# Implementation of XIOS in the Atmospheric Component of the Unifed Model
# In-flight ensemble processing for exascale

Bryan Lawrence[a,b,c,*], Grenville Lister[a,b,*], Jeff Cole[a,b], Yann Meursdesoif[d], Rupert Nash[e], Michele Weiland[e]

[a]*National Centre for Atmospheric Science, UK*
[b]*Department of Meteorology, University of Reading, UK*
[c]*Department of Computer Science, University of Reading, UK*
[d]*LSCE/IPSL, Commissariat for Atomic Energy and Alternative Energies (CEA), France*
[e]*EPCC, The University of Edinburgh, UK*

**Abstract**

Weather and climate science make heavy use of ensembles of model simulations to provide estimation of uncertainty arising from a range of causes (e.g. [1]). Current practice is to write each ensemble member (simulation) out to disk as it is running, and carry out an ensemble analysis at the end of the simulation. Such analysis will include simple statistics, as well as detailed analysis of some ensemble members. However, as model resolutions increase (with more data per simulation), and ensemble sizes increase (more instances), the storage and analysis of this data is becoming prohibitively expensive — many major weather and climate sites are looking at managing in excess of an exabyte of data within the next few years. This becomes problematic for an environment where we anticipate running such ensembles on exascale machines which may not themselves include local storage of sufficient size where data can be resident for long periods of analysis.

There are only two possible strategies to cope with this data deluge: data compression (including "thinning", that is the removal of data from the output) and in-flight analysis. We discuss here some first steps with the latter approach. We exploit the XML IO server (XIOS, [2]) to manage the output from simulations and to carry out some initial analysis en-route to storage.

We have achieved three specific ambitions: (1) We have adapted a current branch of the Met Office Unified Model to replace much of the diagnostic system with the XIOS. (2) We have exploited a single executable MPI environment to run multiple UM instances with output sent to XIOS, and (3) We have demonstrated that simple ensemble statistics can be calculated in-flight, including both summary statistics of individual ensemble members, and cross-member statistics such as means and extremes.

With this ability, we can in principle avoid having all data needing to reside on fast disk when the ensemble simulation is complete. This would allow, for example, deployment on an exascale machine with burst-buffer migrating data directly to tape (or to the wide area network).

*Keywords:* ensemble

## 1. Introduction

As increasing computer power has become available, the weather and climate community have put effort into increasing the spatial resolution within the simulation domain, increasing the domain of simulation (spatial and/or temporal), expanded use of, or complexity of, data assimilation to improve initial conditions, and increasing the number of simulations within "ensembles". Such ensembles are collections of simulations which address the same problem, but where some key aspect of each simulation differs from the others.

Ensembles typically vary along one or more of four axes: initialisation, boundary conditions, physical parameters, and modelling system (aka "Model"). Initialisation ensembles are the mainstay of numerical weather prediction — choosing the right set of initialisations is a science in and of itself [3]. Ensembles along the other dimensions are more normally used in climate science (e.g. the coupled model intercomparison projects such as CMIP6, [4]), but are also increasingly used at shorter timescales for weather related problems. In all cases, the ensembles are used to sample uncertainty — see [1] for an example from climate science, and [5, 6] for a more philosophical and statistical discussions on the use of ensembles for uncertainty evaluation.

Whatever the use, the workflows associated with ensembles are becoming problematic: when ensemble simulations are run in parallel on the same platform they can swamp the available bandwidth to, and volume available at, local storage systems. When they are run sequentially, the volume issues remain, and workflow and queuing delays become new hurdles to surmount. Whether simulations are run sequentially or in parallel the current mode of usage is to write out all the data from each simulation and post-process to produce "ensemble statistics", and

*Corresponding Author

*Email addresses:* `bryan.lawrence@ncas.ac.uk` (Bryan Lawrence), `grenville.lister@ncas.ac.uk` (Grenville Lister), `jeff.cole@ncas.ac.uk` (Jeff Cole), `yann.meurdesoif@cea.fr` (Yann Meursdesoif), `r.nash@epcc.ed.ac.uk` (Rupert Nash), `m.weiland@epcc.ed.ac.uk` (Michele Weiland)

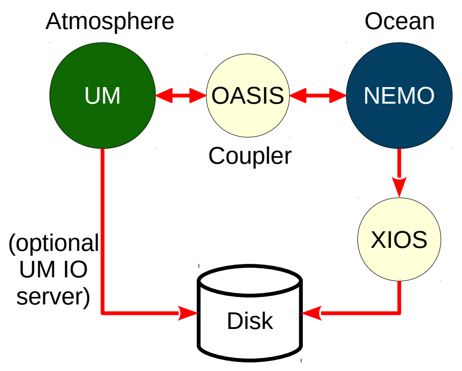


Figure 1: Schematic of the existing UK climate mode.

at many sites this can be difficult, especially for large, long, or high-resolution ensembles. Managing the workflows has become an exercise in logistics [7]. These ensembles are also a major contributor to the vast archives of data held at major weather and climate sites, and together with the cost of adequate disk, the cost of even tape archive is becoming problematic. All these problems are expected to increase in the next few years, not only is there increasing scientific focus on large ensembles (e.g. [8]), problems with getting models ready for exascale [9] will probably mean the first usage of exascale machines in weather and climate will be for large ensembles.

There are only two possible strategies to cope with this data deluge - data compression (including “thinning”, that is the removal of data from the output) and in-flight analysis.

In this paper we introduce some first steps with the latter approach. In particular, we demonstrate the use of a single executable climate ensemble system developed so that each member simulation shares access to an extra component - an IO server - which carries out data reductions before writing data to disk.

Section 2 discusses the climate model environment and introduces the particular IO server that is being used. Section 3 introduces the software that was developed to support the ensemble system and section 4 reports on our experiences with the system. Section 5 summarises our achievements and signposts some of our plans for further work.

## 2. Context

Climate modelling in the UK is predominantly done using variants of the Unified Model, running standalone in “atmosphere only mode” or in coupled mode [e.g. 10, and references therein]. The coupled variant is depicted in Figure 1: the UM atmosphere is coupled via the OASIS coupler [11] to a NEMO [12] ocean. NEMO uses the XIOS [2] to write to disk, and the UM atmosphere either writes directly to disk, or uses an internal UM IO server.

The XIOS already has support for user configurable data reductions, and has already been used to manage ensembles [13], so was a natural target for investigating ensemble support.

### *2.1. The Unified Model*

Developed jointly by the Met Office, its partner organizations, and the academic climate and weather research community, the UM is the model chiefly used to run numerical simulations in the UK academic and research-centre environments. Much of this work is undertaken on ARCHER, accounting for 140 million core-hours annually. The model is run in a wide range of configurations varying from low-resolution, high-fidelity Earth System models, to very high-resolution climate and ultra high-resolution process studies. The UM is an MPI-OMP code written in FORTRAN, with a small amount of C. The UM has very few software dependencies, principally the communications library (GCOM), which is simply a wrapper for MPI, and, depending on precisely how the model is configured, possible dependencies on NetCDF and HDF5. The UM is managed through a workflow system, which combines a model configuration system Rose [14] and workflow scheduling engine Cylc [15].

