

**TASK GEOMETRY FOR COMMODITY LINUX CLUSTERS
AND GRIDS: A SOLUTION FOR TOPOLOGY-AWARE
LOAD BALANCING OF SYNCHRONOUSLY COUPLED,
ASYMMETRIC ATMOSPHERIC MODELS**

I. LUMB* AND B. MCMILLAN

*Platform Computing Inc.,
3760 14th Avenue,
Markham, Ontario L3R 3T7, Canada
E-mail: {ilumb, bmcmillan}@platform.com
* Formerly with Platform Computing Inc.*

M. PAGE AND G. CARR

*National Center for Atmospheric Research,
P.O. Box 3000,
Boulder, CO 80307-3000, USA
E-mail: {mpage, gcarr}@ucar.edu*

State-of-the-art atmospheric models comprise multiple components that interact synchronously. Because some components need to expend more computational effort than others, it is often the case that these components are computationally imbalanced. Although parallelism within a component can reduce the imbalance, there is still a need to coordinate component interaction via a coupler in real time. To address this issue, NCAR identified requirements for Task Geometry — a construct that specifies for the purpose of coordination the run-time topology within and between components. Although it has recently become generically available in Platform LSF HPC, the current focus is commodity architectures based on the Linux operating environment. Crafted originally for CCSM, Task Geometry appears applicable to other systems-coupled atmospheric models such as 4D-Var. Scale-coupled atmospheric models also show promise for application in areas such as subgrid-scale parameterizations for GCM models, and the class of interactions demanded by use cases emerging in the area of Homeland Security. In all cases it is possible to demonstrate via speedup and efficiency that Task Geometry enhances performance, however both metrics are problematical to quantify except in simple cases. With applicability from isolated Linux clusters to grids, Task Geometry remains useful in contexts that span organizational and/or geographic boundaries.

To appear in G. Mozdzynski (Editor), *Proceedings of the Eleventh ECMWF Workshop: Use of High Performance Computing in Meteorology*, World Scientific, 2005.

1. Introduction

Workload Management (WM) solutions aim to match application requirements (demand) with available resources (supply) subject to scientific and/or organizational policies (objectives). Generally speaking, WM solutions address this optimization problem with demonstrable effectiveness — even in today’s highly heterogeneous distributed computing environments in which multiple users/applications compete for the same resources. However, state-of-the-art atmospheric models, present a significant challenge.

A convenient, heuristic approach for understanding this challenge, from the WM perspective,^a is afforded by the granularity^b versus concurrency^c plot provided in Fig. 1.^{29,30} Using standard Cartesian conventions, the four quadrants of Fig. 1 have the following associations:

- Data Parallel (I, Upper Right) — Owing to parallelism that exists in the data, workloads in this quadrant have high degrees of granularity and concurrency. The resulting embarrassingly parallel workloads are well suited to compute farms and desktops that are architected to maximize compute capacity — i.e., workload throughput as a function of time.
- Serial (II, Upper Left) — Serial workloads plot in this quadrant. Owing to their sequential nature, applications in this quadrant have high degrees of granularity but no concurrency.
- Service (III, Lower Left) — Workloads from this quadrant tend to spend more time communicating than computing. As a result they are of low granularity and low concurrency.
- Compute Parallel (IV, Lower Right) — Workloads in this quadrant are the typical focal point for the HPC applications typical of the atmospheric sciences. In this case, the parallelism that exists at the source-code level is exploitable in shared-memory (e.g., via OpenMP) and distributed-memory (e.g., via MPI) contexts. These capability workloads seek to make use of HPC architectures to solve problems that are Grand Challenge in their scope.

^aAlthough instruction/process versus data heuristics exist and are widely used, the granularity versus concurrency approach has proven more appropriate in the WM context.

^bGranularity is a measure of the amount of computation that can take place before there is a need for synchronization or communication. Thus the ratio computation/communication serves as a proxy for the vertical axis of Fig. 1.

^cConcurrency refers to an ability to carry out activities simultaneously. In other words, it is a measure of the degree of parallelism that is present.

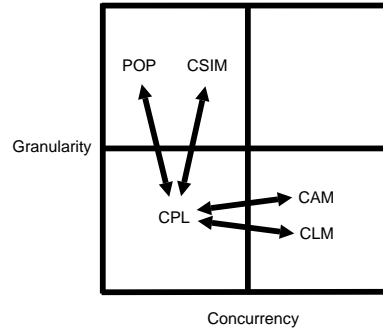


Figure 1. Granularity versus concurrency heuristic for the model components and coupler in CCSM. Double-edge arrows specify interaction between each of the components and the coupler via MPI. Note that the relative placement of a model component within a quadrant is not significant — e.g., CSIM does not necessarily exhibit greater concurrency than POP in quadrant II. Acronyms are expanded in the text.

To fix ideas for the purpose of illustration, i.e., to elucidate the challenge state-of-the-art atmospheric models present to WM solutions, the Community Climate System Model (CCSM)^{12,13,36} is considered in the context of Fig. 1. CCSM is a ‘meta-model’ in that it comprises four separate components^d — i.e., an atmospheric model (The Community Atmosphere Model, CAM),³⁵ a land surface model (The Community Land Model, CLM),³⁷ an oceanic model (The Parallel Ocean Program, POP),⁴² and a sea-ice model (The Community Sea Ice Model, CSIM)³⁸. In reality, most of these CCSM components can execute in serial or multithreaded/distributed-memory/hybrid parallel modes; the range of possibilities is enumerated elsewhere.³⁶ Multithreaded versions of CAM and CLM, plus serial versions of CSIM and POP, have been included in Fig. 1 for illustrative purposes. Workload managers have no issue executing *any one* of these components in isolation. However, CCSM makes *synchronous* use of these four models as components, and accounts for their interaction via an MPI-based coupler (illustrated as CPL in Fig. 1). The resulting workload is Multiple Process, Multiple Data (MPMD) in the components and Single Process, Multiple Data (SPMD) for the interactions between each of the components and the coupler.¹²

Synchronous execution^e of coupled, asymmetric workloads such as

^dCCSM’s architecture allows one of more of these components to be exchanged with alternate models.

^eIn reality, each CCSM component computes, receives data via the coupler, computes,

CCSM present a challenge for workload managers. For this reason, NCAR has an established history of collaborating with providers of WM solutions to enable a functionality that can address workloads such as CCSM. Known as “Task Geometry”, the following section (§2) continues the present use of CCSM, to describe and quantify the impact of this functionality for systems-coupled atmospheric models. Then in §3, Task Geometry is applied conceptually for the first time to scale-coupled atmospheric models. Before identifying conclusions (§5), extending Task Geometry to Grid Computing environments is the subject of §4. Task Geometry contained within a cluster and co-scheduled between clusters, plus interoperability in Grid middleware, are each addressed in turn.

