

# NCAR CCSM with Task-Geometry Support in LSF

Mike Page and George Carr, Jr.  
National Center for Atmospheric Research,  
P.O. Box 3000,  
Boulder, CO 80307-3000, USA  
E-mail: mpage, gcarr@ucar.edu

Ian Lumb and B. McMillan  
Platform Computing Inc.,  
3760 14th Avenue,  
Markham, Ontario L3R 3T7, Canada  
E-mail: ilumb,bmcmillang@platform.com

## 1. Overview

Independent components for modeling ocean, atmosphere, land and sea-ice behavior in changing climate scenarios are coupled together in NCAR's Community Climate System Model (CCSM). At the implementation level, each component is a separate binary resulting in a Multiple Process, Multiple Data (MPMD) application. At the implementation level, some components support a hybrid mode use of the Message Passing Interface (MPI) and OpenMP. System support for applications that utilize both shared and distributed memory under separate program control is a requirement for effective use of available computational resources.

In the Fall of 2000 IBM added a 'task geometry' capability to their LoadLeveler (LoadLeveler for AIX 5L V3.2: Using and Administering) batch subsystem manager to support CCSM and other similar applications at NCAR on an AIX-based, SP-architecture system. Recently NCAR and Platform Computing Inc. have also collaborated to enhance Platform LSF HPC for AIX with task geometry support. By addressing the locality of related tasks, Platform LSF HPC for AIX manages CCSM simulations on NCAR's Bluedawn and Thunder systems.

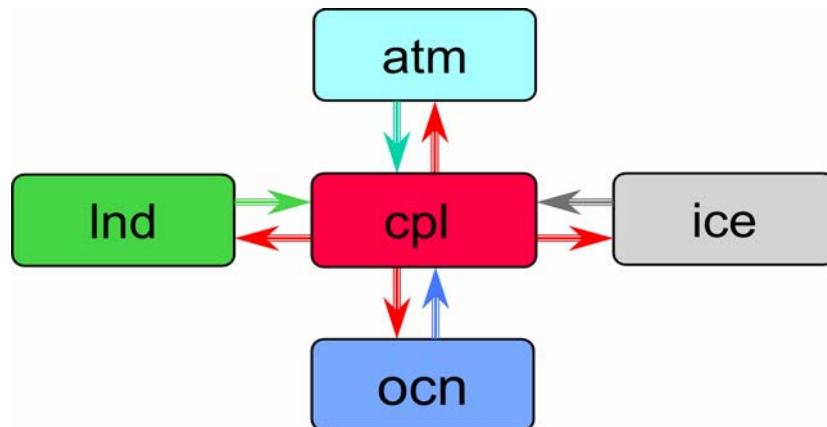
Recently NCAR acquired an IBM Linux cluster with low-latency, high-bandwidth message passing enabled via the Myricom Myrinet interconnect. This system utilizes LSF for batch subsystem management and also supports CCSM's hybrid on Lightning's thinner nodes.

After reviewing the enhancement, usage examples will illustrate how NCAR researchers use this capability to manage asymmetric assignment of CCSM components under LoadLeveler and LSF. A throughput assessment comparing LoadLeveler and LSF is made for the CCSM application utilizing task geometry.

## 2. A General Description of CCSM

The Community Climate System Model 3.0 (CCSM3) is a computational tool used for modeling the Earth's climate. Development of CCSM is supported primarily by NSF and DOE and is freely available to the climate research community for studies of the Earth's climate as it existed centuries ago (at present, only CCSM2 supports this capability), its evolution to current conditions and to predict the climate that may evolve over the next several hundred years [Collins, 2005].

CCSM3 is modular in design and functionality. It is an assemblage of four individual component models which individually model the Earth's atmosphere, ocean, land surface and sea ice. These four components communicate computational results through a coupler module in a "hub and spoke" configuration (Figure 1). A detailed description of CCSM3 is available at <http://www.cesm.ucar.edu/models/ccsm3.0>.