#### *2.1.1. Climate Configuration*

While the UM can be configured with time varying sea-surface acillary fields to mimic fully interactive atmosphere-ocean coupling, it is more commonly the case that a fully coupled configuration is employed. Figure 1 outlines the components of the coupled model — the UM (atmosphere) and NEMO (ocean) run asynchronously with a defined coupling frequency. Fields are exchanged between the two components by means of the OASIS coupler, which performs the required conservative regridding between UM (lat-long) and NEMO (tripolar) grids. The UM currently manages its output through its proprietary IO server scheme (UM-IOS) and NEMO uses XIOS to output diagnostics (prognostics are handled differently).

#### *2.1.2. Current UM IO scheme*

The UM diagnostics system, commonly referred to as STASH, has been developed over 25 years into a highly configurable and versatile scheme for extracting and filtering UM fields. Multiple diagnostics are available, but not all are selected for output in any given simulation. Recent advances have seen the successful development of CF-NetCDF capability circumventing the writing data in UM format. Diagnostic output is made available for selection through the Rose GUI, where in addition, it can be configured for temporal and spatial processing. Rose generates FORTAN namelists from user choices — the workflow is sketched in Figure 2 — which are read at runtime to control model execution. The UM IO scheme is a client-server model; diagnostics are processed and buffered on the compute PEs then moved to the servers (UM-IOS in Figure 2) to be written to disc. The number and placement of server processes is configurable.

#### *2.1.3. Resolution and Performance*

Climate models typically run at resolutions, ranging from low (currently in production, N96 with 96x2 points along each latitude circle - with a resolution of 135 km at the equator) to high (currently in production, N512, 512x2 points, 25 km at the

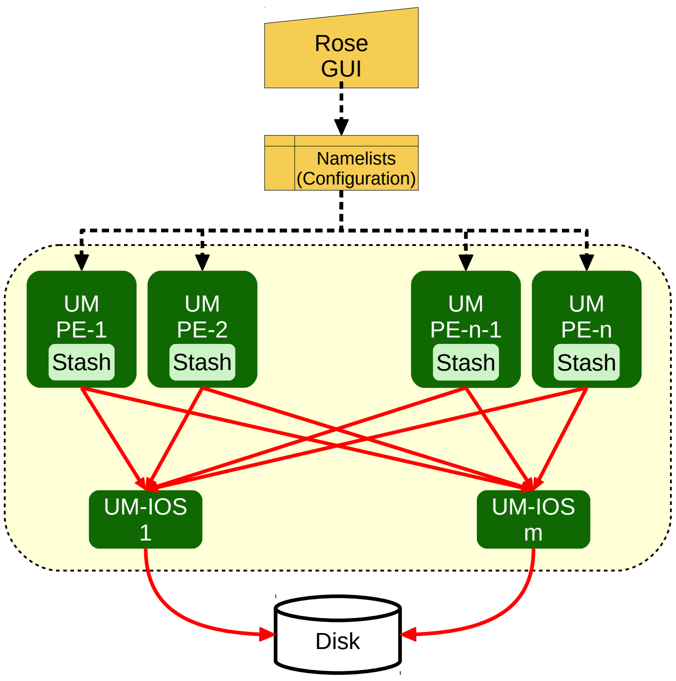


Figure 2: The Unified Model Configuration and output data flow (for an atmosphere only instance): configuration files define the domain decomposition (the number of processing elements, PEs) and the outputs required. Output is written to "STASH" and then, in the IO server configuration, written to multiple IO server instances, which each write to disk. One MPI context (shown in yellow) covers the entire activity. Control flow shown in black, data flow in red.

equator). At the ultra-high reoslution we are currently running N1280 (5km) and developing N2560 (2.5km).

Figure 3 shows the results of some scaling runs for a high-resolution (N512) atmosphere-only climate model, which is essentially the model used throughout this work, on ARCHER and NEXCS (part of the MO Cray XC40), with and without IO. IO was managed through the UM IO servers (see Figure 2), which implement achynchronous IO to write data in the proprietary UM data format. The figure also shows scaling behaviour for a low-resoulution model which includes an expensive atmospheric chemistry scheme. All runs were performed with two OMP threads (required for the UM IO server). The N512 model scales reasonably well out to 15000 cores both and without IO — although changes to the IO profile can have significant impact on performance through increased stalling resulting from inappropriate buffering and/or simply overmatching the machine IO bandwidth (ARCHER being a very heavily loaded machine).

## 2.2. XIOS

The XIOS (XML IO Server) is a software system devloped at the Institute Pierre Simon Laplace (IPSL). XIOS is organized around a heirarchical description of its data through an external XML file providing a simple interface for the user. XIOS provides dedicated servers for asynchronous output to overlap with computation, parallel I/O for single file output and possible performance improvement and simplified down stream data workflow, in addition to multiple file (one per IO server) output. XIOS also offers the prospect of *in situ* data analysis (our use of the phrase *in flight* is based on this potential).

XIOS is 90,000 lines of C++; it is open source software (under CeCiLL licence) and the code is available at `http://forge.ipsl.jussieu.fr/ioserver`. The XIOS build system is based on FCM [16] with support for Intel and Cray compilers, we have built and run it with both. In the work described here, we have primarily used revision 1404.

XIOS is based on a client-server arctitecture, whereby each compute processor interacts with an XIOS client to expose agreed data fields through a mininmalist interface. The set of fields to be exposed is defined by entries in the XML file. A simple FORTRAN call is all that the model needs to do in order to offload data to XIOS, thus:

```
call xios_send_field("field_id", field)
```

where `"field_id"` is a reference to the XML descripion of the field which includes grid and domain information, and `field` is the address of the field data.

XIOS supports multiple data filters and transformations including, importantly for this work, reductions over an axis.