2. Systems-Coupled Atmospheric Models

2.1. *Description*

In use since 2000 at NCAR, “Task Geometry” originally load balanced CCSM workloads in NCAR’s IBM SP-2 operating environment (Black Forest^f) based on IBM AIX. Implemented as a new functionality that was incorporated into WM solution IBM LoadLeveler, via a collaboration between NCAR and IBM, Task Geometry is also in use on NCAR’s IBM p690 environments (Bluedawn and Thunder).⁴⁵ More recently NCAR has incorporated IBM Linux clusters with Myricom Myrinet interconnects, for low-latency, high-bandwidth message passing, into their distributed computing environment. This recent addition motivated NCAR to collaborate with Platform Computing to make Task Geometry *generically* available in WM solution Platform LSF HPC^g — i.e., regardless of the underlying operating environment.

In CCSM, four independent components (i.e., atmosphere, land surface, ocean and sea ice) model the dynamics of the physical systems they represent, and are connected via a flux coupler.³⁶ To fix ideas for the purpose of illustration, suppose that the distributed environment consists of three, quadruple-CPU compute nodes (see Fig. 2).^h In this case, single-threaded land surface (CLM), ocean (POP) and sea-ice (CSIM) components execute

sends data via the coupler, and computes again — at each time step.⁴⁰ In addition, most of the components operate in stages that introduce processing dependencies. Thus ‘synchronous execution’ is an oversimplification.

^fNCAR has decommissioned Black Forest and replaced it with the other environments identified in this section.

^gTask Geometry became officially supported as of Version 6.1 of Platform LSF HPC.

^hOf course, in practice, other nodes are present — i.e., for compute, storage and other

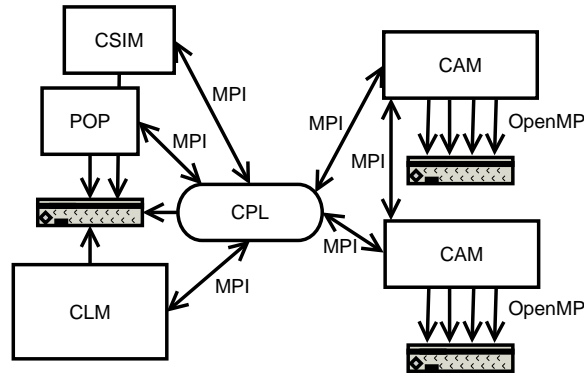


Figure 2. Schematic for a simple CCSM use case in which a multithreaded atmospheric component interacts via MPI with single-thread components for each of the land surface, ocean and sea ice through a flux coupler. Note that single-head arrows denote serial or threaded execution on a compute node, whereas double-head arrows denote interactions via MPI within or between compute nodes.

on a single, quadruple-CPU compute node. Because the component related to the atmosphere (CAM) needs to compute roughly eight times as much as the other components, it's been quadruply threaded for execution on two, quadruple-CPU nodes. Thus the model components operate in an MPMD mode (Fig. 3). The flux coupler (cpl6) also executes on one of the compute nodes (Fig. 2). As illustrated (Fig. 2), the quadruply threaded component representing the atmosphere makes use of the shared memory programming semantics available under OpenMP on each of two compute nodes; interactions between these multithreaded processes occurs via MPI. In contrast, the Single Process, Multiple Data (SPMD) interaction between each of the components and the coupler is facilitated via the distributed memory programming semantics available from MPI (Fig. 3). From Fig. 3, it is also clear that the flux coupler facilitates a feedback process aimed at providing a time-evolving CCSM model run. Component-coupler interactions have been simplified for present purposes; detailed interactions are specified elsewhere.⁴⁰

To load balance CCSM workloads, Platform LSF HPC requires knowledge of the CCSM topology depicted in Fig. 2. This is achieved via the `LSB_PJL_TASK_GEOMETRY` environment variable. In addition to clearly identifying the name of this functionality, naming of this environment variable

purposes. Here 'other' will also include a node (or more) on which the daemons supporting Platform LSF HPC execute.

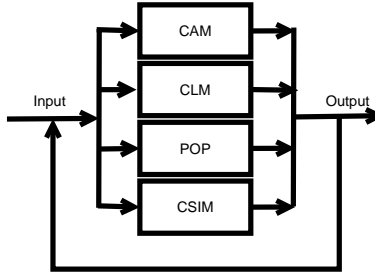


Figure 3. Linear-systems style schematic for a simple CCSM use case in which models for the atmospheric, land surface, ocean and sea ice take inputs and provide outputs that are used in a feedback process to generate results at the next time step. The interaction between model components is facilitated by a flux coupler.

conveys its batch heritage (i.e., via “LSB”) plus its association with the Parallel Job Launcher (i.e., “PJL”) capability of the product. For the example provided in Fig.2, and in the case of the C shell, the appropriate syntax is `setenv LSB_PJL_TASK_GEOMETRY ‘‘{(0,1,2,3),(4),(5)}’’`. This environment variable specifies that tasks (0,1,2,3) correspond to the single-threaded executables (i.e., CLIM, POP, CSIM and CPL) executing on a single, quadruple-CPU compute node, while tasks (4) and (5) correspond to the MPI-OpenMP instance of CAM executing on two, quadruple-CPU compute nodes.ⁱ Collectively the three-tuple values $\{(0,1,2,3),(4),(5)\}$, of the `LSB_PJL_TASK_GEOMETRY` environment variable, convey CCSM topology to Platform LSF HPC. Because script-based submission is the preferred mechanism for initiating CCSM workloads at NCAR, the `LSB_PJL_TASK_GEOMETRY` environment variable is set non-interactively — along with a number of additional environment variables (e.g., these include variables relating to CCSM itself and the parallel operating environment). These CCSM-specific submission scripts also associate components with the appropriate number of threads, and dynamically generate a command file for the relevant parallel operating environment. It is then this command file that is executed by the `mpirun.lsf` functionality of Platform LSF HPC.

In addition to the `LSB_PJL_TASK_GEOMETRY` environment variable setting, the scheduler requires some guidance. In the current example, the maximum number of threads per node is four — a value driven by the atmo-

ⁱNote that the Platform LSF HPC implementation of Task Geometry allows for single and multithreaded tasks in the `LSB_PJL_TASK_GEOMETRY` environment variable.²⁶

sphere component. This maximum-number-of-threads-per-node allocation establishes a processor tile (`ptile`) — i.e., in general a virtual construct for mapping executable units (i.e., processes or threads) to CPUs. `ptile` is one of the possible entries for the `span` directive of the resource-requirement string^{25j} specific to Platform LSF HPC. Because the atmospheric component sets this value to four, the corresponding resource-requirement string is `-R 'span[ptile=4]'`, where `-R` is the `bsub`^k command line option to Platform LSF HPC indicating that a resource-requirement follows. Finally, the scheduler needs to allocate an appropriate number of execution slots^l for CCSM. Multiplying the number of tuples in the `LSB_PJL_TASK_GEOMETRY` environment variable, N_{TUP} , in the model run (i.e., three in the present example) by the `ptile` value (i.e., four in this case) results in the total number of required slots (i.e., 12 in this case). In general,

$$n = ptile \times N_{TUP}.$$

Thus `-n 12` is another `bsub` directive that needs to be set in the CCSM submission script. Additional information on NCAR's ongoing use of task geometry in the CCSM context is available elsewhere.^{12,36,45}

Armed with topology awareness of the model components, and scheduling guidance, Platform LSF HPC dynamically determines the most-appropriate nodes on which to execute CCSM. In addition to Task Geometry, this determination takes into account stated resource requirements, node availability and characteristics, plus relevant scheduling policies (e.g., fairshare). The Task Geometry implementation in Platform LSF HPC supports both proprietary (e.g., IBM Parallel Operating Environment) and Open Source (e.g., LAM/MPI, MPICH-P4 for TCP/IP, MPICH-GM for Myricom Myrinet) MPI implementations. Additional details on Task Geometry under Platform LSF HPC are available in the associated documentation;²⁶ this includes usage notes and examples.

