Attachment or Appendix D

Definition: Climate Modeling and Research System Benchmarks

Solicitation Number Reserved

Table of Contents

1. Benchmarks **3**
1.1. Benchmark Overview *3*
1.2. Code Changes and Audit Trails *3*
1.3. Performance Data *4*
1.4. Materials to be Returned with Solicitation *5*
1.5. Scaling Benchmark *6*
1.5.1. General Comments 6
1.5.2. Running the Scaling Study 6
1.6. Throughput Benchmark *7*
1.6.1. General Comments 8
1.6.2. Throughput Benchmark Scoring 8
1.6.3. Throughput Benchmark Output 9
1.7. I/O Benchmarks *9*
1.7.1. Metadata Operations 9
1.7.2. Fast Scratch Benchmark 10
1.7.3. Long Term Fast Scratch Benchmark (LTFSB) 11
Figure 3 illustrates the filesize distribution for the CM2-HR application. **12**
2. CMRS Acceptance Test **14**
2.1. Hardware *14*
2.2. Entry Criteria for Functionality Test *14*
2.3. Functionality Test *14*
2.4. Performance Test *14*
2.5. Stability Test *14*
2.6. Throughput Test *15*
3. Benchmark Overview **16**
4. Scaling Study **16**
4.1. CM-CHEM and CM2-HR *16*
4.1.1. Model Verification 17
4.1.2. Model Reproducibility 17
4.2. GFS *18*
4.2.1. Model Reproducibilty and Validation 18
4.2.2. Parallelization of GFS 19
4.3. FIM *19*
4.3.1. Building FIM 19

DRAFT

4.3.2. Running FIM 20
4.3.3. Verfication Procedure 21

DRAFT

1. Benchmarks

1.1. Benchmark Overview

The purpose of the benchmark suite is to establish the performance of the proposed CMRS computer sytem(s) in executing simulation codes representative of the current and anticipated computational research conducted by NOAA. The system must be able to successfully run the benchmark suite at the desired scales and demonstrate accuracy to within the limits specified for each separate benchmark code. In order to be considered for award, Offerors must successfully complete the benchmarks described below.

The benchmark is composed of 3 parts with the following goals:

- Application scaling benchmarks for a range of weather and climate applications used by NOAA will be used to evaluate system performance and resource requirements.
- A workflow throughput benchmark, based on the GFDL coupled climate model, will be used to evaluate both the capacity of the proposed system and the performance of the filesystem under load. The benchmark will determine the maximum number of jobs that can be completed in a fixed wall clock time using the full system.
- An I/O benchmark will be used to evaluate filesystem aggregate I/O and metadata transaction rates.

The benchmarks may be obtained by following the instructions in Section 3. The file "Benchmark_Results.xls" is distributed with the benchmark codes. Timings for all jobs should be entered into this spreadsheet.

All material (code, data, scripts, etc.) distributed as part of this benchmark suite are the property of NOAA. The benchmark codes and related confidential information may only be reproduced or copied by the Offeror for their normal use and analysis in conjunction with the benchmark testing. Any changes made by the Offeror to the benchmark codes will become the property of NOAA.

1.2. Code Changes and Audit Trails

The Offeror may make changes to benchmark source code under the following conditions:

1. Changes to scripts are permitted only to account for architectural and queueing system differences. Such modifications must be documented.
2. Changes to codes must be consistent with relevant standards (ie. ANSI language standards, MPI API standards, etc.). No assembly level recoding of source code is permitted.
3. So that ORNL may track changes, all source code modifications must be isolated for conditional compilation using pre-processor #if/#endif definitions:

```
#if (defined CMRS_"OFFEROR" )
```

DRAFT

```
#endif
```

The Offeror should substitute their company name or initials for the "OFFEROR" keyword.

4. The Offeror must document all changes, describing briefly why the change was made and how it impacted performance.
5. The Offeror's benchmark source code and scripts must be returned with the response to this solicitation.

ORNL is interested in coding styles, application optimization and optimization techniques. However, given the constantly evolving research nature of the applications, changes to the benchmark code may or may not produce improvements in the general set of applications for which the benchmarks are surrogate. Therefore, code changes except those required to make the model run correctly are only allowed if the Offeror provides a second, separate set of results, labeled "optimized".

All changes to the benchmark code must be identified, documented and acceptable to ORNL. Changes that are likely to be acceptable for baseline results include:

- Use of commercially supported libraries that are bid as part of the offering. The level of effort required to introduce the library into the general source code base will be evaluated.
- Compiler command lines with performance-specific options including, but not limited to, automatic parallelization.
- Use of commercially available and supported source pre-processors that are bid as part of the offering.
- Use of compiler "directives" within the source.

1.3. Performance Data

Gathering of performance data is targeted for a system equivalent to that offered for the initial delivery. The Test Systems on which the benchmarks are run and for which performance data is reported shall be as close as possible to the initial offered system. ORNL acknowledges that it may not be possible to use the offered system for the solicitation proposal. Therefore ORNL will evaluate performance projections based on the characteristics of the test system (i.e. actual test system size, technology equivalence, etc), thoroughness of data gathering, and projection methodology.