### 2.2.1. XIOS XML Configuration

A key XIOS construct is called a `context`, which may represent a model or a component of a model. In the following XML snippet, we have defined the `atmosphere` context with associated XML elements referencing axis (typically the vertical or pressure axis in a climate model), domain (the horizontal structure), grid (combinations of domains and axes), field (specifying a set of `field_id`s and their associated grids (a given `field_id` can be output on multiple grids)), and file (specifying output frequency, filename, other purely file-related criteria.) The XML file will typically comprise of several contexts, each associated with a logically appropriate model or component.

```
<simulation>
  <context id="atmospere" >
    <axis_definition   src="./axis_def.xml"  />
    <domain_definition src="./domain_def.xml"/>
    <grid_definition   src="./grid_def.xml"  />
    <field_definition  src="./field_def.xml" />
    <file_definition   src="./file_def.xml"  />
  </context>
  <context id="other" >
  ....
  </context>
</simulation>
```

In our experience, the XML configuration for the ocean output does not change much between numerical experiments and simulations, i.e. model output is defined once and hardly changes over time. However, the diagnostic output for atmospheric and coupled climate experments is far more variable, and so the abilty to easily modify the output configuration is important. The existing UM model infrastructure uses Rose for this, and in order to extend the use of XIOS into the atmosphere,

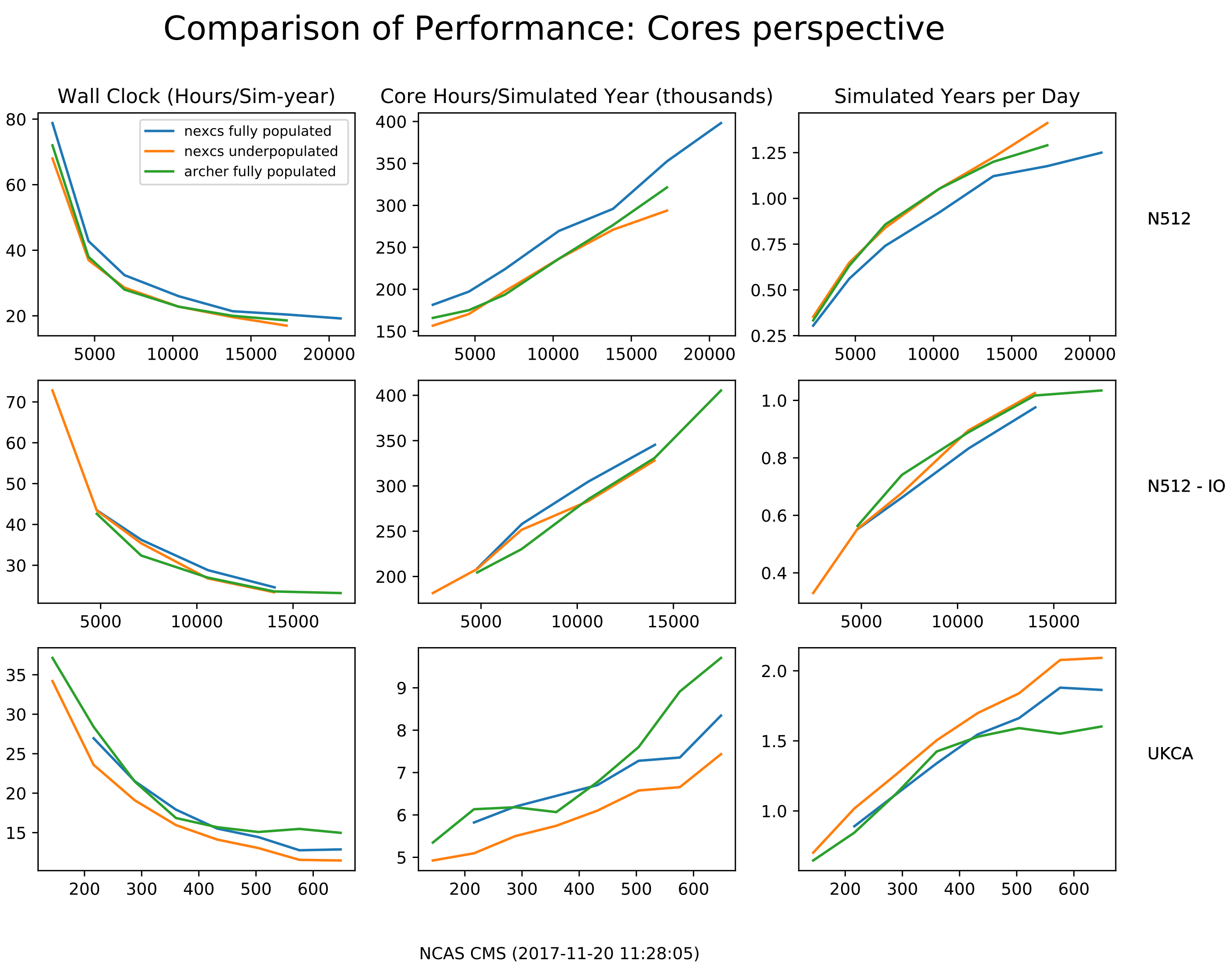


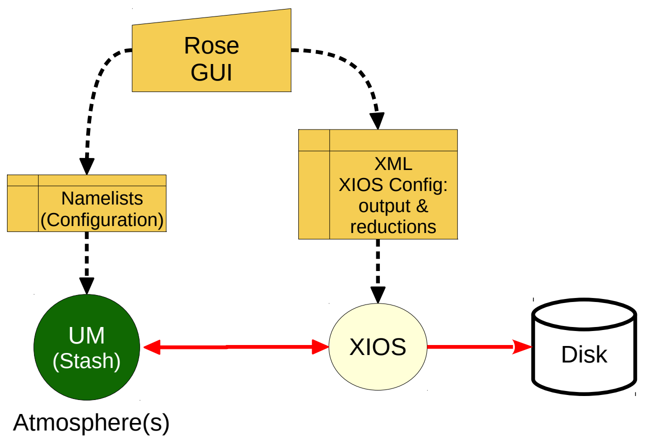
Figure 3: UM atmosphere strong scaling for a high resolution (N512) model with and without IO, and for a low resolution (N96) model with atmospheric chemistry included. All plots display wall clock time against number of cores.

similar diagnostic configuration functionality would be necessary to gain community acceptance.

## 3. Software Modifications

There were four distinct software activities required to develop our ensemble system based on XIOS:

1. The XIOS system had to be inserted alongside the existing diagnostic system (we did not in these experiments completely replace the existing diagnostic system),
2. We had to develop methods of configuring the model ensemble system to request the required diagnostics,
3. We had to work out how to get XIOS to deliver ensemble statstics, and
4. We had work out how to get XIOS to control the ensemble itself.



Rose GUI
Namelists (Configuration)
XML XIOS Config: output & reductions
UM (Stash)
XIOS
Disk
Atmosphere(s)

Figure 4: Maintaining the look and feel of UM diagnostics: the Rose GUI is used to configure both the UM atmosphere via namelists, and the XIOS output. The UM reads the XML files and integrates them with the STASH system and sends the fields to the XIOS server.

### 3.1. *XIOS in the UM*

In inserting XIOS, our goal was to maintian the look and feel of the current system to enable easy user uptake of the new XIOS functionality. Given the maturity of the Rose system and its familarity in the UM user community, the choice was taken to configure XIOS XML directly from Rose. Figure 4 shows the steps to achieve this objective.

There are three elements to the configuration: configuring the ensemble, configuring the ensemble diagnostics, and configuring the output of existing diagnostics via XIOS (although as already noted we did not completely replace the existing output system).

Existing diagnostics are selected by the user in ROSE and associated with three "profiles", namely, time, domain, and usage. The time and domain profiles represent specifications for temporal and spatial processing for the selected diagnostic (time meaning or accumulation, spatial meaning or subspacing, for example). The usage profile, which, in combination with an output stream, determines the file structure for the diagnostic.

