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).



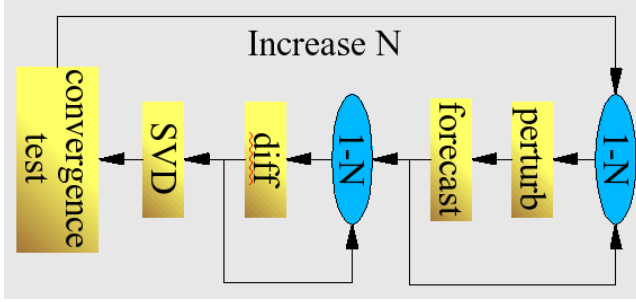
### 2.2 Ocean Acoustics

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 [12, 23].

Having an estimate of the ocean temperature and salinity fields (along with their respective uncertainties) provides the required background information for calculating acous-



Figure 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:

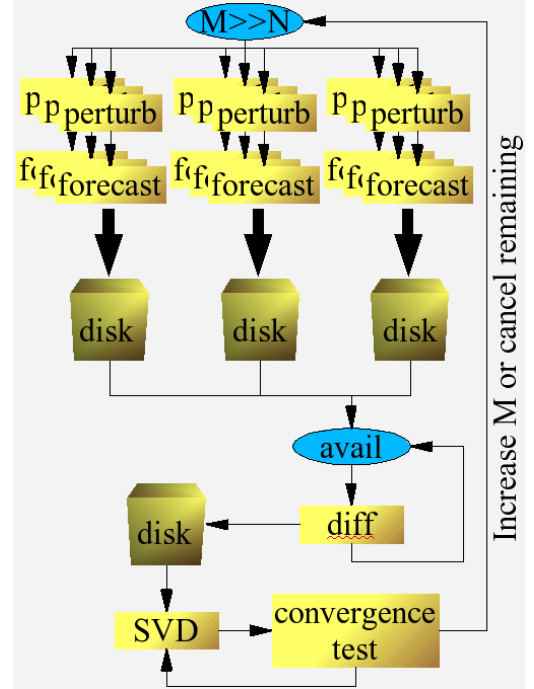
1. The perturb/forecast loop needs to finish for the diff loop to start (or the two loops can be fused (merged). Either way there is no exposed parallelism.
2. The diff loop has a serial bottleneck (the same file is written to). Depending on the variant of the perturbation type employed, it may also 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 calculations 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).

Figure 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 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) [4] or Condor [19]. The shell script catches the kill signal in 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. 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. 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 if the ESSE execution gets stopped, it can only be restarted without rerunning all jobs by switching to a one-job submission per perturbation index strategy.

## 5. ESSE AS AN MTC APPLICATION IN PRACTICE

### 5.1 Special ESSE needs

ESSE and other similar ensemble-based ocean forecasting methodologies are used a 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 desk-side 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 repre-

sentations 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 Local cluster description

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 fileserver serves over 18TB of shared disk 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 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 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 [20], SSHFS [6] 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 [21] software that makes a collection of remote and local resources appear as one large Condor or SGE controlled cluster. This way we are not limited to Condor but we can use our SGE-based setup instead.

#### 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 gpfswan and then a fast distribution from gpfswan 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 1.

As one can see speeds vary appreciably (and a recompilation, however inconvenient it may be - especially for a last minute change of code) can be well worth it. The slow *pert* performance for ORNL appears to be partly related to the PVFS2 filesystem used. In practice this means that the more disparate the hosts used to augment the local compute facilities, the more uneven the progress of the various remote

**Table 1: 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 |

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 to using the Grid to augment local compute resources for ESSE:

- There are a great many computational resources available on the Grid allowing for . 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.
- 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 to 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 2: 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     |

#### 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.

An idea of the times of EC2 instances running pemodel and pert for various instance types is shown in Table 2.

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% 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.

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:  $1.5(GB) \times 0.1 + 10.56(GB) \times 0.17 + 2(hr) \times 20 \times 0.8 = \$33.95$  Use of reserved instances would drop pricing for the cpu usage by more than a factor of 3.

#### 5.4.3 Evaluating ESSE on EC2

There are quite a few clear advantages to 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.
- 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 remove 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 it's 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.

## 6. EXAMPLE ESSE RESULTS FOR MONTEREY BAY

A large Office of Naval Research (ONR)-sponsored, multi-institution coastal predictive skill exercise, the Autonomous Ocean Sampling Network-II (AOSN-II), occurred in August-September 2003 in the Monterey Bay region off central California. The goal of this exercise was to initiate at-sea research of an adaptive observing and prediction system, with the intent to assimilate various data types, adapt the deployment of platforms and allow the relocation of the system to other regions. The Harvard Ocean Prediction System (HOPS) and Error Subspace Statistical Estimation (ESSE) system were utilized in real-time to forecast physical fields and uncertainties, assimilate various ocean measurements (CTD, AUVs, gliders and SST data), provide suggestions for adaptive sampling, and guide dynamical investigations.