**Figure 1. The Hub and Spoke Design of CCSM3**

The modular design of CCSM3 allows it to be run with a variety of component model variations. Any of the components can be removed and replaced with alternative treatments of the respective geophysical system. CCSM3 currently supports a number of run-time options for model components. The options are outlined in Table 1.

**Table 1. CCSM Model Types, Names and Use**

Model Type	Name	Use
Active	atm lnd ice ocn	Actively models physics and dynamics of respective geophysical system
Data	datm dlnd dice docn	Reads existing datasets that were previously written by the dynamical models and passes the resulting data to the coupler
Dead	xatm xlnd xice xocn	Generate unrealistic forcing data internally, require no input data and can be run on multiple processors to simulate the software behavior of the fully active system

Data models provide an effective means of establishing new physical parameterizations for other Active CCSM components since their behavior is prescribed; they do not respond to or provide feedback to other components. Dead models are intended to evaluate communications handling of the coupler.

The atmospheric component of CCSM3 is the Community Atmosphere Model 3.0 (CAM3) and is a descendant of other NCAR atmospheric climate models [Washington, 1982], [Williamson, 1983]. Standard resolutions for the atmospheric computations are T85 for 1.4 degree (128x256x26), T42 for 2.8 degree (64x128x26), and T31 for 3.75 degree (48x96x26). The 26-level vertical grid utilizes a hybrid pressure coordinate system. [Collins, 2004] [Vertenstein, 2004].

Ocean modeling in CCSM3 is performed by the Parallel Ocean Program (POP), version 1.4.3, developed at Los Alamos National Laboratory [Smith, 2002]. Typical resolutions are one degree (320x384x40) and three degree (100x116x25).

The land component is the Community Land Model 3.0 (CLM3). This component operates on the same grid resolution as the atmospheric component. [Levis, 2004] [Oleson, 2004].

The sea ice model, CSIM5, is based on CICE from Los Alamos National Laboratory. The ice model uses the same displaced-pole grid and same resolution as the ocean component [Briegleb, 2004].

The flux coupler, CPL6, performs a number of the data and grid conversions required to pass data from component to component [Craig, 2005].

Each component of CCSM3 is a distinct binary element and generally runs on a separate processing element or across groups of processing elements (sharing of processors by multiple binaries has not been tested) in a computationally distributed environment.

Components may use MPI only or MPI with OpenMP for managing parallel computations and data communications. This varies across components and the component type as shown in Table 2. [Vertenstein 2004]

**Table 2. CCSM3 Components Parallelization**  
(Vertenstein, M, T., et. al.)

<b>Component Name</b>	<b>Version</b>	<b>Type</b>	<b>MPI Parallel</b>	<b>OpenMP Parallel</b>	<b>Hybrid* Parallel</b>
<b>cam</b>	cam3	active	YES	YES	YES
<b>datm</b>	datm6	data	NO	NO	NO
<b>latm</b>	latm6	data	NO	NO	NO
<b>xatm</b>	dead	dead	YES	NO	NO
<b>clm</b>	clm3	active	YES	YES	YES
<b>dlnd</b>	dlnd6	data	NO	NO	NO
<b>xlnd</b>	dead	dead	YES	NO	NO
<b>pop</b>	ccsm_pop_1_4	active	YES	NO	NO
<b>docn</b>	docn6	data	NO	NO	NO
<b>xocn</b>	dead	dead	YES	NO	NO
<b>csim</b>	csim5	active	YES	NO	NO
<b>dice</b>	dice6	data	NO	NO	NO
<b>xice</b>	dead	dead	YES	NO	NO
<b>cpl</b>	cpl6	active	YES	NO	NO

\*Hybrid – using both MPI and OpenMP

By its design, CCSM places an initial requirement on computational platforms of MPMD applications support (multiple binaries). This MPMD design option enables an additional opportunity for the use of OpenMP along with MPI within the same component. Doing so leads to improved computational performance on some platforms. CCSM is quite flexible in that the number of OpenMP threads can also be varied from component to component, a non-uniform use of OpenMP within an MPMD application.