^jResource requirements allow workload-specific information to be detailed in a manner that is digestible by the scheduler. This information may impact how the workload is scheduled and managed.

^k`bsub` is the command used in Platform LSF HPC for job submission. It is detailed elsewhere.²⁶ Graphically oriented interfaces are also available.

^l'Slot' is a term specific to Platform LSF HPC. It corresponds to a resource provider of processing capability. In HPC, there is typically a single slot allocated for a single thread or process.

2.2. Performance Enhancement

By multithreading the most computationally demanding component, i.e., that representing the atmosphere in the above example, Task Geometry achieves a very important outcome: Improved overall application performance. To better appreciate this improvement, and in keeping with the spirit of Amdahl's Law, let t_{CAM} , t_{CSIM} , t_{CLM} and t_{POP} , respectively represent the speed^m for an iteration of the atmosphere, sea ice, land surface and ocean components. The simultaneous, independent execution of these models means that the overall speed for an iteration of CCSM, t_{CCSM} , is given by

$$t_{CCSM} = \max\{t_{CAM}, t_{CSIM}, t_{CLM}, t_{POP}, t_{CPL}\}, \quad (1)$$

where t_{CPL} is the speed impact of invoking the flux coupler. Even though the coupler is not purely overhead, i.e., it also performs computations (e.g., re-gridding) in addition to mediating communication between CCSM components, it is reasonable to assume that $t_{CSIM} \approx t_{CLM} \approx t_{POP} \gg t_{CPL}$, while $t_{CAM} \approx 8 \cdot t_{CSIM}$, the overall speed of a CCSM model run without Task Geometry equates to t_{CAM} . This emphasizes that the atmosphere is the *rate-determining model* in the overall speed of a CCSM model run. This situation is illustrated in Fig. 4.

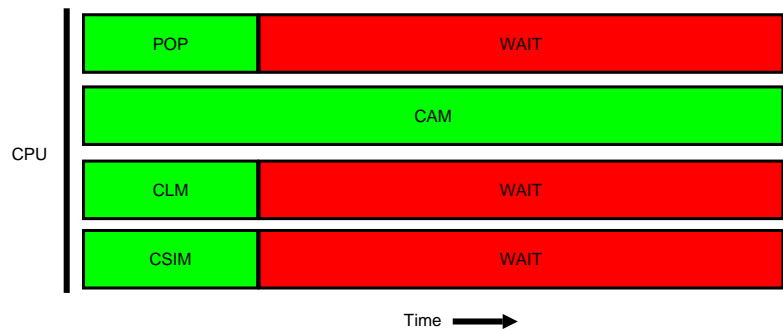


Figure 4. Execution profile for an iteration of multiple-module CCSM. Because the atmosphere component (CAM) takes roughly eight-times longer to complete its calculations, all other modules experience extensive periods of inactivity before interaction involving flux coupler can take place (solid vertical lines). Note that the horizontal dimension represents time, and that this schematic is not to scale.

^mHere *speed* refers to the time required for the model corresponding to the CCSM component to execute.

In reality, Fig. 4 oversimplifies an iteration of CCSM. CAM, CLM and CSIM operate alternately in one of two primary processing modes — i.e., R2S and S2R. Furthermore, these stages form dependencies such that CAM enters its R2S stage while both CLM and CSIM operate in their S2R stage. These processing dependencies result in the execution profile illustrated in Fig. 5.

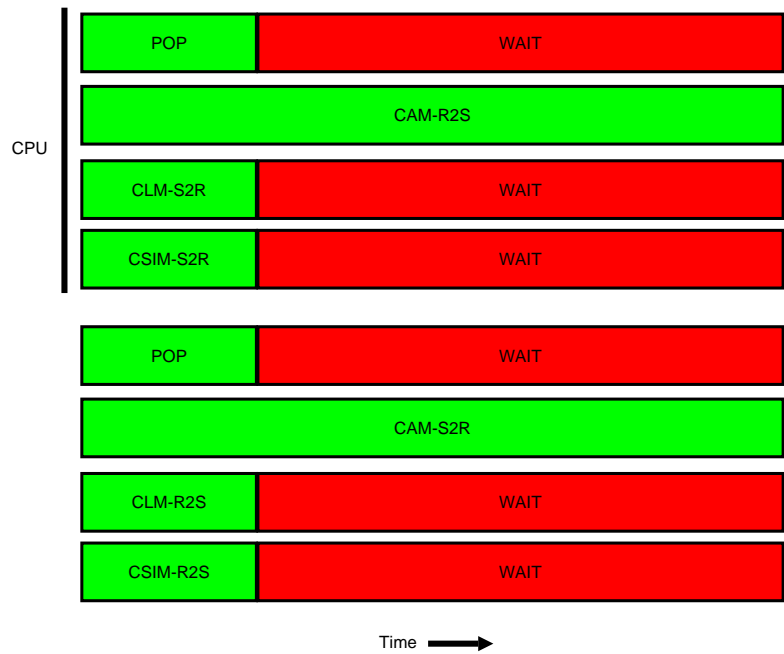


Figure 5. Execution profile for an iteration of multiple-stage, multiple-module CCSM. Because the atmosphere component (CAM-R2S or CAM-S2R) takes roughly eight-times longer to complete its calculations, all other modules experience extensive periods of inactivity before interaction involving flux coupler can take place (solid vertical lines). Note that the horizontal dimension represents time, the lower portion of the figure is a continuation of the upper portion, and that this schematic is not to scale.

The multithreaded implementation of the atmospheric model, when used together with Task Geometry, results in comparable speeds for each of the CCSM components — i.e., $t_{CAM}/N_{THDS} \approx t_{CSIM} \approx t_{CLM} \approx t_{POP} \gg t_{CPL}$. Thus the speed of CCSM before (i.e., t_{CAM}), and after

10

(i.e., t_{CAM}/N_{THDS})

$$t_{CCSM} = \max \left\{ \frac{t_{CAM}}{N_{THDS}}, t_{CSIM}, t_{CLM}, t_{POP}, t_{CPL} \right\}, \quad (2)$$

the use of Task Geometry yields

$$Speedup \approx \frac{t_{CAM}}{t_{CAM}/N_{THDS}} \approx N_{THDS}. \quad (3)$$

This performance enhancement in the overall speed of CCSM, which scales approximately as the number of threads (i.e., N_{THDS}) applied to the atmosphere model, is illustrated in Figs. 6.