If technologies such as SMT are available on the benchmarking system, ORNL would like to see results with and without this feature. Please insert sheets or sections into the benchmark results spreadsheet and clearly indicate how many tasks, threads, physical cores and nodes were used for each run.

DRAFT

Baseline results must use all cores on each socket and all sockets on each node. Additional results may be provided if improved performance and capacity are achieved in other configurations.

1.4. Materials to be Returned with Solicitation

The Offeror must provide, in tar/gzip format, the source code and scripts used and the requested verification output for all aspects of the benchmark, as described in the benchmark instructions and README files, on ISO-9660 CDROM. All written responses and spreadsheets called for in these sections must be returned with the solicitation Proposal in printed form and digitally on ISO-9660 CDROM.

The Offeror's proposal shall provide a detailed but concise description of the benchmark system and system configuration. The Offeror should also provide information on the benchmark system in the Offeror_info worksheet of the Benchmark_Results spreadsheet. This should include but not be limited to:

- The number of physical and logical processors on the system.
- Processor characteristics, including
  - cycle time
  - socket configuration (number of cores per socket, availability of multi-threading, memory per core)
  - node configuration (number of sockets per node)
  - peak performance,
  - vector length
  - cache configuration
  - total and application memory available to each core, socket, and compute node
  - memory type
- A description of the communication fabric of the system
- The hardware and software supporting the file system(s) used for the benchmark
- OS version, user configurable kernel and system parameters

The Offerer shall provide a complete, concise description of the data-gathering procedures, the data gathered and any extrapolation methodology used. All timings shall be presented in whole units of seconds. Fractional timings that are less than 0.5 shall be rounded "down" to the nearest integer; timings that are greater than or equal to 0.5 shall be rounded "up" to the nearest integer. The Offeror's proposal must report what compiler, compiler version, compiler options, libraries, etc. were used for each benchmark and the purpose of each option should be reported.

Offerors are not allowed to change the floating-point precision of any of the benchmarks.

With respect to the data describing the Test System, the Offerer shall describe how the proposed system will differ from the Test System used for the proposal preparation. Further, the Offerer

DRAFT

shall describe how the data provided and the extrapolations from the Test System show that the installed system will perform as offered.

1.5. Scaling Benchmark

1.5.1. General Comments

The scaling benchmark comprises four applications:

- CM2-CHEM
- CM2-HR
- GFS
- FIM

The goal of the Scaling Study is to measure individual application performance, scaling and resource requirements. Ideally, data for the Scaling Study should be collected using the same Test System used for the Throughput Benchmark. Lacking this consistency, detailed documentation of the system differences shall accompany Scaling Study data. Commentary concerning the scaling and performance implications of the system differences shall be provided as well.

Descriptions of the individual benchmark experiments are provided with each of the benchmark codes. See Section 3, Benchmark Instructions and the README files included with the benchmark source for details.

1.5.2. Running the Scaling Study

Applications shall be run on as few processor cores as practical for the given experiment and shall be scaled to as many cores as possible. It is clear that at some number of cores the performance improvement of an application with respect to a particular experiment may flatten and perhaps decline (the “rollover” point of the scaling curve). For all experiments, the Offerer shall provide data, documentation and projections as necessary up to and including the rollover point.

In order to obtain a reasonable understanding of the scaling curve, the Offeror shall provide a minimum of 4 data points for each experiment. Note that the choice of core counts shall show the range of performance. It is of no use to have "scaling data" with data points taken at roughly similar core counts.

Results of the scaling study must be entered into the appropriate sections of the benchmark_results.xls spreadsheet.

The Offeror shall include one of the scaling study points for CM2-HR at the core count and decomposition that is proposed for the throughput.

DRAFT

1.6. Throughput Benchmark

The goal of the throughput benchmark is to measure system performance under workload and runtime environment. The proposed system shall maximize the overall throughput and minimize the execution time as measured by the number of CM2-HR workflow instances that can be completed in the allotted time and the reduction in wall clock time for each instance as compared with the baseline measurement provided by ORNL.

The workflow is a sequence of steps designed to represent the complete end-to-end execution of a single modeling application. The procured system will be responsible for only a subset of the full end-to-end workflow processing. In particular, it is anticipated that the procured system will

- Set up the workflow run directory from data copied to ORNL for the purposes of the specific run and from data located at ORNL
- Run the model using the requested resources
- Move/copy the run output to
    - The long term scratch filesystem
    - The fast scratch directory used for the next segment of the model run

For throughput benchmarking, a “workflow instance” is comprised of 2 consecutive CM2-HR simulation segments. The wall clock time of these steps as measured from a single script will be used to evaluate the proposed system performance.

