JPL

Ice Sheet System model

Parallel Capabilities

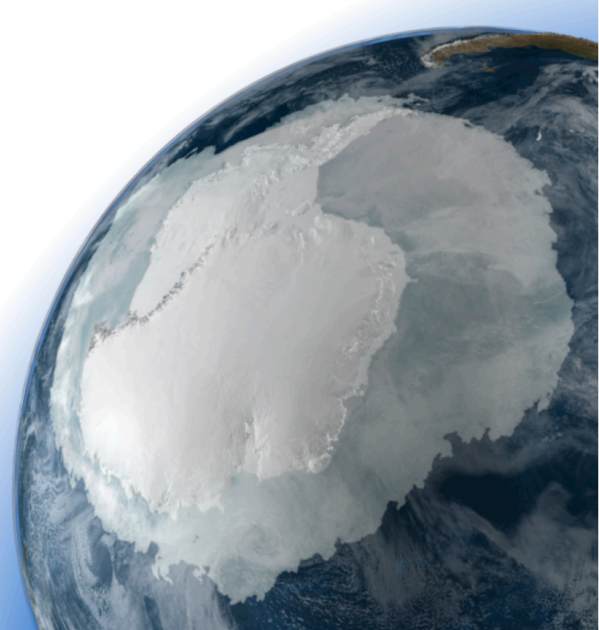
Eric LAROUR[1], Eric RIGNOT[1,3], Mathieu MORLIGHEM[1,2], Hélène SEROUSSI[1,2] Chris BORSTAD[1], Feras HABBAL[1,3], Daria HALKIDES[1,4], Behnaz KHAKBAZ[1], John SCHIERMEIER[1], Nicole SCHLEGEL[1]