This familiar diagnostic interface was extended for use with XIOS by the addition of ensemble configuration and of XIOS output streams to complement regular UM output streams, and the developmemt of a utility to identify diagnostics destined for XIOS and translate their traditional STASH description into XML.

#### 3.1.1. *STASH to XML utility*

The `stash2xiosxml` utility was developed to make the translation from STASH to XML. It is a Python utility which loads configuration files generated by Rose into a heirarchical data structure, extracts and collates information relevant for XIOS diagnostics, cross references with metadata held in a master UM configuration (the "STASHMaster") file to retrieve grid information, and writes the XML. Figure 5 is a schematic of the steps involved in XML creation; Rose creates (possibly many) configuration files which are inputs to `stash2xiosxml`; the XIOS standard XML file iodef.xml is output along with its subcomponents.

Of the 4017 possible diagnostcs availble to a standard atmosphere UM model, the current setup can handle the 3672 which are available on the main model time step. Diagnostics available on other time steps (the radiation time step for example, which is typically a large multiple of the model time step) need special handling and are not covered in this work.

### 3.2. *XIOS UM ensemble*

In order to take full advantage of the existing XIOS reduction capabilities, we introduced an *ensemble axis*. This simply attaches an extra dimension to the data generated by the ensemble to indicate to which ensemble member a particular field belongs. This greatly simplifies `context` management in XIOS, since now the entire ensemble is associated with a single `context` — XIOS simply sees an (n+1)-dimensional data set for the ensemble, where for a single model it saw an n-dimensional data set. Applying reductions over the ensemble axis provides the mechanism for generating ensemble statistics.

Figure 6 describes the key features of our XIOS UM ensemble configuration. From the XIOS viewpoint, the UM ensemble is simply a model — XIOS clients reside on each PE and send fields to XIOS servers. An individual ensemble member runs in its familiar environment — exactly as before.

Several XML files are augmented for the ensemble case; the additonal ensemble axis feeds through to ensemble XML grid, and file definitions. An ensemble grid would appear thus:

```
<grid_definition>
<grid id="um-atmos_grid">
<domain domain_ref="um-atmos_domain" />
<axis axis_ref="um-atmos_vertical" />
<axis axis_ref="ensemble" />
</grid>
</grid_definition>
```

Attaching this grid to a field will result in its output of data for all ensemble members.

### 3.3. *UM code for an ensemble*

For the ensemble case, the model communicator provided by `um_xios_init_mpi` (see section 3.6) is split again for each ensemble member. Individual ensemble member models run on their own model communicators, and as previously for the single model case, non-XIOS IO (read and write input, logging files, check points...) are handled in their own separate ensemble-member spaces.

Other than additions to `um_shell` to ensure that each memeber executable runs in a specific directory (a requirement imposed by the Rose infrastructure) and minor additions to accommodate setting up the ensemble axis, the code described in Figure 7 suffices for the ensemble case.

### 3.4. *Rose ensemble configuration*

As set up through Rose, a UM simulation is defined as a collection of configuration files (in INI format). On job submission, (i) Rose creates namelists from the configuration files; (ii) creates a work space on the HPC; (iii) copies the namelists (and other data) into the work space; (iv) sets up and submits a PBS script to run the parallel job. We have developed a Rose ensemble member set up task to perform items (i)-(iii), which, in conjunction with the use of Rose optional override configuration files to specify parameters for each ensemble member, creates the appropriate environment for each member to run in on the HPC. An overide file typically contains one line to specify an initial condition or a parameter value.

A separate task to run the ensemble then creates the appropriate MPMD aprun command for submission of multipe UM instances and XIOS.

### 3.5. *Configuring ensemble output*

XIOS currently supports max, min, and mean reductions. Our interest is primarily in the ensemble mean. It is a general feature of XIOS that data transformations are performed on the the basis of the target grid for the output field. In the example

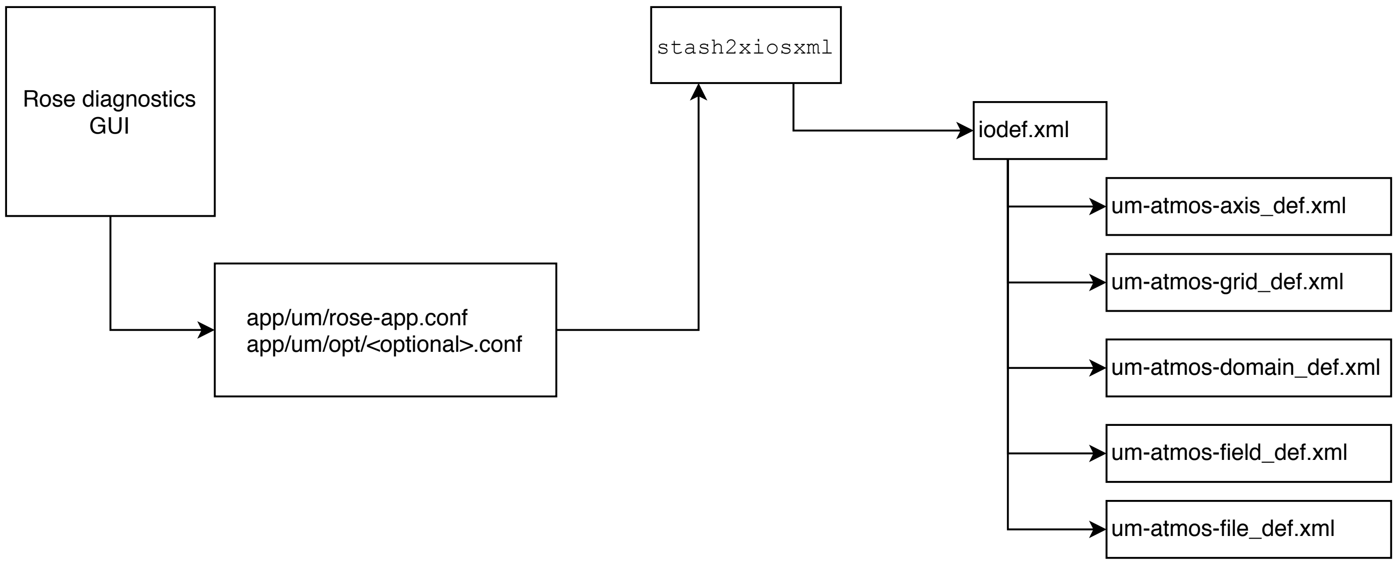


Figure 5: Work flow for creating XML files from STASH requests. The familiar Rose GUI isolates the user from details about XIOS XML. `stash2xiosxml` reads configuration files generated by Rose to create an XML description of the diagnostic output.