For the purpose of timing the benchmark, it is assumed the initial run directories have been set up and the model input data are already in place. The run script provided will initiate the first segment and run it to completion. It will then move files, set up the run directory for the second segment, run that segment and move the resulting files. The Offeror shall report the timings produced by each segment as well as the time for the complete job script including all data movement. If benchmark runs are done using only one filesystem, data shall be copied and then deleted to ensure that blocks are actually moved.

The Offeror may modify the script to use whatever sequence of operations the Offeror defines as optimal for use in a production setting such that the run directory for the second segment is setup in such a way that running the segment will not overwrite the history files from the previous segment. The next segment may begin as soon as the model output from the previous segment is in a state where it will not be overwritten by the new segment. During the acceptance test, the run script must initiate the movement of the history files from the FS to the LTSFS in a manner the Offeror recommends for the production setting.

A "completed work flow" shall mean that the next segment (segment 3) of the model run is ready to run. This implies that the new segment run would not overwrite history files from the previous segment. It also implies that the restart files from the previous segment (which were written to the RESTART/ subdirectory) are now readable from the INPUT/ directory. This can be accomplished (for example) by an Offerer-designed set of move, copy and/or link operations. These operations need not be completed prior to start of the next segment. For example, the

DRAFT

Offerer may design a set of operations that starts the next model segment in a new directory having spawned a set of processes executing concurrently to clean up the old run directory.

For reference, current methodologies move the job segment history files and copies of the history files to the equivalent of the LTFS. The scripts also push a copy of the history files into the run INPUT/ subdirectory. All operations complete serially before the next segment initiates. Alternate approaches are encouraged. For example, a different approach might simply setup a new run directory and initiate asynchronous data movement processes for the history files from the previous segment. The new segment may then start as soon as its run directory is established.

Performance of the post-run data movement from the FS to the LTFS must be consistent with FS disk space availability requirements to maintain the full production workload.

1.6.1. General Comments

In the ideal case, throughput benchmark measurements are taken on the systems proposed for delivery using the same queuing and scheduling software being proposed for the installed system. It is understood that realization of the ideal case is unlikely. Thus, it is generally expected that Offerers will take performance measurements on systems with the software scheduling and queuing infrastructures currently available to them. After evaluating interactions between instances competing for resources, projection methodologies may be used to produce the proposed configuration.

At time of delivery, Offerers will be required to demonstrate the proposed system capacity and work flow instance wallclock performance.

1.6.2. Throughput Benchmark Scoring

The Throughput Benchmark must be completed in no more than 3.5 hours. Offerors shall measure and report the workflow component timings as described above. Based on this data and other aspects of the offered system(s), the Offeror shall propose the number of workflow instances that can be completed within the 3.5-hour benchmark time on the proposed system. This defines the system's Capacity. Improvements in wallclock time to completion relative to the baseline define the system's Capability. The run scripts used for all throughput measurements shall be returned with the benchmark output.

As part of the Acceptance Test for a system, the Offeror shall supply and launch run scripts compatible with the offered queuing system for all work flow instances. If asynchronous data movement techniques are employed to clean up from a previous run while overlapping with a new segment start, it must be demonstrated that the proposed workflow performance can be achieved even in the presence of data moving processes. It is essential that the Offeror account for potential interactions between the work flow instances in proposing the offered throughput time.

DRAFT

The proposed computational system shall demonstrate runtime variability consistent with the Solicitation Technical Specification for all benchmarks and job load during the acceptance time period. The Offeror shall explain any variability outside this limit to the satisfaction of ORNL.

ORNL is interested in the sustainable performance of the system under "normal load". Therefore all file system(s) supporting the benchmark runs shall be fragmented and filled to at least 60% capacity. Additionally, the benchmark shall be evaluated with the storage system running in degraded mode with at least 10% of the LUNS that contain the active file system(s) being rebuilt. Offerors should propose the methods by which file system fragmentation, 60% capacity and "degraded mode" shall be achieved for the Acceptance Test for both the FS and the LTFS.

The Offeror shall verify reproducibility of the model at the same decomposition and core count. The reproducibility of the atmospheric and ocean components of the model may be verified through a series of checksums and global integrals written to stdout at the end of the run. See the verification directory for CM2-Chem for details. Note that this can only check for platform "self consistency"; it is not expected that the Offeror will bitwise reproduce ORNL provided output.

1.6.3. Throughput Benchmark Output

The file "Benchmark_Results.xls" has been distributed with the benchmark codes. In this file, an Excel spreadsheet has been provided for the Throughput Benchmark. The CM2-HR model contains functions that report the Total runtime, Initialization, Main loop and Termination timing in terms of the minimum process time (tmin), the maxium process time (tmax) and the average process time. The Offeror shall report the maximum process time (tmax) for the Total Runtime, Initialization, Main loop and Termination for each segment of each work flow instance, as well as the job runtime (which includes data movement time) in Benchmark_Results.xls.