### 3. Task Geometry

Task geometry is a terminology associated with a feature of advanced batch management subsystem that allows a user to specify the configuration of (MPI) computational tasks across nodes of a distributed processing architecture according to a scheme that enhances parallel computation and data communications performance.

**Table 3. The History of Blackforest at NCAR**

Year	Node Type	Node Count	Processors per Node	Peak GFlops
1999	Winterhawk I	148	2	237
2000	Winterhawk II	151	4	906
2001	Winterhawk II Nighthawk II	315 3	4 16	1962

Task geometry capability was first developed at NCAR for the AIX LoadLeveler batch management subsystem in 2000 to enhance the computational performance of MPMD programs on Blackforest such as CCSM.

In AIX LoadLeveler, task geometry is specified by the keyword syntax:

```
#@ task_geometry={(task id,task id,...)(task id,task id, ...) ... }
```

**NOTE:** In LoadLeveler, task id refers to an MPI task

In this expression, curly brackets enclose parenthesized lists of task/node assignments. The number of nodes assigned to a run is equal to the number of pairs of parentheses. Within the parentheses, consecutive integer values, starting with 0, assign tasks to nodes. Thus:

```
#@ task_geometry={{(0)(1)(2,3,4,5)(6)}
```

would call for the reservation of four nodes and assign one task to each of the first two nodes allocated from the resource pool, four tasks to the third node and one to the fourth. What is not completely evident from this declaration is the number of processors per node that are available. In the instance that the number of tasks assigned to a node is less than the number of processors attached to that node, the task might be advantageously multi-threaded, i.e. run multiple OpenMP threads from a master task. OpenMP threads are not accounted for in the AIX task geometry syntax because LoadLeveler is not 'thread-aware' and treats every job (or task) as if it only has one thread. For systems that utilize IBM's Workload Manager (WLM) to support LoadLeveler, LoadLeveler can become more thread-aware if the number of ConsumableCpus is specified as part of the job setup. NCAR does not use WLM so it is the responsibility of the user to know, before job submission, what task layout is desired.

Task Geometry also requires specification of a Unix command file (referenced by the LoadLeveler parameter MP\_CMDFILE) that defines the threadedness of each binary to be run in the model. The command file is generated on the fly by the CCSM run script and depends on a system-dependent environment file for ordering of the components in the command file. A command file corresponding to the above task geometry specification might look like this if the environment file specifies the ordering of components to be cam, clm, ice, ocn and cpl:

```
env OMP_NUM_THREADS=4 cam  
env OMP_NUM_THREADS=4 cam  
env OMP_NUM_THREADS=1 clm  
env OMP_NUM_THREADS=1 clm  
env OMP_NUM_THREADS=1 dice  
env OMP_NUM_THREADS=1 docn  
env OMP_NUM_THREADS=1 cpl
```

This example is drawn from a CCSM test run conducted on NCAR's Bluedawn system. Bluedawn is an IBM RS600 cluster with Power4 processors connected by a Colony switch and running AIX51. It contains four four-way nodes. With this specification it now becomes evident that this particular CCSM run has specified that the tasks running on both nodes 0 and 1 will be active cam and each of these nodes will run four OpenMP threads. Node 3 will run a pair of active clm tasks that share work via MPI communications (no OpenMP threading). The same node will also run the dice and docn data models. Both of these components will run single-threaded. The CCSM coupler, cpl6, will run single-threaded and alone on the last and fourth node to coordinate data interchange between the other four CCSM components but note that, according to this setup, three processors on the fourth node will be idle. Note that leaving processors idle is not a recommended configuration for CCSM users.

#### 4. Task Geometry Benchmarks

Thunder is a larger RS6000 system at NCAR that has been upgraded to a Federation switch. Thunder consists of four 16-way nodes and is like Bluedawn in all other respects. Three benchmark runs were conducted on Thunder to demonstrate the performance benefits of task geometry and compare the performance of pure MPI with hybrid MPI/OpenMP in CAM3, the most compute-intensive CCSM component.