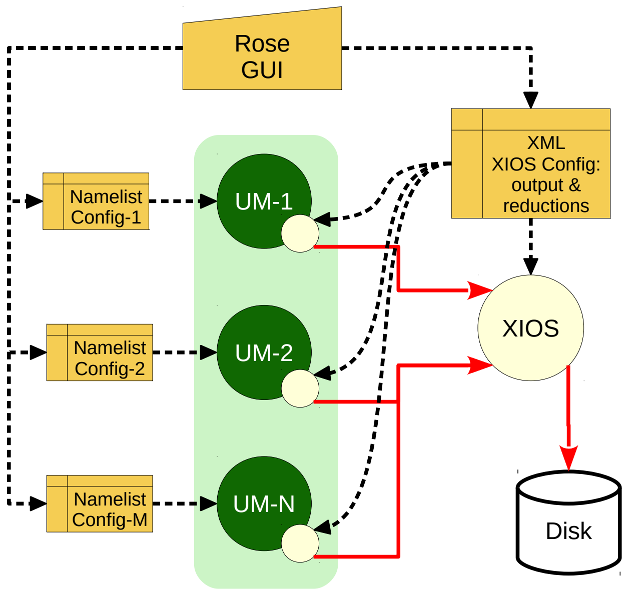


Figure 6: Key control and data flow concepts for XIOS control of a UM ensemble. The XIOS controls the output of each UM instance by exploiting XIOS clients within each process of each UM instance. Output from the clients goes through XIOS server instances to disk.

in section 3.2, the target grid spanned all ensemble members; a reduction over the ensemble axis is achieved by appropriately defining the grid on which to output the field. In the following, the ensemble-reduction grid `um-atmos_grid-reduce` includes instruction to reduce over the ensemble axis with operation `average`.

```
<grid id="um-atmos_grid-reduce">
<domain domain_ref="um-atmos_domain" />
<axis axis_ref="um-atmos_vertical" />
<scalar id="3d-ensmean">
<reduce_axis operation="average" />
</scalar>
</grid>
```

Our Python utility `stash2xiosxml` (section 3.1.1) includes support for creating XML for ensemble reductions.

### *3.6. New code in the UM*

Approximately 3000 additional lines of code were added to the UM, and isolated in files located in a subdirectory of the regular UM source code (`/um/src/control/xios`) prefixed with `um_xios`. Figure 7 indicates where the new code impacts the UM. The majority of work is done in the model initialisation stages with only minimal changes required to intercept and redirect diagnostics to XIOS. The global communicator is split by a native XIOS call in `um_xios_init_mpi` which returns the communicator for use by the model. The additional XIOS functionality is entirely independent from all other UM IO. Checkpoint files, logging files, and any diagnostics selected not to use XIOS, are handled through traditional UM means.

### *3.7. Build system and job submission*

Minor changes were made to the Rose GUI and UM build configuration files to accommodate the use of the XIOS library.

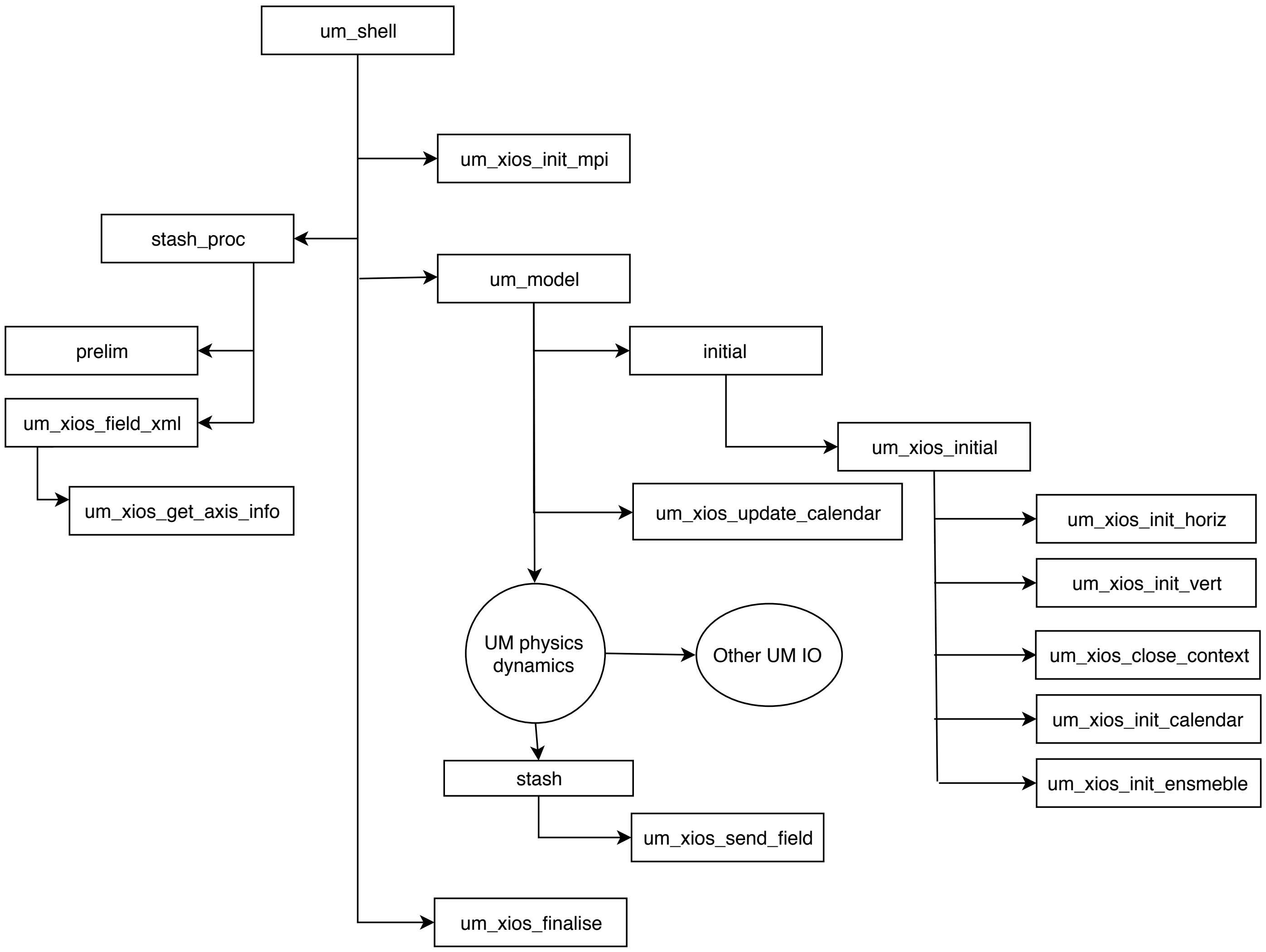


Figure 7: Areas in the UM where use of XIOS impacts the code. Diagnostics are intercepted in the routine stash. All other UM IO (logging, checkpointing, and diagnostics not directed to XIOS) is handled in the traditional manner.

Minor changes were made to the Rose GUI and the scripts which generate the job submission file to enable the launch of the UM and XIOS in MPMD mode.