The model writes to stdout (standard out - i.e. the "screen"). This information shall be captured (such as by piping to a file) and returned for all model instances. Only ascii output shall be returned.

1.7. I/O Benchmarks

1.7.1. Metadata Operations

To verify the metadata performance of the file system, please download the mdtest benchmark from http://sourceforge.net/projects/mdtest. For instructions on building and running this benchmark, please refer to README file provided in the source code distribution obtained from this link. The mdtest benchmark must be run with the following parameters for 4, 16, 32 and 64 processes:

- 20 iterations (-i)
- 100 creat/stat/remove per process (-n)
- single target directory (-d) on FS or LTFS

All output of this test shall be reported.

DRAFT

1.7.2. Fast Scratch Benchmark

The fast scratch benchmark (FSB) is a synthetic workload meant to mimic the I/O workload of CM-CHEM. This benchmark utilizes the IOR benchmark and runs multiple concurrent invocations of the IOR benchmark each with different request sizes and is run on the CMRS compute nodes. While much of the workload is large block (1MB) reads and writes, a substantial percentage of the workload is small block I/O. Figure 1 illustrates the baseline request size distribution during a run of the CM-CHEM application.

IOR can be download from http://sourceforge.net/projects/ior-sio. For instructions on building this benchmark, please refer to the README file provided in the source code distribution obtained from this link. For instructions on how to run this benchmark, please refer to the README file found in the tar file provided by ORNL.

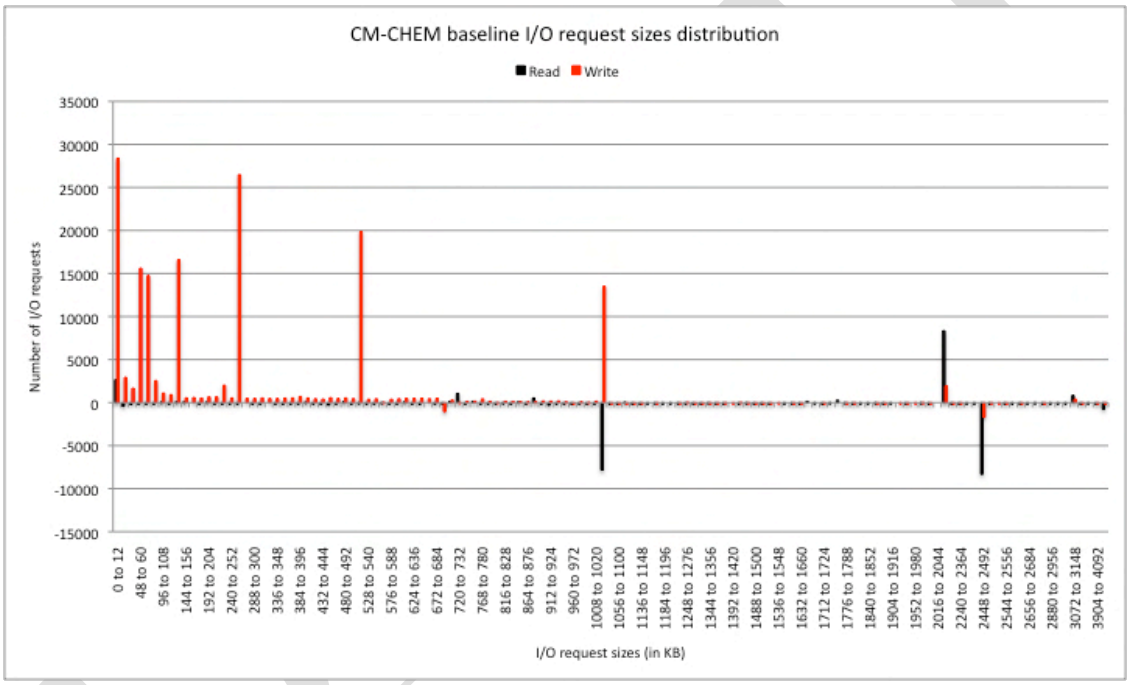

Figure 1. Baseline Request Size Distribution During CM-CHEM

Figure 2 illustrates the single restart request size distribution during a run of the CM-CHEM application during the checkpoint phase