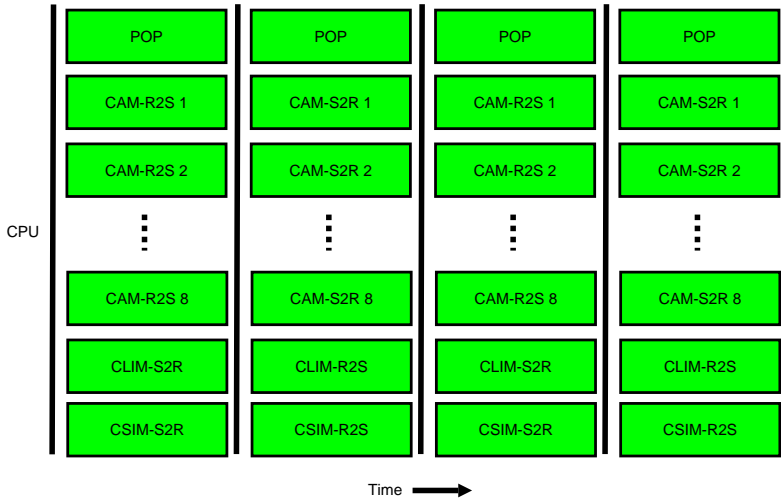


Figure 6. Execution profile for multiple-stage, multiple-module CCSM with Task Geometry. Because the atmosphere component has been multithreaded, represented here as “CAM-x2x 1” through “CAM-x2x 8”, it is now load balanced effectively with all other modules. Periods of computational inactivity are minimized and largely due to interaction involving flux coupler (solid vertical lines). Again the horizontal dimension represents time, and as before this schematic is not to scale.

For completeness, speedup calculations should also incorporate workload-management overhead, t_{WM} . In the case of Platform LSF HPC this ‘overhead’ will likely include:

- Standard life cycle management time. In addition to time spent in the execution phase, workloads will accumulate over-

head while they are pending,¹¹ suspended, or undergoing a check-point/restart/migration operation; and

- Post-dispatch workload initiation and termination times. These times will be increased in cases where execution environment pre- and/or post-conditioning is required.

With the addition of workload-management overhead, Eq. 2 becomes

$$t_{CCSM} = \max \left\{ \frac{t_{CAM}}{N_{THDS}}, t_{CSIM}, t_{CLM}, t_{POP}, t_{CPL} \right\} + t_{WM}. \quad (4)$$

It is standard practice to follow quantification of speedup with quantification of efficiency — i.e., speedup normalized by the number of processors used, N_{CPU_s} . Summing speedup over all parallel (CAM) and serial (CLM, CSIM, POP and CPL) components yields an efficiency of

$$Efficiency = \frac{Speedup}{N_{CPU_s}} \approx \frac{N_{THDS} + Speedup^{Serial}}{N_{CPU_s}} \approx \frac{8 + 4}{12} \approx 100\% \quad (5)$$

for a stage (either R2S or S2R) in an iteration of CCSM. In practice the inherent complexity of CCSM makes efficiency estimation challenging. This complexity derives from two primary sources:

- Multiple modes of parallelization and invocation — As noted previously, most CCSM components can execute in serial, multithreaded, distributed-memory parallel, or even a hybrid mode.^{36,40,45} In addition, components can be invoked in ‘active’, ‘data’ or ‘dead’ modes.^{45^o}
- Component-coupler interaction — As noted previously, each CCSM component computes, receives data via the coupler, computes, sends data via the coupler, and computes again.⁴⁰ In addition, component-coupler interactions are not, in general, synchronized.⁴⁰

Thus mode parallelization and invocation has the potential for impact at each time step, while component-coupler interactions have the potential for impact within a time step. It is this variability that makes detailed speedup and efficiency calculations challenging to impractical. In fact, in practice, it is a combination of ‘hard’ (e.g., run-time statistics) and ‘soft’

¹¹In Platform LSF HPC, workload is placed in a pending state while it awaits requested resources for scheduling and dispatch.

^oBriefly active, data and dead modes correspond respectively to component execution, data manipulation involving the coupler, and simulation.

(e.g., intuition gained from experience) skills that are required for effective use of CCSM.⁴⁰

2.3. *Related Usage*

Task Geometry allows for topology-aware coordination within and between the components involved in a CCSM model run. As alluded to, at the outset of section §2, an MPMD implementation of a data-assimilation application also exists. In the case of Meteorological Service of Canada's (MSC) pre-operational 4D-Var, interaction between a data-assimilation component (based on 3D-Var) and a global model (the MSC Global Environmental Multiscale (GEM) model), is achieved via a coupler.⁴⁶ Thus it would appear that pre-operational 4D-Var could be executed via the Task Geometry implementation in Platform LSF HPC with minimal effort — i.e., effort would need to focus only on the details of the submission script. As Task Geometry accounts for application topology at run time, no additional effort (e.g., recompilation) is required. Other candidates for systems-coupled Task Geometry are likely to include the Integrated Global System Model (IGSM),⁴⁸ atmosphere-biosphere interactions⁹ modeled via CLIMBER,⁴⁷ plus models for Environmental Prediction.¹⁰

3. Scale-Coupled Atmospheric Models

3.1. *Description*

The Task Geometry functionality shows promise for utility in other atmospheric applications where more effort is likely to be required. For example, phenomenological approaches are currently used to parameterize the physics of cloud formation for GCMs. Today, the range of scales requiring representation, makes use of direct numerical simulation of cloud processes (e.g., Cloud Resolving Convection Parameterization,²⁰ CRCP) computationally impractical.²⁸ Although use of IBM BlueGene/L shows significant promise in this context,²⁸ Task Geometry also has the potential to have an impact in more-commoditized operating environments — based on Linux or some other operating system.

To fix ideas for the purpose of illustration,^P suppose that the overall

^PIt is acknowledged that this example grossly oversimplifies the use of a GCM, CRCP, and their interaction. Its express purpose is to demonstrate the utility of Task Geometry in the context of scale-coupled atmospheric models. Subsequent use will determine more-realistic applications of Task Geometry in this context.

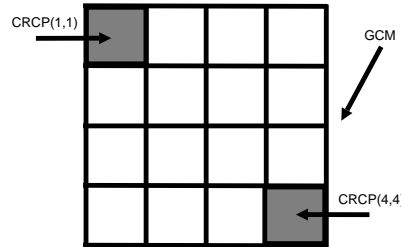


Figure 7. Highly simplified schematic for the solution domain of a GCM (large rectangle) interacting with a model that parameterizes the effects of cloud formation (small rectangles). To illustrate use of Task Geometry in this context, instances of CRCP acting on this grid are numbered CRCP(1,1) through CRCP(4,4).

two-dimensional solution domain for the GCM is represented by the large rectangle shown in Fig. 7; then, each of the 16, small rectangles in the same figure represents a solution domain for *an instance* of the CRCP model. Thus the feedback from CRCP as a subgrid-scale (SGS) model for the GCM can be illustrated as in Fig. 8.

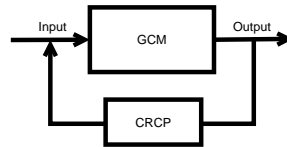


Figure 8. Linear-systems style schematic for a simple GCM-CRCP use case. Here the GCM is involved in a feedback loop with the SGS model, CRCP, to generate results at the next time step. This interaction needs to be facilitated by a coupler.