To exercise our new MTC implementation of ESSE, we repeated the calculations of AOSN-II. The ESSE forecast for September 5, 00:00 GMT was initialized from an error nowcast for September 3, 00:00 GMT. The background ocean field on September 3, 00:00 GMT is a HOPS forecast simulation which assimilates all available and calibrated data up to September 2, 10:00 GMT.

The dominant 600 eigenvectors of the posterior error covariance estimate for September 3, 0000 GMT were utilized to perturb the ocean fields. A white noise of an amplitude proportional to the estimated absolute and relative errors in the observations is added to this random combination, in part to represent the errors truncated by the error subspace. An ensemble of forecast simulations, each forced by forecast COAMPS atmospheric fluxes issued on September 2, was then carried out. These ensemble members were then utilized to compute the standard deviations shown in Figs. 5,6.

Figure 5: ESSE uncertainty forecast for sea-surface temperature ( $^{\circ}\text{C}$ )

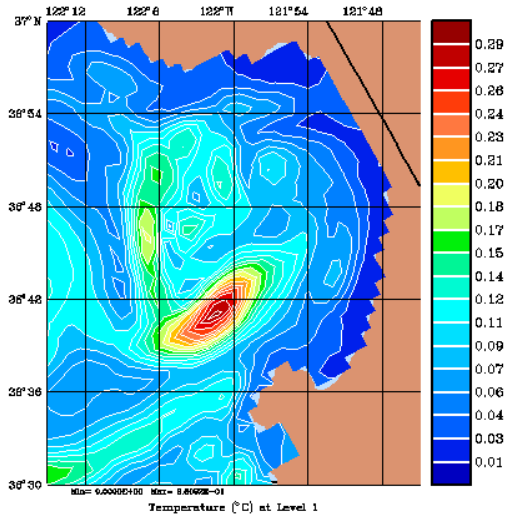
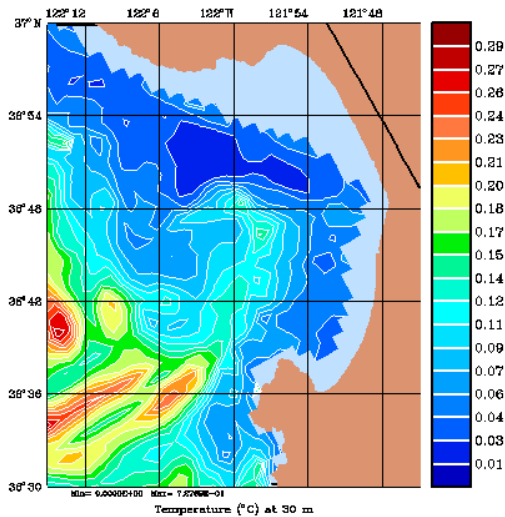


Figure 6: ESSE uncertainty forecast for 30m temperature ( $^{\circ}\text{C}$ )



## 7. 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 prestage 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 nested HOPS calculations 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 [1].

Another area where MTC would be most valuable is the intelligent coordination of autonomous ocean sampling networks. To achieve optimal and adaptive sampling [5, 9, 14, 22, 24], 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 [18].

## 8. 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.

## 9. ACKNOWLEDGMENT

PFJL, PJH and JX are grateful to 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.

## 10. REFERENCES

- [1] Evangelinos, C., Lermusiaux, P.F.J., Geiger, S., Chang, R.C. and Patrikalakis, N.M. 2006. Web-Enabled Configuration and Control of Legacy Codes: An Application to Ocean Modeling. *Ocean Modeling*, 13, 197-220.
- [2] Gardiner, C.W. 1983. *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*, Springer-Verlag, 442 pages.
- [3] Gelb, A. (ed.) 1974. *Applied Optimal Estimation*, MIT Press, Cambridge, MA.
- [4] Gentsch, W. 2001. Sun Grid Engine: Towards Creating a Compute Power Grid. In *Proceedings of 1st IEEE International Symposium on Cluster Computing*