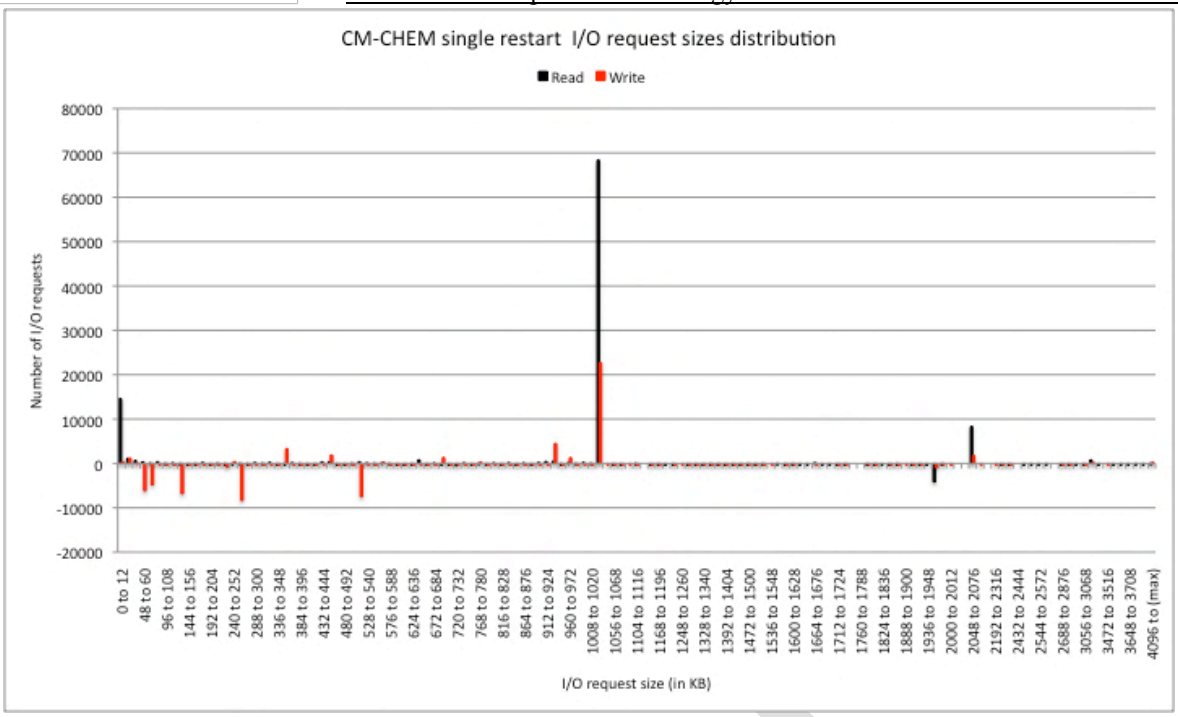

Figure 2. Single Restart Request Size Distribution During CM-CHEM

1.7.3. Long Term Fast Scratch Benchmark (LTFSB)

The LTFSB simulates the workload of moving multiple files between the FS and LTFS (a critical component of the CMRS workflow). This benchmark uses the IOR benchmark suite to simulate this workload and is run on the LDTNs. For instructions on how to run this benchmark, please refer to the README file found in the tar file provided by ORNL.

DRAFT

Figure 3 illustrates the filesize distribution for the CM2-HR application.