All tests assigned 48 processors to CAM3 at T31\_gxv5 resolution. The first test defined 48 MPI processes with no OpenMP threads (single threaded) across three nodes. The second test defined 12 MPI processes each with 4 OpenMP threads each and the third defined 6 MPI processes with 8 OpenMP threads each. Processor counts for CAM and the other components are shown in Table 4. Active ocn (POP 1.4.3), Active ice (CSIM5), Active clm (CLM3) and the coupler (CPL6) all run on the same node assigned to processors as indicated.

**Table 4. CCSM Component Configuration**

Component	Number of Processors
CAM	48 Test 1 – Pure MPI Test 2 – 12 MPI Tasks, 4 OpenMP threads each Test 3 – 6 MPI Tasks, 8 OpenMP threads each
POP	4
ICE	2
LND	8
CPL	2

CAM3 is regularly used in this hybrid MPI/OpenMP implementation demonstrated in the second and third tests. Internally, the dynamics portion of CAM3 is not threaded and all communication among CAM3 processes is performed by MPI. The physics portion of CAM3 can be threaded via OpenMP under the control of an MPI process. Adding threads can reduce the volume of MPI communications and increase the average size of a message, improving performance. This is true on IBM SP systems such as Bluedawn and Thunder where several processors share a single network port.

CCSM performance is measured in simulated years per wall clock day for a specific number of processors (or Years/Day). The performance for the three configurations is shown in Table 5.

**Table 5. CCSM Performance**

Configuration	Simulated Years Per Day
48 MPI processes, no OpenMP threads	18.87
12 MPI processes, 4 OpenMP threads per MPI process	20.34
6 MPI processes, 8 OpenMP threads per MPI process	21.94

## 5. Load Sharing Facility – LSF

In 2003, NCAR/SCD began evaluation of the Load Sharing Facility, LSF, from Platform Computing. LSF offered an attractive alternative to continued maintenance of an in-house batch job scheduler and a great deal of adaptability to facilitate fair share allocation of NCAR system resources to the multiple communities of users that SCD supports. LSF also offered an attractive solution to batch management needs on high performance Linux cluster systems that SCD was investigating at the same time. An anticipated movement towards a heterogeneous computing environment that includes multiple architectures and operating systems coupled to a desire to make those systems interoperable drew more focus to LSF as a solution to this type of growth and diversity.

LSF was first installed on the Bluedawn and Thunder systems for internal testing. Since CCSM is NCAR's flagship climate model, it was highly desirable to be able to run this code efficiently under LSF as had been the case for LoadLeveler and task\_geometry since 2000.

In early April 2004 NCAR initiated discussions with Platform on the development of a task geometry feature for LSF. The result was an equivalent capability and a syntax with which LoadLeveler users were immediately familiar. Differences exist but a conversion from AIX LoadLeveler to LSF is straightforward if examples of all of the common programming models (serial, pure MPI, pure OpenMP, MPMD with MPI and hybrid) are provided to users.

For LSF the task geometry syntax is:

```
setenv LSB_PJL_TASK_GEOMETRY "{(0)(1)(2,3,4,5)(6)}"
```

which closely resembles the LoadLeveler directive. Parentheses and curly brackets have the same meaning. But in this case, rather than a batch subsystem directive, an environmental variable is set.

LSF possesses the thread awareness that LoadLeveler without WLM does not. LSF actually treats MPI tasks and OpenMP threads the same when it counts processes. LoadLeveler only counts MPI tasks and leaves tracking of OpenMP threads to the user.

If `LSB_PJL_TASK_GEOMETRY` is set as shown and additionally the user specifies the number of processors required for the run with the “`BSUB -n ntasks`” directive, then for `ntasks` equal to 16 the four nodes indicated by the four pairs of parentheses will run four threads on each node (assuming that no processor runs more than one thread as is the usual practice). This is enforced in LSF with a “`#BSUB -R “span[ptile=4]”`” directive. The `-R` directives in LSF are used for resource reservation such as processors, memory or software licenses (Using Platform LSF HPC 6.1). `ptile=4` specifies the use of four processors per node. The value 4 is derived from a count of the maximum number of threads on any of the requested nodes.