As in the case with CCSM workloads (§2), Platform LSF HPC acquires topology awareness of a GCM-CRCP workload via an environment variable setting. To further develop the primary example of this section, and supposing that all systems have 4 CPUs, the assignment statement for the topology-awareness environment variable is `setenv LSB_PJL_TASK_GEOMETRY '{(0),(1),(2,3,4,5),(6,7,8,9),(10,11,12,13),(14,15,16,17)}'`. Thus the GCM runs four OpenMP threads on the first-assigned node (the first, i.e., (0) tuple of `LSB_PJL_TASK_GEOMETRY`), the coupler executes as a single thread on the second-assigned node (the second, i.e., (1) tuple of `LSB_PJL_TASK_GEOMETRY`), while CRCP runs four separate instances each in a single-threaded mode on each of the remaining 4 nodes (i.e., the remain-

ing four tuples of `LSB_PJL_TASK_GEOMETRY`). Because the GCM executes in a shared-memory space, `-R 'span[ptile=4]'` is again demanded as a resource-requirement setting. Finally, Platform LSF HPC needs to schedule this workload on $n = ptile \times N_{TUP} = 4 \times 6 = 24$ slots. This is accomplished via the option `-n 24` to the `bsub` command of Platform LSF HPC.

3.2. Performance Enhancement

Without parallelization and Task Geometry, the GCM waits for CRCP to work through each instance of its calculations in sequence. This results in the execution profile illustrated schematically in Fig. 9. Because each instance of the SGS model performs independent calculations, essentially a function call from the main program to CRCP, there exists a latent parallelism that can be exploited. By matching 16 instances of CRCP to 16 low-CPU-count nodes via a coupler and using Task Geometry (Fig. 10), results in the execution profile shown in Fig. 11.

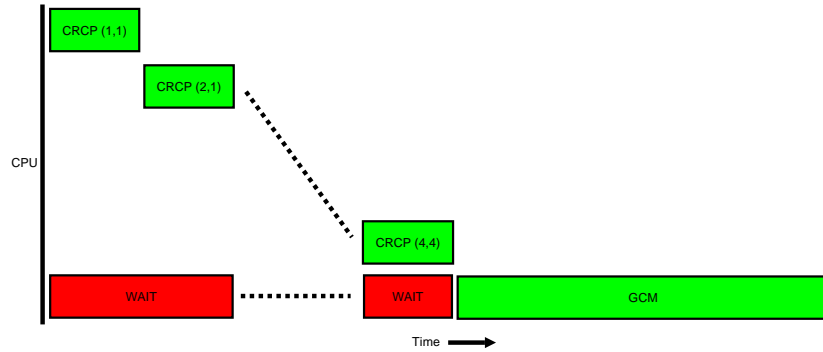


Figure 9. Execution profile for GCM-CRCP interaction. Here the GCM waits until CRCP works through each of its calculations in sequence. Note that the horizontal dimension represents time, and that the schematic is not to scale.

Following the approach for performance analysis of § 2.2, suppose that t_{GCM} , t_{CRCP} , and $t_{GCM+CRCP}$, respectively represent the speed of the GCM computations, the speed of each CRCP function call, and combined speeds. With these assignments, the overall speed of the GCM-CRCP workload is

$$t_{GCM+CRCP}^{BEFORE} = t_{GCM} + \sum_{i=1}^{M_{CALLS}} t_{CRCP}^i, \quad (6)$$

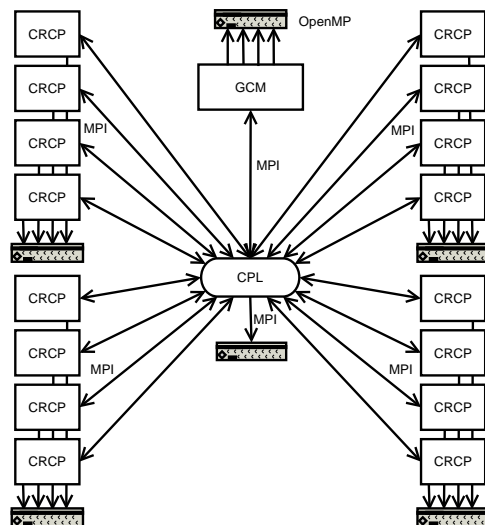


Figure 10. Schematic for a simple use case in which a multithreaded GCM interacts via an MPI-based coupler with multiple instances of CRCP.

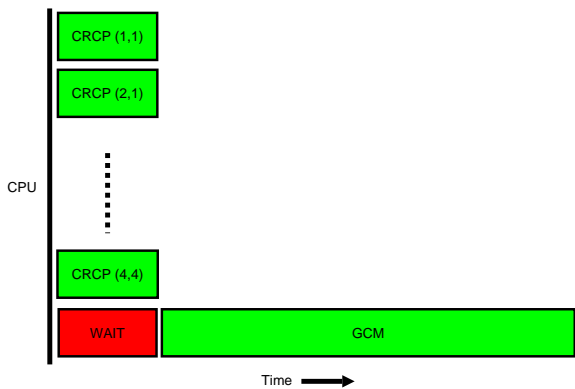


Figure 11. Execution profile for GCM-CRCP interaction after use of Task Geometry. Here the GCM waits until CRCP works through one calculation. Note that the horizontal dimension represents time, the solid vertical lines represent interactions involving coupler, and that the schematic is not to scale.

where the CRCP function calls need to be summed. The parallelization introduced via Task Geometry, and illustrated schematically in Fig. 10,

scales the summation in Eq. 6 by the number of CPUs, N_{CPUs} , i.e.,

$$t_{GCM+CRCP}^{AFTER} = t_{GCM} + \frac{\sum_{i=1}^{M_{CALLS}} (t_{CRCP}^i + t_{CPL}^i)}{N_{CPUs}}. \quad (7)$$

Note that Eq. 7 includes a term to account for the speed of the interaction between CRCP and the GCM via the coupler, and any computational effort (e.g., re-gridding) expended by the coupler, t_{CPL} . The speedup in this context is

$$Speedup = \frac{t_{GCM+CRCP}^{AFTER}}{t_{GCM+CRCP}^{BEFORE}} \approx \frac{1}{N_{CPUs}} \approx 16, \quad (8)$$

if the speed of the coupler is ignored. This means that the overall efficiency is given by

$$Efficiency = \frac{Speedup^{GCM} + Speedup^{CRCP}}{N_{CPUS}}. \quad (9)$$

In the case of the current example,⁹ $Speedup^{GCM} = 4$, $Speedup^{CRCP} = 16$ and $Speedup^{CPL} = 1$. Because the total number of CPUs allocated by Platform LSF HPC is 24, the efficiency is about $(4 + 16 + 1)/24 \approx 87.5\%$. Note that the maximum achievable efficiency is $(4 + 16 + 4)/24 = 100\%$. This overbooking of CPUs encourages GCM and/or CRCP allocations that better account for the requirements of the coupler.