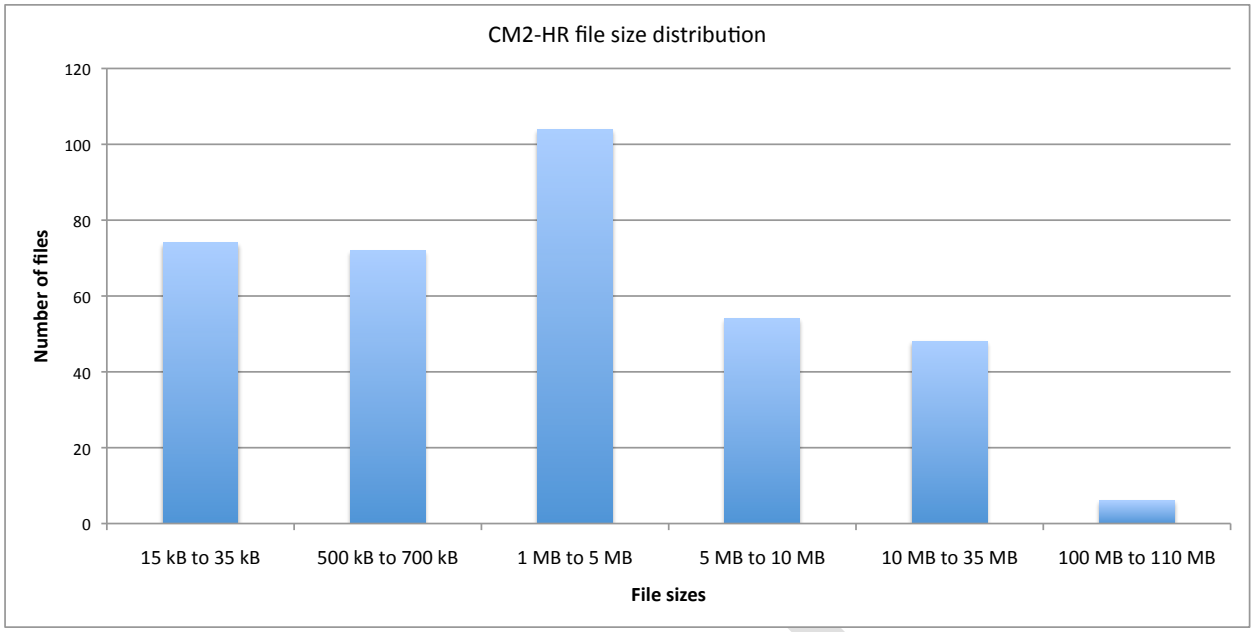

Figure 3. Filesize distribution for CM2-HR

DRAFT

Attachment or Appendix E

Definition: Climate Modeling and Research System Acceptance Test

Solicitation Number Reserved

Table of Contents

DRAFT

2. CMRS Acceptance Test

The CMRS acceptance test (AT) will comprise multiple components where the overall goal is to ensure that the system as a whole is capable of delivering the scientific output required by the NOAA project. The AT will be divided into 5 phases that will test the system infrastructure with a combination of applications and synthetic benchmarks. This section provides an overview of the acceptance test process, where final details will be negotiated as part of the contract award and prior to delivery of the system. The five phases are Hardware, Functionality, Performance, Stability, and Throughput.

The Acceptance Test must be successfully executed for each delivered subsystem, and for any substantive upgrade.

2.1. Hardware

The system will boot and run vendor diagnostics. All vendor diagnostics must pass and details of the diagnostic runs will be delineated to the AT team.

2.2. Entry Criteria for Functionality Test

Prior to entering the final four phases of the acceptance test the vendor will demonstrate the system capabilities by running the HPCC benchmark where the HPL component will achieve at least 70% of peak performance.

2.3. Functionality Test

The Functionality Test will contain but not be limited to system administration tests (reboots, infrastructure connectivity, compiler and library capabilities (C/C++, Fortran, MPI, OpenMP, Pthreads, optimized math libraries), the I/O subsystem and the generation of correct results from test cases from the benchmark applications outlined with this Solicitation.

2.4. Performance Test

The performance capabilities will be tested for all applications in the benchmark suite, the compiler functionality and implementations of the MPI and thread based programming models (OpenMP and Pthreads). Furthermore, synthetic tests will be applied to the I/O subsystem. All metrics to be applied to the system acceptance will be determined during contract award negotiation and prior to delivery.

2.5. Stability Test

The stability test will be a synthetic workload along with an optional (at the discretion of ORNL) workload from friendly users to the system. The synthetic workload will contain components

DRAFT

from both the functionality and performance tests. All application or systems errors during the stability test will be tracked and a root cause will be determined. A 95% correct completeness metric will be applied to all jobs run during the stability test. There will be a time limit for the stability test with an available uptime metric to be applied. The time limit will be at least 2 weeks. Downtimes for reboots do not count toward the stability time clock but do count toward the overall availability of the system.

2.6. Throughput Test

The throughput test will holistically determine the job throughput and data movement capabilities among the various file systems. See the Benchmarks section for an explanation of the throughput test.

DRAFT

Appendix X: Benchmark Instructions

3. Benchmark Overview

The CMRS benchmark suite contains a set of scaling benchmarks for a range of weather and climate codes, a workflow throughput benchmark and an I/O benchmark. This Appendix provides instructions for running the CMRS benchmark suite to establish the performance and capacity of the proposed CMRS computer system(s).

4. Scaling Study

4.1. CM-CHEM and CM2-HR

CM-CHEM is a coupled global model with a 2 degree, 48 level atmosphere with numerous chemical species and a 1 degree ocean. It represents near horizon climate research efforts. Given the resolution of the atmosphere and ocean models, the scalability of this application is limited to approximately 900 cores on the Cray XT systems located at ORNL.

CM2-HR is much higher resolution in both atmosphere and ocean. The model is constructed from 0.5 degree atmosphere and a 0.25-degree ocean. In a time frame exceeding the time line for the system procurement, the target for this model is a 0.25 degree atmosphere and a 0.1-degree ocean. This model is currently under construction and does not yet contain the "physics" of WF1. While it is believed that ultimately this job will scale to 8000 or more cores, the current early implementation (0.5 degree atmosphere + 0.25 degree ocean) appears to be limited to around 2100 cores on the Cray XT systems located at Oak Ridge National Lab. The scaling limitation is believed to be related to the boundary layer exchange model grid. Work is ongoing to alleviate this scaling bottleneck but will not be available for the benchmark.

CM-CHEM and CM2-HR are built on the NOAA / GFDL Flexible Modeling System (FMS). Both models use a cubed-sphere grid for the atmosphere and a tri-polar grid for the ocean. Both models break the total number of cores allocated to the application into 2 disjoint sets: a set running the atmosphere and land and a set running the ocean and sea ice. The sets "join" to perform the boundary layer calculations.

Both models are scalable in "discrete units" only.

Core counts for the atmosphere set are defined by 6*M*N where M and N are integers. As currently run in production, M is one of {N, N+1, N-1} (i.e. the sub-domain decomposition is either "square" or "slightly rectangular"). Other configurations with M and N as integers may also be possible.