Completion of this setup for a CCSM run under LSF is the same as was discussed for a LoadLeveler run. The command file name is passed to LSF as an option to the `mpirun.lsf` command that substitutes for `poe`. On an AIX system such as Bluedawn or Thunder, `poe` is declared as `elim` (external load information manager) by an “`BSUB -a ‘poe’`” LSF directive.

The salient content of a CCSM run script for LSF would be written in this way:

```
#BSUB -a 'poe'           # select the POE elim
#BSUB -n 16             # number of tasks
#BSUB -R "span[ptile=4]" # the largest number of tasks asked for in task geom
#BSUB -x               # exclusive use of node (not_shared)
.
.
.
# Setup LSF Task Geometry
setenv LSB_PJL_TASK_GEOMETRY "{(0)(1)(2,3,4,5)(6)}"
.
.
.
mpirun.lsf -cmdfile cmdfile
```

## 6. Throughput benchmarks

Tests to analyze batch job throughput were conducted on NCAR’s Bluedawn system to compare performance of LoadLeveler against LSF. There was a reasonable expectation that the two systems would be roughly equivalent but this was not completely borne out by the tests. At this time the difference in throughput performance is not fully understood but the results, even if not totally conclusive, show factors that are worthy of further discussion.

The tests were developed from a suite of applications that are commonplace in the typical NCAR job mix. They include:

POP – run as a standalone application, not as a CCSM component for this test

MM5 – a regional weather forecast model

WRF – a regional weather forecast model

MHD1 – a magnetohydrodynamics model of solar activity

MHD – similar to MHD1

Lite1 – a fortran features regression tester

Lite2 – same as Lite1

CCSM – the climate model discussed above

In the charts below, the application name is listed to the left with a numerical indication of the processor count attached. Lite1 and Lite2 run in a single processor but do not allow node sharing so their effective processor count is four. A large variety of POP jobs was run with varying memory and time requirements as inferred by the names of individual jobs.

The jobs were submitted to batch queues of varying priority ranging from high (P – premium) to low (E - economy). Run queues are noted to the right of the application names. Bluedawn was initially fully idled prior to job submission, CCSM was the first job submitted because it was known that this would fully occupy all of Bluedawn's processors and allow the remaining job load to be quickly submitted and give the job scheduler opportunity to determine job ordering from that point onward.

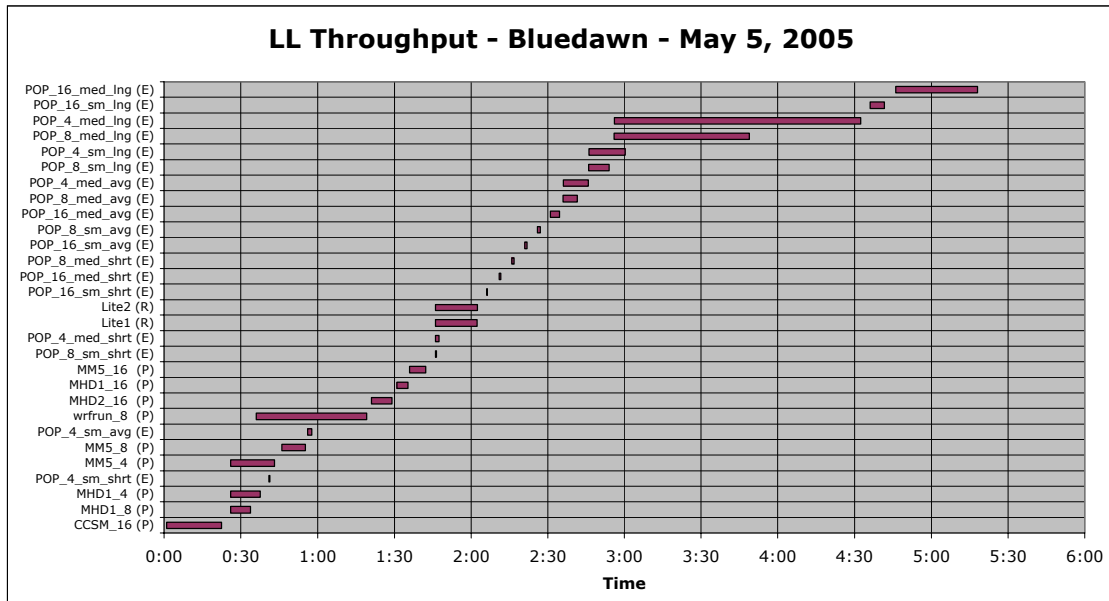
Job start and finish times are indicated by the horizontal bars in the graphs.