[1]Jet Propulsion Laboratory - California Institute of Technology

[2]Laboratoire MSSMat, École Centrale Paris, France

[3]University of California, Irvine

[4]Joint Institute for Regional Earth System Science & Engineering, UCLA

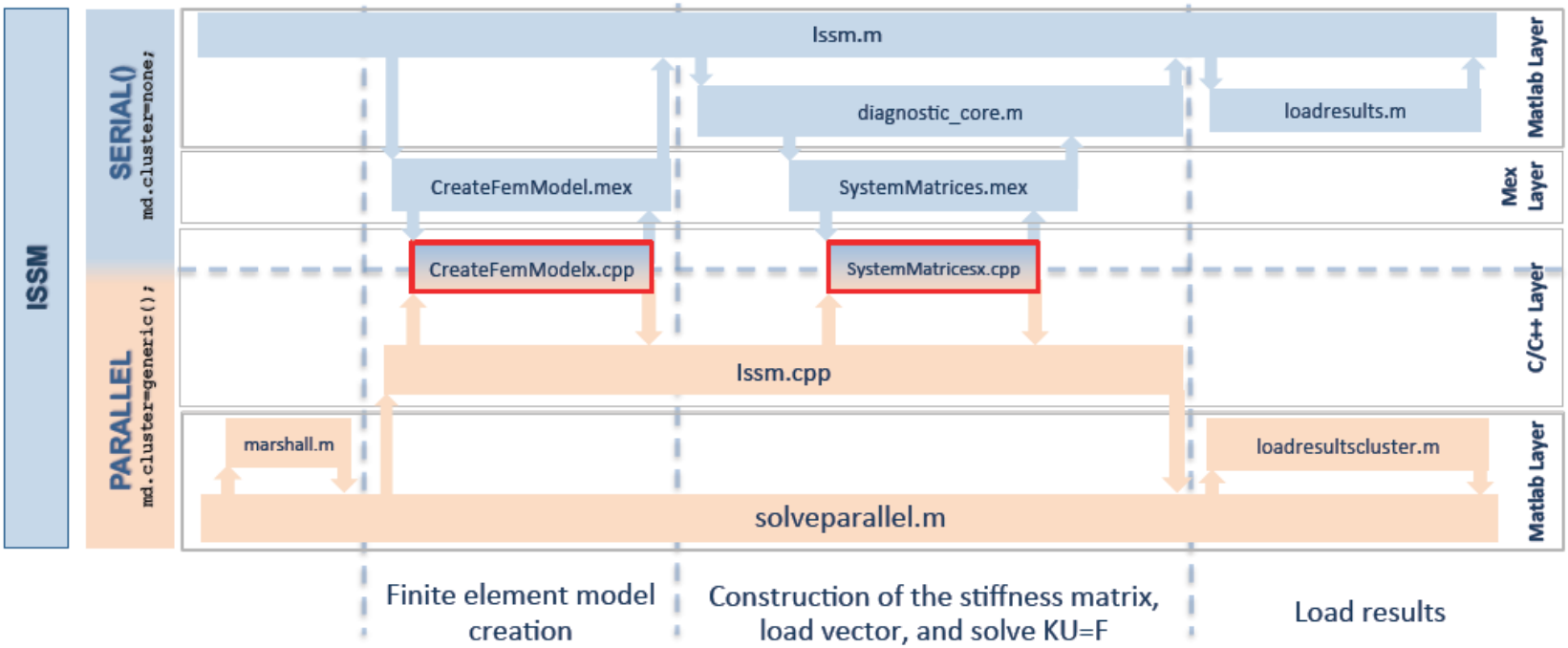


ISSM

Outline

Introduction

ISSM code can be compiled into one executable that can be run independent of Matlab, in any cluster, be it shared or distributed.



Parallel libraries

ISSM relies on a series of libraries to implement parallelism:

- PETSc Portable, Extensible Toolkit for Scientific Computation [Balay et al., 1997, Balay et al., 2008, Balay et al., 2009]. PETSc is a suite of data structures and routines for the scalable solution of scientific applications modeled by partial differential equations. Used mainly for its parallel structures (Vec and Mat objects) and iterative parallel solvers.
- MPICH1,2: Message Passing Interface [Gropp et al., 1996, Gropp and Lusk, 1996] to manage parallel communications between all cpus during solution sequences.
- METIS: Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices [Karypis and Kumar, 1998]. This package is used to partition objects such as elements and vertices across a cluster. This partitioning scheme results in partitions that have equal numbers of elements on each cluster node.
- MUMPS: Multifrontal Massively Parallel Sparse direct solver [Amestoy et al., 2001, Amestoy et al., 2006]. Direct solver that suffers few convergence issues. Relied upon often for the solution of any system of equations.

Parallel structures

ISSM relies on several objects (C language) implemented in PETSc for parallelization:

- Vec object: holds a distributed vector with different parts distributed row wise on every cpu. The solution vector computed by issm.exe is an example of Vec object.
- Mat object: holds a distributed matrix with different parts distributed row wise on every cpu. The stiffness matrix computed by issm.exe in almost all drivers is an example of Mat object.
- Ksp and PC objecst: PETSc structures that hold a solver context and a conditioner context. This is inherent to PETSc and the way solvers are run.

Both Mat and Vec objects can be serialized for output, or during the solution, for ease of use in indexing.

On the Matlab side

To launch on a specific cluster, type the following:

```
1  md.cluster=generic('name',oshostname(),'np',3);
2  md=solve(md,'DiagnosticSolution);
```

The following settings are available on a generic cluster:

```
>> md.cluster

ans =

class 'generic' object 'ans' =
    name: murdo
login:
np: 3
port: 0
codepath: /Users/issm/Desktop/issm/trunk/bin
executionpath: /Users/issm/Desktop/issm/trunk/execution
valgrind: /Users/issm/Desktop/issm/trunk/externalpackages
valgrindlib: /Users/issm/Desktop/issm/trunk/externalpacka
valgrindsup: /Users/issm/Desktop/issm/trunk/externalpacka
```

Matlab cluster classes

Cluster classes are implemented for different types of clusters, such as NASA Pleiades cluster, or generic cluster, or linux-64 clusters, etc. The cluster classes can be found in

```
$pwd
/Users/issm/Desktop/issm/trunk/src/m/classes/clusters
$ ls
castor.m  cosmos.m  gemini.m  generic.m  none.m  pfe.m  pollux.m  README
```

The main routines implemented in a cluster are:

```
%GENERIC cluster class definition
%
%   Usage:
%      cluster=generic('name','astrid',);
%      cluster=generic('name','astrid','np',3);
%      cluster=generic('name',oshostname(),'np',3,'login','username');

classdef generic
    properties (SetAccess=public)
+-- 12 lines: % ----------------------------------------------------------------------
end
methods
+-- 24 lines: function cluster=generic(varargin) % ------------------------------------
+-- 14 lines: function disp(cluster) % ---------------------------------------------------
+--  9 lines: function checkconsistency(cluster,md,solution,analyses) % ----------------
+-- 44 lines: function BuildQueueScript(cluster,md) % ------------------------------------
+-- 29 lines: function LaunchQueueJob(cluster,md,options)% -------------------------------
+-- 27 lines: function Download(cluster,md)% ---------------------------------------------
end
end
```

These routines are called by solve.m script to run on that particular cluster. First build queuing scripts, then send them to the cluster, where issm.exe is compiled, along with a binary input file, and run on the cluster.