Core counts for the ocean set can be more flexible. Experience has demonstrated that with most compilers, a completely malleable executable performs less well than an executable compiled such that almost all array sizes are known at compile time (the latter being known as a "static memory size" executable). Decompositions for "static memory" executables are limited to core

DRAFT

counts which produce the same number of sub-domain points on every process. An equivalent way to define this is that the number of cores used for each of the "X" and "Y" directions must divide the global grid evenly in each direction where X*Y = the total number of cores devoted to the ocean component.

Finally since all processes must synchronize to perform the boundary layer calculations as a single group, load balance considerations limit the number of atmosphere/ocean decompositions which can be used together effectively

The model contains functions that report the Total runtime, Initialization, Main loop and Termination timing in terms of the minimum process time (tmin), the maxium process time (tmax) and the average process time. The Offeror shall report the maximum process time (tmax) for the Total Runtime, Initialization, Main loop and Termination for each study instance in Benchmark_Results.xls.

The model writes to stdout (standard out - i.e. the "screen"). This information shall be captured (such as by piping to a file) and returned for all model instances. Only ASCII output shall be returned.

The Government requires that one of the scaling study points for CM2-HR shall be provided at the core count and decomposition proposed for the throughput.

4.1.1. Model Verification

The scripts CM-CHEM-verification and CM2-HR-verification are set up to run 2-day simulations and print information needed for verification. These should be run for both the reproducible executable and the higher optimization level by changing the value of “set executable” in the script.

4.1.2. Model Reproducibility

The output from CM-CHEM and CM2-HR (i.e. the history and restart files) is bitwise "reproducible" as defined below. The Government requires that there exist one or more modes in which each of these reproducibility characteristics can be maintained.

- Reproducibility across core count and problem decomposition. In this mode, the history and restart files resulting from a given model segment bitwise reproduce the results from the same code compiled with the same compiler version and settings and run on the same type of hardware (i.e. compute nodes and communication infrastructure) with the same runtime libraries regardless of run core count or decomposition. In addition to possible compiler settings necessary to produce this behavior, the model must be run with the xgrid_nml fortran namelist variable set to: make_exchange_reproduce=.true.
- Absolute reproducibility at the same core count. In this mode, the history and restart files resulting from a given model segment bitwise reproduce the results from the same code compiled with the same compiler version and settings and run on the same type of

DRAFT

hardware (i.e. compute nodes and communication infrastructure) with the same runtime libraries. This result shall hold regardless of compiler settings (i.e. executables compiled from identical source code bases with the same compiler version, compiler and linker settings to the same libraries shall produce bitwise identical restart and history files when run on the same model segment input files and parameter settings at the same core count and problem decomposition). Further, this result shall hold for all input file sets and parameters. This implies that an existing executable run in an identical environment on the same input at the same core count and problem decomposition shall bitwise reproduce history and restart files at all times.

To verify reproducibility across core count, the Offeror shall run the model for two (2) simulation days with make_exchange_reproduce=.true. for each problem decomposition along with any compiler flags and environment variables necessary to achieve reproducibility. See scripts CM2-HR-repro and CM-CHEM-repro. The reproducibility of the atmospheric and ocean components of the model may be verified through a series of checksums and global integrals written to stdout at the end of the run. See the verification directory for CM2-Chem for details. Note that this can only check for platform "self consistency"; it is not expected that the Offerer will bitwise reproduce Government provided output.

4.2. GFS

GFS is a global spectral weather model developed and used at NOAA NCEP. To build the GFS executable you will need to download and build ESMF version 2.2.2 release date 03/16/06 from http://www.esmf.ucar.edu/download/releases.shtml. Please see instructions provided in the README file of the GFS directory.

The initial condition files for this application are binary big endian. To generate little endian files, NOAA has provided a byteswap program. See fendian_conv.c provided with the application.

Two different resolution jobs are provided, T190 and T510. The T190 job is provided for porting purposes. The scaling benchmark should be run using the T510 files.

4.2.1. Model Reproducibilty and Validation

The GFS is bitwise reproducible with varying MPI task count and threads. Results should reproduce using the same code compiled with the same compiler version and settings and run on the same type of hardware with the same runtime libraries regardless of run core count or decomposition.

Scripts and reference output files are provided for validation of results. Results for RMS values for various output fields at each vertical level are produced. The results after 24 hours should match to 5 digits accuracy for surface pressure and temperature and the RMS difference of the

DRAFT

temperature fields should be less than 0.5. See the README file in the gfs/verify directory for detailed instructions.

4.2.2. Parallelization of GFS

GFS is a hybrid model utilizing both OpenMP and MPI for parallelization. The MPI decomposition is 1-dimensional, which limits the MPI scalability to about 2/3 of the wave truncation. T510 is thus limited to about 340 MPI tasks. The OpenMP scalability is linear to four threads and good to eight threads on the IBM Power systems at NOAA NCEP. As a result, the T510 job scales to about 2600 cores on the NCEP CCS IBM Power6 nodes. The model can be run with any MPI task count up to the scalability limit and will run as a single MPI task if sufficient memory is available. It cannot be run without MPI. Due to the I/O design, the memory used by each MPI task is about the same for all tasks except for the last task, which uses much more memory. The last MPI task gathers the entire model state and writes it to secondary storage asynchronously while model integration continues on the other tasks. As a result, the last MPI task may need to be run on a node with fewer other tasks.