Results from the LoadLeveler test show a preference for job scheduling based on queue priority. This was especially the case in other tests that were run with BPS in place. This is possibly one of the configuration parameters that was not fully removed in the LoadLeveler reconfiguration that had to be done for this test. It is also possible to see evidence of significant idle time between task termination and task initiation most likely due to a polling interval for LoadLeveler that could be reduced for better performance.

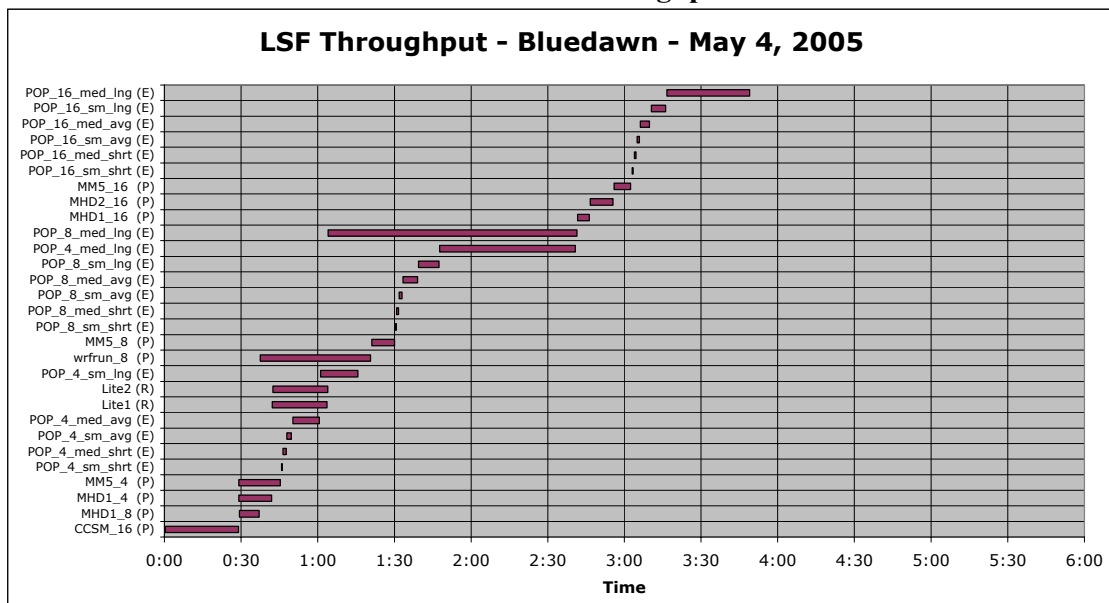
LoadLeveler required almost 5 1/2 hours to process this job load whereas LSF completed the test in less than 4 hours. The initial job in the mix, CCSM, differs in runtime between the two systems but this is undoubtedly due to the data-staging requirements encountered by LSF but avoided in the following LL test. This was an oversight of the tester.

LSF certainly does well for this job mix in terms of resource scheduling. But a concrete conclusion of superior or inferior performance of either of these systems would be premature until a closer investigation of the configuration of each one can be conducted.

### Chart 1. Loadleveler Job Throughput Test



### Chart 2. LSF Job Throughput Test



## 7. Conclusions

LSF has been installed and utilized on NCAR systems that operate alongside those that utilize LoadLeveler for batch subsystem management. With the development of task geometry support, it has shown itself to be every bit as functional as LoadLeveler.