## 4. Results

We inserted our modifications into version of the GA7.0 UM 10.7 AMIP [17] model which we run at both low resolution (N96) and high resolution (N512). For the purposes of evaluating the use of XIOS, we carried out two types of tests:

1. Single Model Simulations: where we had the twin goals of understanding the cost and behaviour of XIOS in the context of the UM, and establishing a baseline for,
2. Ensembles of Model Simulations — the main goal of our work.

To support these two experiments we configured the model to run for 24 model hours and 9 model hours respectively, and in both cases, writing out the data hourly. Neither experiment reflects the normal balance of compute to output, but they allow us to evalute the XIOS performance without long and expensive integrations. We selected 50 fields as diagnostics from several UM physics and dynamics sections, to include 2-d and 3-d fields, output as instantaneous or time accumulations, with the aim to mimic typical near-future climate integeration data volumes.

### *4.1. Single Model Simulation*

A key requirement for the output capability of an IO server is that it can support the requsite data processing and writing to disc asynchronously and without stalling or throttling the main computation (keeping hundreds to tens of thousands of processors idle while data is written to disc is an expensive prospect). Our main single UM test focussed on confirming that XIOS is capable of handling large amounts of data without impacting model computation times.

We ran the single model simulation at both N96 and N512 using 96 and 5184 cores respectively, but we concentrate here on results from the N512 tests as they were more onerous and the core-counts more typical of production. The test configuration created 214 GB of data at a constant hourly rate, which translates to 77TB/model-year. While this data rate far in excess of that for a typical climate model run, which is of order 2TB/model-year, our test does not capture the variations in compute to output intensity of a climate run, which would require a significantly longer integration to sample. Our ensemble runs (see section 4.2.2) do generate data at higher intensity but reveal as yet unresolved IO performace concerns.

XIOS provides many opportunities for performance optimisations which we have not explored, nevertheless, the results in Table 2 with 6 XIOS servers in multipe-file mode, clearly indicate that XIOS is effectively hiding IO from computation. The 7% XIOS overhead is an acceptable penalty for the rate of data production seen here - run time jitter is frequently at this level as seen in the no-IO result. Single file output can incur a larger IO overhead and requires more tuning for optimal preformance, which is out of scope here, and perhaps of less interest in the longer term with the increased interest in data sharding and the capabilities to manage distributed data inherent in the latest data analyis software [18].

### 4.2. *UM ensembles*

We applied our developments to ensembles of N96 and N512 models. The complete set of experiments we have carried out is summarised in table 3. Results from our initial experiments are presented in Figure 8. Data points in blue are for XIOS multiple-file output, and in red for single-file output (note, as for the single model case, we have made no attempt to optimise the XIOS configuration). The ensembles are configured to write diagnostics for all members and the ensemble avarages for all fields. This configuration is **not** how we anticipate production runs to work but serves as a test case.

#### 4.2.1. *N96*

The multiple-file ensemble performance is encouraging with relatively flat peformance over the range of ensemble sizes - perfect performance is represented by horizontal lines. The 100-member ensemble generates 45TB/mode year.

#### 4.2.2. *N512*

We initially ran the N512 model at two processor decompositions - in Figure 8, N512 runs labelled with a (1) used on 960 PEs/ensemble member, those labelled (2) ran with 5184 PEs/ensemble member. It is worth noting that these and all of our ensembles ran *out of the box* - to our knowledge XIOS has not been tested under such conditions, where the 10-member ensemble requires half of ARCHER to run on to generate 467TB/model year. Scaling for the N512 ensemble is less encouraging at first glance. It is clear that single-file output, based on MPI-IO has a significant impact on performance, and while multiple-file output performs better, its scaling is not ideal. The small model is clearly scaling better than the large model, and much of this must be to do with the increased processor counts in the large model and the associated communications, which are clearly exacerbated when the data is also being reduced to

| Resolution | Ensemble Size | |
|---|---|---|
| | 100 | 10 |
| N96 | 9600 | |
| N512 | | 54000 |

Table 1: Maximum core counts for the largest ensembles

| | no IO | XIOS | UM IO (single PE) |
|---|---|---|---|
| run time | 400 | 400 | 830 |
| IO overhead | | 30 | 428 |

Table 2: Single model N512 IO performance. The integration ran for 24 model hours to create 214GB of diagnostic output (equivalent to 77TB/year.) All times in seconds with model initialisation time removed. Note, run time jitter is quite large.

one or a few PEs for writing. None of these results are particularly surprising: XIOS was not designed for ensembles, and the communications patterns have not been optimised accordingly. Future work aims to address this.

These initial tests conflate two issues; (i) internal XIOS scaling and; (ii) speed to write to disc. Figure 9 shows some preliminary results aimed at better understandng how internal XIOS operations scale with ensemble size. These runs write out only the ensemble averages, are configured for multiple-file output, and data is written to /tmp to eliminate the actual time to write to disc. The figure shows data for multiple runs (with significant jitter) and lines which represent best fits to the scattered data points. There is no clear candidate responsible for the gradual relative slowing - but nor is the lack of scaling as critical as might be assumed from results in Figure 8.

## 5. Summary and Further Work

We have achieved three specific ambitions: (1) We have adapted a current branch of the Met Office Unified Model to replace much of the diagnostic system with the XIOS. (2) We have exploited a single executable MPI environment to run multiple UM instances with output sent to XIOS, and (3) We have demonstrated that simple ensemble statistics can be calculated in-flight, including both summary statistics of individual ensemble members, and cross-member statistics such as means and extremes.

We plan to properly profile the ensemble runs to fully understand the scaling behaviour - our previous attempts were frustrated by technical difficulties with our utility of choice.

In our current setup, failure of a single ensemble member will crash the entire ensemble; we have inherited UM checkpointing of course, so restarting the ensemble mid-stream is possible, but ideally we should be able to manage such a failure more gracefully. We can imagine a scenario whereby we would trap ensemble member errors and subsequently send *missing data* or otherwise signal the failure to XIOS; a modified XIOS would then handle the situation in a manner to be determined.

There is also scope to greatly improve the range of ensemble processing capability within XIOS and indeed in the entire en-

| Name | Resolution | Decomposition | Output | Ensemble-Size/Core-Count | | |
|---|---|---|---|---|---|---|
| N96:single(1) | N96 | 12x8=96 | Single | 100/9600 | 20/4800 | 1/96 |
| N96:multi(1) | N96 | 12x8=96 | Multi | 100/9600 | 20/4800 | 1/96 |
| N512:single(1) | N512 | 24x40=960 | Single | 10/9600 | 5/4800 | 1/960 |
| N512:multi(1) | N512 | 24x40=960 | Multi | 10/9600 | 5/4800 | 1/960 |
| N512:single(2) | N512 | 72x72=5184 | Single | 10/51840 | 5/25920 | 1/5184 |
| N512:multi(2) | N512 | 72x72=5184 | Multi | 10/51840 | 5/25920 | 1/5184 |

Table 3: The XIOS experiments carried out. Two different resolutions (N96, N512) with two different output file configurations (single: output from all model instances and processors writing to one file, and multi: output written to multiple files) and a range of ensemble sizes and processor decompositions (and total core counts).