ORNL would like to see results of a scaling study run in hybrid mode (OpenMP plus MPI).

The Offeror shall report the run time for each core count in the Benchmark_Results.xls spreadsheet.

4.3. FIM

The Flow-following finite-volume Icosahedral Model (FIM) from NOAA ESRL is a global global weather prediction model currently under development in the Global Systems Division of NOAA/ESRL. The FIM employees an Arbitrary Lagrangian-Eulerian (ALE) vertical coordinate running on a dynamic icosahedral horizontal grid. This ALE vertical framework is based upon a 'hybrid' structure, utilizing σ terrain-following levels near the surface and isentropic coordinates in the free atmosphere. The horizontal resolution of the icosahedral elements (which are primarily hexagons, with the exception of 12 pentagons) and hybrid levels are dynamically configurable at model run time.

Physical parameterizations in FIM match those used in the operational Global Forecast System (GFS) developed at the National Center for Environmental Prediction (NOAA/NCEP). This allows FIM forecast initialization from the GFS analysis. Hybridization in the vertical coordinate is achieved through a unique flux corrected transport scheme.

http://fim.noaa.gov

4.3.1. Building FIM

FIM utilizes the Scalable Modeling System (SMS) parallel programming package, an open source parallelization toolkit, also developed at NOAA/ESRL/GSD. SMS was developed to

DRAFT

simplify development and parallelization of atmospheric and oceanic models. This package provides a portable, directive-based parallelization library, simplifying the message passing required to support NWP models running on distributed (or shared) massively parallel computer systems. To enable development of the icosahedral grid utilized by FIM, SMS was enhanced to support an indirect addressing mode for non-Cartesian grids

To build the SMS components locate the sms directory and follow the directions detailed in the 'INSTALL' text file located at the top level of the sms directory. This procedure employs the familiar 'configure – make – make install' paradigm common to many open source packages. Note that installing this package in a non-standard location is supported by specifying a –dir option during the configure step (e.g. ./configure –dir my_sms_dir). Running the regression tests ensures your SMS installation is correct.

To build the FIM model and associated utilities change your current directory to the fim directory and follow the directions detailed in the 'README' text file at that location. To specify the correct location for the SMS library and utilities you'll need to either set the environment variable 'SMSM' to the correct location or modify this variable in 'Makefile' located in the FIMsrc directory.

4.3.2. Running FIM

Many of FIM's runtime variables are controlled through use of a namelist (FIMnamelist). This namelist provides a mechanism to control a wide variety of parameters used by FIM at runtime, including location of data files, number of compute processors, use of I/O processors, horizontal and vertical resolution, and many more. While you are welcome to experiment with these variables, with the exception of those mentioned specifically below none of these variables should be altered for reported benchmark results.

Once the FIM executables and utilities have been successfully built, change the working directory to the FIMrun directory as indicated in the README file. There you will find the FIMnamelist text file specifying the runtime parameters. The first two variables in this file specify the ComputeTasks and MaxQueueTime parameters used when submitting your job. At runtime an additional 32 processors will be used for I/O processing.

Find the variables 'DATADIR' and 'DATADR2' in the FIMnamelist file. These should be changed to reflect the location the 'fimdata' directory was loaded on your system. You may wish to test your FIM installation by running a lower resolution (done by changing the Grid level ('glvl') and/or smaller number of vertical levels ('nvl') parameters. These should be set to 9 and 64 respectively for reported results.

Please see the instructions provided with the benchmark code.

DRAFT

4.3.3. Verfication Procedure

As a check to ensure the model ran correctly and produced a reasonable answer please examine the reported precipitation and mass figures reported near the end of this file. While results will be identical across different numbers of processors on any machine we understand there will be small differences across various architectures and compilers. We expect the 'Global 3D mass' and 'Global 3D water vapor' results at time step 5760 (1 day) to be within 5 significant digits accuracy of the results in the sample (5.1232E+18, and 1.3697E+16 respectively).

Please provide runtimes as reported in the stdout file produced by the fim run for at least 5 runs in a range of processor counts. The table below is an example of runtimes from NOAA/ESRL's Jet MPP system. While these results only reflect the runtime for the model portion of the package, we will also require a grib file produced in the post directory to verify your results.

Compute Tasks	Total processors	Runtime (seconds)
560	592	8793.6
800	832	6094.4
1040	1072	4944.9
1280	1312	4180.5
1520	1552	3737.1
1760	1792	3047.3
2000	2032	2865.1

Figure 4. NOAA/ESRL FIM Results for Jet MPP

DRAFT