- and the Grid (May 15 - 18, 2001). CCGRID. IEEE Computer Society, Washington, DC, 35.
- [5] Heaney, K.D., Gawarkiewicz, G., Duda, T.F. and Lermusiaux, P.F.J. 2007. Non-linear Optimization of Autonomous Undersea Vehicle Sampling Strategies for Oceanographic Data-Assimilation. In special issue on "Underwater Robotics", *Journal of Field Robotics*, 24(6), 437-448.
- [6] Hoskins, M.E. 2006. SSHFS: super easy file access over SSH. *Linux J.*, 146 (Jun. 2006), 4.
- [7] Ide, K., Courtier, P., Ghil, M., and Lorenc A.C. 1997. Unified notation for data assimilation: operational, sequential and variational. *J. of the Meteorol. Soc. of Japan*, 75(1B), 181-189.
- [8] Jazwinski, A.H. 1970. *Stochastic Processes and Filtering Theory*, Academic Press.
- [9] Lam, F.P, Haley Jr., P.J., Janmaat, J., Lermusiaux, P.F.J., Leslie, W.G., Schouten, M.W., te Raa, L.A. and Rixen, M. 2009. At-sea Real-time Coupled Four-dimensional Oceanographic and Acoustic Forecasts during Battlespace Preparation 2007. In special issue of the *Journal of Marine Systems on "Coastal processes: Challenges for Monitoring and Prediction"*, J.W. Book, M. Orlic and M. Rixen, Eds.. doi:10.1016/j.jmarsys.2009.01.029
- [10] Lermusiaux, P.F.J. 2006. Uncertainty Estimation and Prediction for Interdisciplinary Ocean Dynamics. In special issue on "Uncertainty Quantification". J. Glimm and G. Karniadakis, Eds., *Journal of Computational Physics*, 176-199.
- [11] Lermusiaux, P.F.J. 2007. Adaptive Modeling, Adaptive Data Assimilation and Adaptive Sampling. In special issue on "Mathematical Issues and Challenges in Data Assimilation for Geophysical Systems: Interdisciplinary Perspectives", *Physica D*, C.K.R.T. Jones and K. Ide Eds., 230, 172-196.
- [12] Lermusiaux, P.F.J. and Chiu, C.-S. 2002. Four-dimensional data assimilation for coupled physical-acoustical fields. In *Acoustic Variability*, N.G. Pace and F.B. Jensen, Eds., SACLANTCEN. Kluwer Academic Press, 417-424.
- [13] Lermusiaux, P.F.J., Chiu, C.-S., Gawarkiewicz, G.G., Abbot, P., Robinson, A.R., Miller, R.N., Haley Jr., P.J., Leslie, W.G., Majumdar, S.J., Pang, A. and Lekien, F. 2006. Quantifying Uncertainties in Ocean Predictions. In special issue on "Advances in Computational Oceanography", *Oceanography*, T. Paluszkiwicz and S. Harper, Eds., 19(1), 92-105.
- [14] Lermusiaux, P.F.J., Haley Jr., P.J., and Yilmaz, N.K. 2007. Environmental Prediction, Path Planning and Adaptive Sampling: Sensing and Modeling for Efficient Ocean Monitoring, Management and Pollution Control. *Sea Technology*, 48(9), 35-38.
- [15] Lermusiaux, P.F.J. and Robinson, A.R. 1999. Data assimilation via error subspace statistical estimation, Part I: theory and schemes. *Mon. Wea. Rev.* 127(7), 1385-1407.
- [16] Lermusiaux, P.F.J., Robinson, A.R., Haley Jr., P.J., and Leslie, W.G. 2002. Advanced interdisciplinary data assimilation: Filtering and smoothing via Error Subspace Statistical Estimation. In *Proceedings of "The OCEANS 2002 MTS/IEEE" conference*, IEEE, Holland Publications, 795-802.
- [17] Raicu, I., Foster, I., and Zhao Y. 2008. Many-Task Computing for Grids and Supercomputers, *Proceedings of "IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08)"*.
- [18] Sapsis, T.P. and Lermusiaux, P.F.J. 2009. Dynamically orthogonal field equations for continuous stochastic dynamical systems. *Physica D*, doi:10.1016/j.physd.2009.09.017.
- [19] Thain D., Tannenbaum T. and Livny M. 2005. Distributed computing in practice: the Condor experience. Research Articles. *Concurr. Comput. : Pract. Exper.* 17, 2-4 (Feb. 2005), 323-356. doi:10.1002/cpe.v17:2/4
- [20] Walker, E. 2006. 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 - Volume 3* (Seattle, WA). USENIX Association, Berkeley, CA, pp. 9
- [21] Walker, E., Gardner, P.J., Litvin, V., and Turner, E. 2007. Personal Adaptive Clusters as Containers for Scientific Jobs. *Cluster Computing*, 10(3), Sept. 2007.
- [22] Wang, D., Lermusiaux, P.F.J., Haley Jr., P.J., Eickstedt, D., Leslie, W.G. and Schmidt, H. 2009. Acoustically Focused Adaptive Sampling and On-board Routing for Marine Rapid Environmental Assessment. In special issue of the *Journal of Marine Systems on "Coastal Processes: Challenges for Monitoring and Prediction"*, J.W. Book, M. Orlic and M. Rixen, Eds. doi:10.1016/j.jmarsys.2009.01.037.
- [23] Xu, J., Lermusiaux, P.F.J., Haley Jr., P.J., Leslie, W.G. and Logutov, O.G. 2008. 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, doi:10.1121/1.2988093.
- [24] Yilmaz N.K., Evangelinos, C., Lermusiaux, P.F.J. and Patrikalakis, N.M. 2008. Path Planning of Autonomous Underwater Vehicles for Adaptive Sampling Using Mixed Integer Linear Programming. *IEEE Transactions, Journal of Oceanic Engineering.* 33 (4), 522-537. doi:10.1109/JOE.2008.2002105