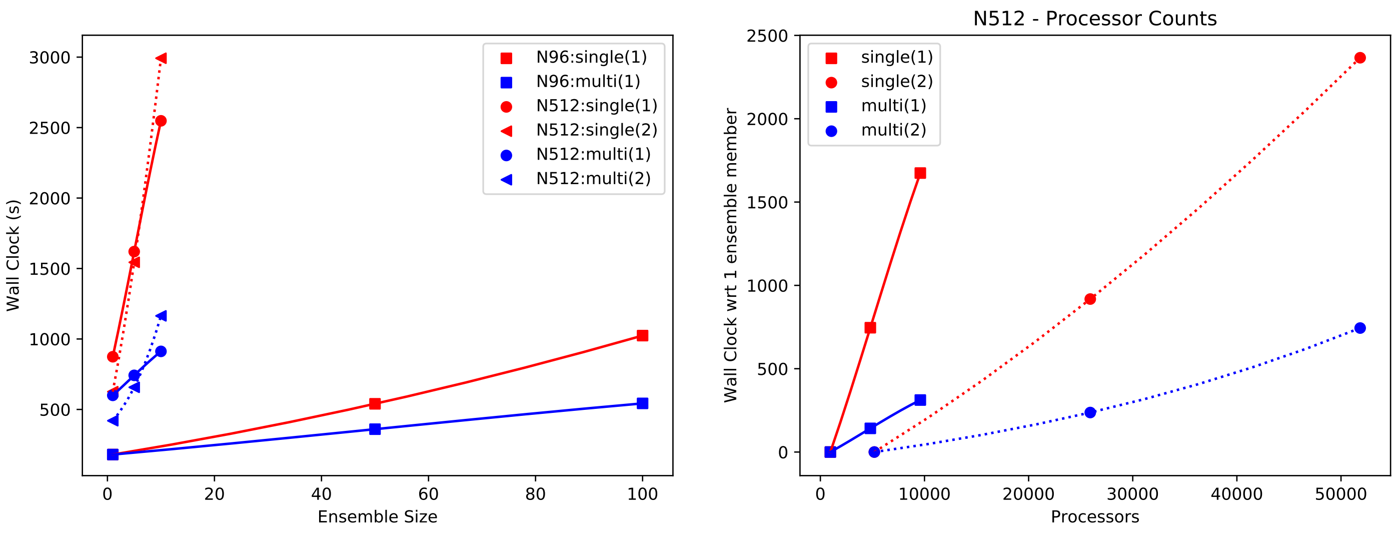


Figure 8: Wall clock time against sensemble size (left panel) and, for the N512 runs, processor count (right panel). Data is shown from the different ensemble configurations and sizes described in table 3. If XIOS were completely hiding I/O (and ensemble reductions) the lines in the left panel would be horizontal.

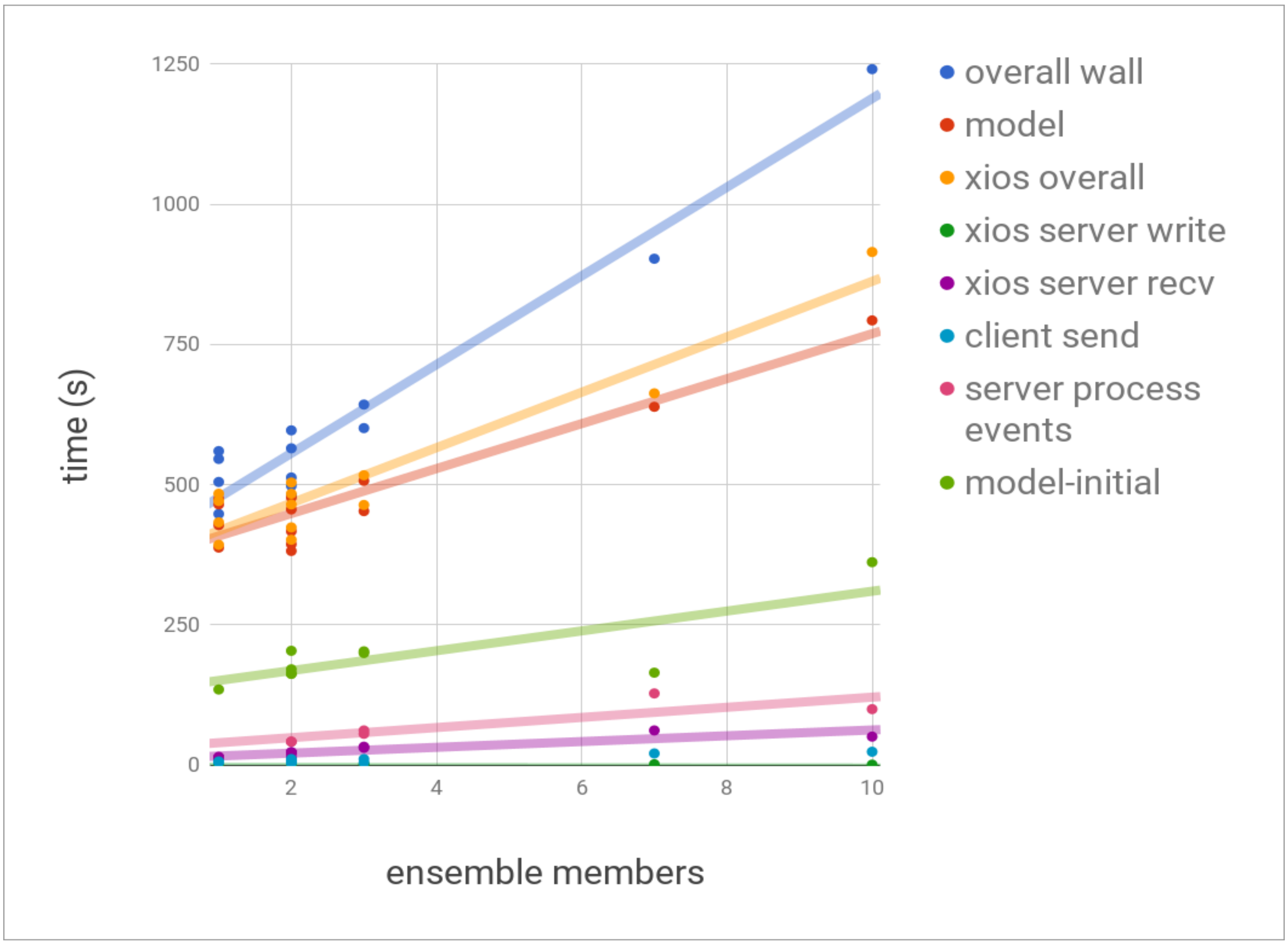


Figure 9: N512 ensemble with a multiple-file XIOS configuration with output to /tmp. Timings for several internal XIOS operations are reported, along with timings for a represetative ensemble member at each ensembe size (showing flat performace as expected). Multiple runs are reported for selected ensemble sizes showing moderate jitter.

semble management activity. As a result of the success of this project, we have secured further funding through ESiWACE2 to address some of these issues.