3.3. Related Usage

As another example, experiments like Joint Urban 2003⁵¹ illustrate that the controlled release of effluents motivates challenges and opportunities for the atmospheric sciences in a Homeland Security context. Such scenarios may necessitate an interplay between atmospheric models in (formerly) independent domains. More specifically, and again with the aid of a ‘coupler’, Task Geometry might facilitate modeling involving Large Eddy Simulation^{7,39} (LES) of passive tracers in the Urban Boundary Layer with mesoscale models (e.g., MM5⁴¹ or RF^{34,43}) that address regional effects.³¹ In this case, an SGS model (i.e., the LES model) provides input to a broader-scale model (e.g., MM5 or WRF) regarding dynamics on unresolved scales (i.e., the Boundary Layer). This approach presents a potentially compelling, more-productive alternative to the somewhat redundant, sequential approach to

⁹Assuming a speedup of 4 relating to the 4 OpenMP threads in comparison to a purely serial execution of the GCM.

multigrid methods in common use today.²⁷ Following through at an even higher level of abstraction, these scale-coupled atmospheric models might serve as prognostic instruments in emergency-response platforms.^{1,49}

Use of Task Geometry in scale-coupled contexts was originally suggested in regard to modeling the magnetohydrodynamics of Earth's magnetic field.³¹ In this case, SGS effects are parameterized for the large-scale, three-dimensional geodynamo models¹⁷ by the SGS model¹¹ via a 'coupler'. Because use is made of existing larger-scale SGS models, most of the extra effort needs to focus on the 'coupler'. Fortunately couplers, and the frameworks they serve to define, are active areas of current research.⁸

4. Extension to The Grid

With Platform LSF MultiCluster, clusters based on Platform LSF HPC can be rapidly transformed into a Grid-computing environment.^r Because Linux clusters are already virtualized resources, this transformation exploits existing natural affinities.³³ Given that the specifics of this transformation are detailed elsewhere,^{24,33} attention here focuses on Task Geometry in the context of Platform LSF MultiCluster. More specifically, attention here focuses in turn on use cases driven by the two, primary use models provided by Platform LSF MultiCluster — namely job forwarding and resource leasing.

4.1. Job Forwarding

In production deployment since 1996, the Job Forwarding use model of Platform LSF HPC is based on send/receive queues.^s These queues allow for workload exchange (i.e., forwarding) between co-operating clusters, and serve as cluster-wide containers for managing workload against a rich array of scheduling policies. In the Job Forwarding use model, sites retain a very high level of local autonomy (i.e., sites selectively identify resources available for Grid use) while benefiting from resource sharing that crosses geographic and/or organizational boundaries.^t Typically used to ensure

^rAlthough a more-formal definition of a Grid computing is deferred elsewhere,¹⁶ positing 'Grid computing' as a cluster-of-clusters or clusters-of-clusters serves current purposes.

^sJob Forwarding supports one-to-one, one-to-many, many-to-one and many-to-many relationships between send/receive queues.

^tSecurity is a more-significant concern as organizational boundaries need to be crossed. Together with customers and partners, Platform has crafted solutions that address aspects of this security challenge via AFS, DCE/DFS, The Grid Security Infrastructure⁴ and Kerberos.

maximal utilization of all compute resources across an entire enterprise, Job Forwarding applies to mixed workloads — e.g. serial applications, parametric processing, plus shared and distributed memory parallel applications. Supported scheduling policies include advance reservation, backfill, fairshare, goal-oriented service level agreements, memory and/or processor reservation, preemption, etc.

Ignoring the challenge of data locality,²¹ and under the Job Forwarding use model of Platform LSF MultiCluster, MPMD CCSM model runs are ‘forwarded’ as-a-whole from a submission to execution cluster. This means that all activities, including interaction between the four components representing the atmosphere, land surface, ocean and sea ice, via the coupler, are confined to the execution cluster. Thus Fig. 2 adequately illustrates a CCSM model run in the context of the execution cluster. A similar statement applies to Fig. 10 in the case of GCM-CRCP model runs. In the case of the Job Forwarding use model of Platform LSF MultiCluster, it’s possible to make use of Task Geometry, without any additional effort on the part of the scientist performing the modeling. Additional information on usage of this Platform LSF MultiCluster use model is detailed elsewhere.²⁴

4.2. Resource Leasing

Resource Leasing is a use model that appeared more recently^u in Platform LSF MultiCluster. In this use model, clusters based on Platform LSF HPC have resources (e.g., compute nodes, software licenses, etc.) that can be provided or consumed subject to some agreement. On execution of the workload, the consuming cluster assimilates provided resources in the context of a ‘resource lease’.

To fix ideas for the purpose of illustration, and again ignoring data locality, consider a CCSM model run via Task Geometry involving clusters at the two sites depicted by the dashed-line rectangles of Fig. 12. Each of the clusters is running an instance of Platform LSF HPC and the inter-cluster interaction is facilitated by the Resource Leasing use model of Platform LSF MultiCluster. As illustrated here, a CCSM model run is submitted at Site A. It makes use of compute nodes local to Site A for the land surface, ocean and sea-ice CCSM components, plus a *leased* compute node from Site B for the atmosphere component. Although six additional compute nodes have been exported from Site B, they do not factor into this

^uResource Leasing first appeared in Version 5.0 of Platform LSF MultiCluster.

scenario. As before (e.g., Fig. 2), the flux coupler interacts with all components via MPI. Implicit in this scenario, of course, is the assumption that the interactions between the model components and the coupler are fairly tolerant of the inherent network characteristics (i.e., bandwidth and more importantly latency) between the two sites. It is important to note that this is a bona fide example of co-scheduling an MPI application in a Grid computing context^{29,30} — without any need for human intervention. As in the case of Job Forwarding, Resource Leasing can apply various scheduling policies, and make use of the extensible scheduler framework present in Platform LSF HPC. An analogous example could easily be developed for a scale-coupled use case via Task Geometry.

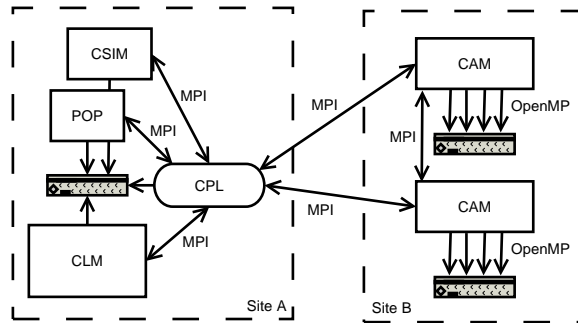


Figure 12. Schematic for a simple CCSM use case in which multithreaded atmospheric components interact via MPI with single-threaded components for each of the land surface, ocean and sea ice through a flux coupler. Note that the atmosphere component executes on leased resources that are physically situated at Site B. All other CCSM components, and the flux coupler, execute on compute resources local to the submission cluster (Site A). Co-scheduled Task Geometry is made possible through the Resource Leasing use model of Platform LSF MultiCluster.