Matlab cluster classes

Once the results are computed, we download the results back and post-process. The download is controlled by the setting waitonlock.

```
>> md.settings

ans =

general settings parameters:
io_gather               : 1             -- I/O gathering strategy for result outputs (default 1)
lowmem                  : 0             -- is the memory limited ? (0 or 1)
results_as_patches      : 0             -- provide results as patches for each element (0 or 1)
output_frequency        : 1             -- frequency at which results are saved in all solutions with
waitonlock              : 30            -- maximum number of minutes to wait for batch results, or re
```

If waitonlock is Inf, the solve routine will lock until user tells it the download is finished. If waitonlock fails (CTRL-D from user, or matlab crash), one can alway rely on downloading from the cluster:

```
>> md=loadresultsfromcluster(md);
```

On the Cluster side

On the cluster side, the LaunchQueueJob routine of the cluster class will ship a binary file (marshalled data of the model) onto the cluster, and launch a script. Here is the directory (trunk/execution) where things happen:

```
[issm@issm test240-12-12-2011-23-6-43-24624]$ pwd
/Users/issm/Desktop/issm/trunk/execution/test240-12-12-2011-23-6-43-24624
[issm@issm test240-12-12-2011-23-6-43-24624]$ ls
test240-12-12-2011-23-6-43-24624.tar.gz  test240.errlog  test240.outbin  test240.petsc
test240.bin                              test240.lock    test240.outlog  test240.queue
```

test240.bin is the test240 binary input file to issm.exe. test240.queue is the script that is launched on the cluster. test240.petsc holds the petsc settings. Once the cluster runs, we get a test240.outbin (which is downloaded by md=loadresultsfromcluster(md), or by md=solve(md,...) + outlog and errlog files.

Launching on Cluster side

Here is a typical launch script such as test240.queue

```
#!/bin/sh
mpiexec -np 3 /Users/issm/Desktop/issm/trunk/bin/issm.exe DiagnosticSolution /Users/issm/Desktop/issm/
```

Here is the typical petsc settings test240.petsc:

```
%Petsc options file: test240.petsc written from Matlab solver array

+NoneAnalysis
-mat_type mpiaij
-ksp_type preonly
-pc_type lu
-pc_factor_mat_solver_package mumps
-mat_mumps_icntl_14 120
-pc_factor_shift_positive_definite true

+DiagnosticVertAnalysis
-mat_type mpiaij
-ksp_type preonly
-pc_type lu
-pc_factor_mat_solver_package mumps
-mat_mumps_icntl_14 120
-pc_factor_shift_positive_definite true
```

test240.queue can be used again and again until satisfying results. test240.petsc settings can be modified until convergence is reached. If you don't want to handle what's going on on the cluster side, you don't need to, as the routine md=solve(md,...) handles everything transparently for you.

Bibliography I


Amestoy, P. R., Duff, I. S., Koster, J., and L'Excellent, J.-Y. (2001).
A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling.
SIAM J. Matrix Anal. Appl., 23(1):15–41.


Amestoy, P. R., Guermouche, A., L'Excellent, J.-Y., and Pralet, S. (2006).
Hybrid scheduling for the parallel solution of linear systems.
Parallel Computing, 32(2):136–156.


Balay, S., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H. (2008).
Petsc users manual.
Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory.


Balay, S., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H. (2009).
PETSc Website.
http://www.mcs.anl.gov/petsc.

Bibliography II


Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F. (1997).
Efficient Management of Parallelism in Object Oriented Numerical Software Libraries.
In Arge, E., Bruaset, A. M., and Langtangen, H. P., editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press.


Gropp, W., Lusk, E., Doss, N., and Skjellum, A. (1996).
A high-performance, portable implementation of the MPI message passing interface standard.
Parallel Computing, 22(6):789–828.


Gropp, W. D. and Lusk, E. (1996).
User's Guide for `mpich`*, a Portable Implementation of MPI*.
Mathematics and Computer Science Division, Argonne National Laboratory.
ANL-96/6.


Karypis, G. and Kumar, V. (1998).
A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices.
University of Minnesota.

Bibliography III


Thanks!