We note also that the Met Office have expressed an interest in assisting us to lodge our UM branch into a UM release in light of our successes and in view of their anticipated use of XIOS in LFric [1].

[1]LFric will be the successor to the UM. It uses a cubed-sphere mesh with mixed finite element methods and a domain specific language. Work is ongoing to support finite element data types through XIOS.

## Acknowledgements

This work uses the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk).This work is funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk).

We acknowledge support from the National Centre for Atmospheric Science (NCAS.)

## References

[1] E. Hawkins, R. Sutton, The Potential to Narrow Uncertainty in Regional Climate Predictions, Bulletin of the American Meteorological Society 90 (8) (2009) 1095–1107. doi:10.1175/2009BAMS2607.1.

[2] Y. Meurdesoif, H. Ozdoba, A. Caubel, O. Marti, The XML I/O Server (2012).

[3] R. Buizza, P. L. Houtekamer, G. Pellerin, Z. Toth, Y. Zhu, M. Wei, A Comparison of the ECMWF, MSC, and NCEP Global Ensemble Prediction Systems, Monthly Weather Review 133 (5) (2005) 1076–1097. doi:10.1175/MWR2905.1.

[4] G. A. Meehl, R. Moss, K. E. Taylor, V. Eyring, R. J. Stouffer, S. Bony, B. Stevens, Climate Model Intercomparisons: Preparing for the Next Phase, Eos, Transactions American Geophysical Union 95 (9) (2014) 77–78.

[5] W. S. Parker, Predicting weather and climate: Uncertainty, ensembles and probability, Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics 41 (3) (2010) 263–272. doi:10.1016/j.shpsb.2010.07.006.

[6] M. Leutbecher, Ensemble size: How suboptimal is less than infinity?, Quarterly Journal of the Royal Meteorological Societydoi:10.1002/qj.3387.

[7] M. S. Mizielinski, M. J. Roberts, P. L. Vidale, R. Schiemann, M.-E. Demory, J. Strachan, T. Edwards, A. Stephens, B. N. Lawrence, M. Pritchard, P. Chiu, A. Iwi, J. Churchill, C. del Cano Novales, J. Kettleborough, W. Roseblade, P. Selwood, M. Foster, M. Glover, A. Malcolm, High-resolution global climate modelling: The UPSCALE project, a large-simulation campaign, Geosci. Model Dev. 7 (4) (2014) 1629–1640. doi:10.5194/gmd-7-1629-2014.

[8] S. G. Yeager, G. Danabasoglu, N. Rosenbloom, W. Strand, S. Bates, G. Meehl, A. Karspeck, K. Lindsay, M. C. Long, H. Teng, N. S. Lovenduski, Predicting near-term changes in the Earth System: A large ensemble of initialized decadal prediction simulations using the Community Earth System Model, Bulletin of the American Meteorological Societydoi:10.1175/BAMS-D-17-0098.1.

[9] B. N. Lawrence, M. Rezny, R. Budich, P. Bauer, J. Behrens, M. Carter, W. Deconinck, R. Ford, C. Maynard, S. Mullerworth, C. Osuna, A. Porter, K. Serradell, S. Valcke, N. Wedi, S. Wilson, Crossing the chasm: How to develop weather and climate models for next generation computers?, Geoscientific Model Development 11 (5) (2018) 1799–1821. doi:10.5194/gmd-11-1799-2018.

[10] H. T. Hewitt, D. Copsey, I. D. Culverwell, C. M. Harris, R. S. R. Hill, A. B. Keen, A. J. McLaren, E. C. Hunke, Design and implementation of the infrastructure of HadGEM3: The next-generation Met Office climate modelling system, Geoscientific Model Development 4 (2) (2011) 223–253. doi:10.5194/gmd-4-223-2011.

[11] S. Valcke, The OASIS3 coupler: A European climate modelling community software, Geoscientific Model Development 6 (2) (2013) 373–388. doi:10.5194/gmd-6-373-2013.

[12] G. Madec, NEMO Ocean Engine, Note du Pôle de modélisation, Institut Pierre-Simon Laplace (IPSL), France, No 27, ISSN No 1288-1619, 2008.

[13] L. Bessières, S. Leroux, J.-M. Brankart, J.-M. Molines, M.-P. Moine, P.-A. Bouttier, T. Penduff, L. Terray, B. Barnier, G. Sérazin, Development of a probabilistic ocean modelling system based on NEMO 3.5: Application at eddying resolution, Geoscientific Model Development Discussions (2016) 1–24doi:10.5194/gmd-2016-174.

[14] Met Office Modelling Infrastructure Support Systems, ROSE (Jul. 2018).

[15] Hilary James Oliver, Matt Shin, Ben Fitzpatrick, Andrew Clark, Oliver Sanders, Declan Valters, Sadie Bartholomew, Kerry Smout-Day, challurip, David Matthews, Scott Wales, Tomek Trzeciak, Bruno P. Kinoshita, Jonny Williams, Rosalyn Hatcher, Annette Osprey, Alex Reinecke, ivorblockley, Martin Dix, lhuggett, Kevin Pulo, Andrew Huang, Cylc: The Cylc Meta Scheduler (Jul. 2018). doi:10.5281/zenodo.59457.

[16] Met Office Modelling Infrastructure Support Systems, FCM: A modern Fortran build system + wrappers to Subversion for scientific software development (Jun. 2018).

[17] D. Walters, A. Baran, I. Boutle, M. Brooks, P. Earnshaw, J. Edwards, K. Furtado, P. Hill, A. Lock, J. Manners, C. Morcrette, J. Mulcahy, C. Sanchez, C. Smith, R. Stratton, W. Tennant, L. Tomassini, K. Van Weverberg, S. Vosper, M. Willett, J. Browse, A. Bushell, M. Dalvi, R. Essery, N. Gedney, S. Hardiman, B. Johnson, C. Johnson, A. Jones, G. Mann, S. Milton, H. Rumbold, A. Sellar, M. Ujiie, M. Whitall, K. Williams, M. Zerroukat, The Met Office Unified Model Global Atmosphere 7.0/7.1 and JULES Global Land 7.0 configurations, Geoscientific Model Development Discussions (2017) 1–78doi:10.5194/gmd-2017-291.

[18] D. Hassell, J. Gregory, J. Blower, B. N. Lawrence, K. E. Taylor, A data model of the Climate and Forecast metadata conventions (CF-1.6) with a software implementation (cf-python v2.1), Geoscientific Model Development 10 (12) (2017) 4619–4646. doi:10.5194/gmd-10-4619-2017.