4.3. A Grid of Grids

The highly collaborative nature of atmospheric science ultimately demands interoperability between technologies used to craft Grid infrastructures at the middleware level. This means, for example, that Platform LSF MultiCluster must be able to interoperate with the Globus Toolkit² and/or UNICORE.¹⁵

To facilitate this interoperability in the case of The Globus Toolkit (see Fig. 13), and to pave the way for eventual compliance with the emerging

standards for Grid-enabled Web services,^{5,6} Platform created The Community Scheduler Framework (CSF).²³ CSF is an Open Source contribution by Platform that has recently appeared in Version 4.0 of The Globus Toolkit.³ Although CSF is a recent contribution, it has been generally available for over a year, and has a modest following - including members of the public- and private-sector atmospheric sciences community. With job, reservation and queuing services, CSF provides a comprehensive, standards-compliant framework for the development of Grid-level schedulers. In addition to being Open Source, CSF makes use of Web Services Agreement (WS-Agreement) — a Global Grid Forum (GGF)¹⁴ specification co-authored by the Globus Alliance, IBM and Platform. WS-Agreement details the basis for negotiating consumer-provider relationships for entities belonging to one or more virtual organizations. Although CSF currently targets compute-oriented workloads, it has the potential to factor in other contexts where there is a need to negotiate consumer-provider relationships.³²

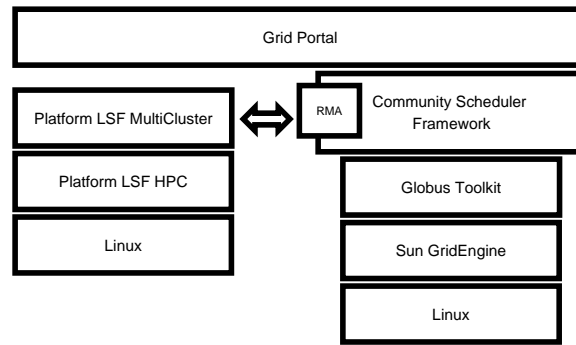


Figure 13. Simplified schematic for interaction at the Grid-middleware level. The Grid based on Platform technology interacts with a Globus-based Grid via a Resource Manager Adapter (RMA) that plugs into The Community Scheduler Framework. Ultimately this extends heterogeneity to the workload-manager level — i.e., Platform LSF HPC indirectly interacts with Sun GridEngine. This Grid of Grids is accessible via a portal.

To complete the solution stacks presented in Fig. 13, it is necessary to add access points to these Grid-based environments. This access is typically mediated via a portal.^{44,50} Because portals abstract away underlying complexity, while serving as community-specific points of presence, this continues to be an active investment area for organizations involved in the atmospheric sciences.^{18,19}

5. Discussion

State-of-the-art atmospheric models present a challenge for WM solutions: They are composed of independent, asymmetric components that are synchronously coupled. The resulting complex application, needs to be managed as a whole in today's highly heterogeneous and highly distributed computing environments. Motivated by this scientific imperative, NCAR collaborated first with IBM and then with Platform Computing to detail a new functionality known as Task Geometry. IBM and Platform subsequently implemented Task Geometry in IBM LoadLeveler and Platform LSF HPC, respectively. The Platform implementation is available for all UNIX and Linux-based environments supported by Platform LSF HPC.

Task Geometry is a construct that allows topological information about the atmospheric model to be conveyed to the workload manager. Armed with this information the workload manager is able to optimally schedule the atmospheric model, subject to policies, on the most-appropriate computational resources. By exploiting the parallelism that already exists within and between model components, Task Geometry allows highly asymmetric workloads to be load balanced. This coordinated load balancing also results in significant performance gains, as well-balanced workloads allow for better overall performance. Speedup and efficiency calculations can be used to quantify the improvement achieved.

Although Task Geometry was originally developed to address CCSM workloads, it has a broader range of applicability. Specifically, CCSM and 4D-Var are two examples of systems-coupled workloads. A second class of applicability was also identified as scale-coupled workloads. Task Geometry has not been applied previously to scale-coupled use cases, and as a result, effort will be needed to frame atmospheric models appropriately. More specifically, the required effort will need to focus on submission specifics (e.g., a submission script) and the development of a coupler. Parameterized effects from unresolved scales for larger-scale models, as well as novel usage motivated by Homeland Security, are examples representative of Task Geometry in scale-coupled use cases. It is also anticipated that scale-coupled usage of Task Geometry could play a role in addressing the spatial aspects of Adaptive Mesh Refinement (AMR)²² — a numerical approach that dynamically enhances spatial and or temporal domains of interest. There is clearly ample motivation for further investigation.

With a storied history of co-dependence spanning a half century already,¹⁰ it is not surprising that NWP is still generating leading-edge

challenges for computing. This is fortunate for other areas of the physical sciences, as they can benefit directly from leading-edge solutions like Task Geometry. In addition to allusions to the study of Earth's geodynamo as a scale-coupled use case for Task Geometry, it is clear that this functionality will find utility in areas as diverse as astronomy, the Life Sciences, and materials science. And this applies to both public- and private-sector HPC applications.

The fractal structure of HPC architectures progresses from the processor to the cluster to the grid — the latter of which can be thought of as a cluster of clusters. Platform LSF MultiCluster allows for the cluster-to-grid progression via two use models — Job Forwarding and Resource Leasing. Whereas Job Forwarding workloads are constrained to execute within a cluster, Resource Leasing workloads are co-scheduled between one or more clusters. Because Task Geometry is compatible with both of these use models, the utility of commodity architectures can be leveraged by making transparent organizational and/or geographic boundaries. This transparency also takes into account the requirement for the highest levels of interoperability between grid middleware, a requirement that underlines the importance of open standards.¹⁴

Acknowledgments

The authors acknowledge Mariana Vertenstein of the Climate and Global Dynamics Division, NCAR, for various discussions on CCSM, plus Hanna Mullane of ECMWF and Karen Sopuch of Platform Computing for their encouragement and assistance. Additionally, IL acknowledges David McMillan of York University (Toronto, Canada) for ongoing discussions regarding modeling the geodynamo.

References

1. N. R. Adam, V. Atluri, and V. P. Janeja. Agency interoperation for effective data mining in border control and homeland security applications. *The Fifth National Conference on Digital Government (dg.o 2004)*, Seattle, Washington, USA, May 2004.
2. The Globus Alliance. The Globus Toolkit. <http://www.globus.org>.
3. The Globus Alliance. GT4: CSF. <http://www.globus.org/toolkit/docs/4.0-contributions/csf/>.
4. The Globus Alliance. Overview of The Grid Security Infrastructure. <http://www.globus.org/security/overview.html>.
5. The Globus Alliance. Towards Open Grid Services Architecture. <http://www.globus.org/ogsa>.