Future usage of LSF enables capabilities that are not possible with LoadLeveler. This includes inter-platform operability, grid capabilities and even the possibility of running applications such as CCSM across diverse architectures and operating systems to achieve a new level of load balancing and performance. LSF also enables a centralized queuing structure for all of this diversity that will be an area future exploration.

NCAR recently acquired and installed the Linux cluster system mentioned early on in this article and uses LSF alone for the batch management subsystem. There has not been sufficient time as yet to fully explore what this system means for CCSM. The authors hope to be able to present results for this system in the near future.

### **Acknowledgements:**

The authors wish to express their sincere appreciation for the helpful guidance provided by Mariana Vertenstein (NCAR/CGD) in the many discussions of CCSM usage and its description. The assistance of John Ellis (NCAR/SSG) was extremely critical, very timely and generously offered and was just as critical to the successful completion of the tests that were conducted for this report as the actual testing itself. Without the help and advice of Vadim Elisseev (Platform Computing) who found that magic switch that made things work these tests could not have been completed.

Thank you, all.

### **References**

- Briegleb, B. P., C. M. Bitz, \*E. C. Hunke, \*W. H. Lipscomb, M. M. Holland, J. L. Schramm, and R. E. Moritz, 2004. Scientific description of the sea ice component in the Community Climate System Model, Version Three. NCAR Tech. Note NCAR/TN-463+STR, 70 pp
- Collins, W.D, 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, 2005. The Community Climate System Model: CCSM3 to be published in the CCSM Special Issue, J. Climate, **11(6)**, to appear
- Collins, W. D., P. J. Rasch, B. A. Boville, J. J. Hack, J. R. McCaa, D. L. Williamson, J. T. Kiehl, B. Briegleb, C. Bitz, \*S.-J. Lin, M. Zhang, and Y. Dai, 2004. Description of the NCAR Community Atmosphere Model (CAM 3.0). NCAR Tech. Note NCAR/TN-464+STR, p. 226.
- Craig, A. P., R L Jacob, B Kauffman, T Bettge, J Larson, E Ong, C Ding, Y He. Cpl6: The New Extensible, High-Performance Parallel Couler for the Community Climate System Model. Submitted for publication Special Issue on Climate Modeling, Int'l J. HPC Apps., Vol 19, #3, August, 2005
- Drake, J. B., P W Jones, G R Carr Jr. Overview of the Software Design of the CCSM. Submitted for publication Special Issue on Climate Modeling, Int'l J. HPC Apps., Vol 19, #3, August, 2005

Levis, S., G. B. Bonan, M. Vertenstein, and K. W. Oleson, 2004. The Community Land Model's Dynamic Global Vegetation Model (CLM-DGVM): Technical description and user's guide. NCAR Tech. Note NCAR/TN-459+IA, p. 50.

LoadLeveler for AIX 5L V3.2: Using and Administering at <http://publib.boulder.ibm.com/infocenter/clresctr/index.jsp?topic=/com.ibm.cluster.loadl.doc/loadla32/am2ug300/am2ug30020.html>

Oleson, K. W., Y. Dai, G. Bonan, \*M. Bosilovich, R. Dickinson, \*P. Dirmeyer, \*F. Hoffman, \*P. Houser, S. Levis, G.-Y. Niu, P. Thornton, M. Vertenstein, Z.-L. Yang, and X. Zeng, 2004. Technical description of the Community Land Model (CLM). NCAR Tech. Note NCAR/TN-461+STR, 174 pp

Smith, R.D. and P. Gent 2002. Reference manual for the Parallel Ocean Program (POP). Los Alamos Unclassified Report LA-UR-02-2484.

Using Platform LSF HPC 6.1 at [http://platform.com/services/support/docs\\_home.asp](http://platform.com/services/support/docs_home.asp)

Vertenstein, M, T Craig, T Henderson, S Murphy, G R Carr Jr, N Norton, CCSM3.0 User's Guide, June 25, 2004, <http://www.cesm.ucar.edu>.