6. The Globus Alliance. The WS-Resource Framework. <http://www.globus.org/wsrfr>.
7. S. P. Arya. *Introduction to Micrometeorology, Second Edition*. Academic Press, 2001.
8. V. Balaji. A comparative study of coupling frameworks: The MOM case study. *Eleventh ECMWF Workshop: Use of High Performance Computing in Meteorology*, Reading, UK, October 2004.
9. G. Brasseur, W. Steffen, and K. Noone. Earth System focus for Geosphere-Biosphere Program. *Eos Trans. Am. Geophys. Un.*, 86(22):209, 213–214, 2005.
10. G. Brunet. The first hundred years of Numerical Weather Prediction. In I. Kotsireas and D. Stacey, editors, *Proceedings of The 19th International Symposium on High Performance Computing Systems and Applications, HPCS 2005*, pages 276–279. The IEEE Computer Society, 2005.
11. B. A. Buffet. A comparison of subgrid-scale models for large-eddy simulations of convection in the Earth’s core. *Geophys. J. Int.*, 153:753–765, 2003.
12. G. Carr. Porting and performance of the Community Climate System Model (CCSM3) on the Cray X1. *Eleventh ECMWF Workshop: Use of High Performance Computing in Meteorology*, Reading, UK, October 2004.
13. W. D. Collins, C. M. Bitz, M. L. Blackmon, G. B. Bonan, C. S. Bretherton, J. A. Carton, P. Chang, S. C. Doney, J. J. Hack, T. B. Henderson, J. T. Kiehl, W. G. Large, D. S. McKenna, B. D. Santer, and R. D. Smith. The Community Climate System Model: CCSM3. *J. Climate*, 11(6):to appear, 2005.
14. The Global Grid Forum. The global grid forum. <http://www.ggf.org/>.
15. The UNICORE Forum. Unicore. <http://www.unicore.org>.
16. I. Foster. What is The Grid? A Three Point Checklist. *GRIDtoday*, 1(6), 2002.
17. G. A. Glatzmaier, D. E. Odgen, and T. L. Clune. Modeling the Earth’s dynamo. In R. S. J. Sparks and C. J. Hawkesworth, editors, *The State of the Planet: Frontiers and Challenges in Geophysics*, volume IUGG Volume 19 of *Geophysical Monograph 150*, pages 13–24. American Geophysical Union, 2004.
18. M. W. Govett. A WRF portal for model test and verification. *NOAA Technical Report*, 2003.
19. M. W. Govett. Grid Computing: The development of a portal for model test and verification. *NOAA Tech 2003*, Silver Spring, Maryland, USA, October 2003.
20. W. W. Grabowski. Coupling cloud processes with the large-scale dynamics using the Cloud-Resolving Convection Parameterization (CRCP). *J. Atmos. Sci.*, 58(9):978–997, 2001.
21. G. Hayward and I. Lumb. Data agnostic resource scheduling in the grid. In I. Kotsireas and D. Stacey, editors, *Proceedings of The 19th International Symposium on High Performance Computing Systems and Applications, HPCS 2005*, pages 230–235. The IEEE Computer Society, 2005.
22. R. Henderson, D. Meiron, M. Parashar, and R. Samtaney. Parallel com-

- puting in computational fluid dynamics. In J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, and A. White, editors, *Sourcebook of Parallel Computing*, pages 93–144. Morgan Kaufman, 2003.
23. Platform Computing Inc. The Community Scheduler Framework. <http://sourceforge.net/projects/gcsf>.
 24. Platform Computing Inc. *Using Platform LSF MultiCluster, Version 6.1*. Platform Computing Inc., 2004.
 25. Platform Computing Inc. *Using Platform LSF, Version 6.1*. Platform Computing Inc., 2004.
 26. Platform Computing Inc. *Using Platform LSF HPC, Version 6.1*. Platform Computing Inc., 2005.
 27. E. Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge University Press, 2003.
 28. R. Loft. Price and power aware approaches to advancing atmospheric science. *Eleventh ECMWF Workshop: Use of High Performance Computing in Meteorology*, Reading, UK, October 2004.
 29. I. Lumb. HPC Grids. In A. Abbas, editor, *Grid Computing: A Practical Guide to Technology and Applications*, pages 119–133. Charles River, 2003.
 30. I. Lumb. Production HPC reinvented. *login.*, 28(4):15–22, 2003.
 31. I. Lumb. The metric of scale: Real-world experiences via infrastructural software. *High Performance Computing and Communications Conference: Exploring the Next Frontier, 19th Annual*, Newport, Rhode Island, USA, April 2005.
 32. I. Lumb and K. D. Aldridge. Grid-enabling the Global Geodynamics Project: The introduction of an XML-based data model. In I. Kotsireas and D. Stacey, editors, *Proceedings of The 19th International Symposium on High Performance Computing Systems and Applications, HPCS 2005*, pages 216–222. The IEEE Computer Society, 2005.
 33. I. Lumb and C. Smith. Integrating Linux clusters into the Grid. *SysAdmin Magazine, Clustering Supplement 2003*, 12(8):49–52, 2003.
 34. J. Michalakes, J. Dudhia, D. Gill, T. Henderson, J. Klemp, W. Skamarock, and W. Wang. The Weather Research and Forecast Model: Software architecture and performance. In *Proceedings of the Eleventh ECMWF Workshop: Use of High Performance Computing in Meteorology*. World Scientific, 2005.
 35. NCAR. The Community Atmosphere Model. <http://www.cesm.ucar.edu/models/atm-cam/>.
 36. NCAR. Community Climate System Model (CCSM3). <http://www.cesm.ucar.edu/models/ccsm3.0>.
 37. NCAR. The Community Land Model. <http://www.cgd.ucar.edu/tss/clm/>.
 38. NCAR. The Community Sea Ice Model. <http://www.cgd.ucar.edu/ccr/bettge/ice/>.
 39. NCAR. Large Eddy Simulation. <http://www.mmm.ucar.edu/research/surface/les.html>.
 40. NCAR. Load Balancing CCSM3. <http://www.cesm.ucar.edu/models/ccsm3.0/ccsm/doc/UsersGuide/UsersGuide/node10.html>.
 41. NCAR. MM5 Community Model. <http://www.mmm.ucar.edu/mm5/>.

42. NCAR. The Parallel Ocean Program. <http://climate.lanl.gov/Models/POP/>.
43. NCAR. The Weather Research and Forecasting Model. <http://www.wrf-model.org/>.
44. NICE. NICE EnginFrame. <http://www.enginframe.com>.
45. M. Page, G. Carr, I. Lumb, and B. McMillan. NCAR CCSM with task-geometry support in LSF. *ScicomP11*, Edinburgh, Scotland, June 2005.
46. S. Pellerin. MPMD implementation of pre-operational MSC 4Dvar on IBM p690. *Eleventh ECMWF Workshop: Use of High Performance Computing in Meteorology*, Reading, UK, October 2004.
47. V. Petoukhov, A. Ganopolski, V. Brovkin, M. Claussen, A. Eliseev, C. Kubatzki, and S. Rahmstorf. CLIMBER-2: A climate system model of intermediate complexity: I. Model description and performance for present climate. *Clim. Dyn.*, 16:1–17, 2000.
48. R. G. Prinn. Complexities in the climate system and uncertainties in forecasts. In R. S. J. Sparks and C. J. Hawkesworth, editors, *The State of the Planet: Frontiers and Challenges in Geophysics*, volume IUGG Volume 19 of *Geophysical Monograph 150*, pages 297–305. American Geophysical Union, 2004.
49. SAP and Rutgers University. Next-generation emergency response. *SAP SAPPHERE 2005, SAP Innovation Pavilion*, Boston, Massachusetts, USA, May 2005.
50. The GridPort Team. GridPort. <http://gridport.net>.
51. United States The Department of Energy. Joint Urban 2003: Atmospheric dispersion study in Oklahoma City. <http://ju2003.pnl.gov/>.