Running MITgcm forward and adjoint on TACC machines and sverdrup

An T. Nguyen, Patrick Heimbach, Helen Pillar, Kylie Kinne, Kirstin Schulz, Matthew Goldberg

January 25, 2023

0 Modules, optfiles, idev:

0.1 Modules

Prior to discussing setting up the specific domains, we need to check that the modules appropriate for the MITgcm are loaded. This is my list of modules on **stampede2**, which is loaded by default in my **.bashrc**, which is available at **/home1/03901/atnguyen/.bashrc_stampede2**.

```
login1.stampede2 $ module list

Currently Loaded Modules:
  1) autotools/1.1     2) cmake/3.16.1    3) xalt/2.10.2
  4) TACC              5) intel/18.0.2    6) libfabric/1.7.
  7) impi/18.0.2       8) python2/2.7.15  9) netcdf/4.3.3.1    10) git/2.24.1

```

On **Frontera**, these are my modules in **/home1/03901/atnguyen/.bashrc_frontera**:

```
 cd ~/
 -rwx------ 1 atnguyen G-816140 2379 Jan 22  2020 .bashrc_frontera

login3.frontera(1004)$ module list
Currently Loaded Modules:
  1) autotools/1.2   3) pmix/3.1.4      5) xalt/2.10.2   7) intel/19.1.1   9) python3/3.7.0  11) git/2.24.1
  2) cmake/3.16.1    4) hwloc/1.11.12   6) TACC          8) impi/19.0.9   10) netcdf/4.6.2
```

0.2 Optfiles

In addition we have also created optfiles that link appropriate libraries and compilers for the skylake nodes on stampede2 and frontera:

```
/work/03901/atnguyen/BitBucket/computing/optfiles/
  linux_amd64_ifort+mpi_stampede2_aste
  linux_amd64_ifort+mpi_frontera_aste1
```

On sverdrup the optfiles are at:

```
/home/atnguyen/nansen/computing/optfiles/
linux_amd64_ifort+mpi_svedrup_seaice_tides
linux_amd64_idort+mpi_sverdrup_aste
optfile_mvapich2_sverdrup
```

0.3 Interactive node

To ask for an interactive skylake node on stampede2 for 2 hours (max), the command is:

```
idev -A atn-startup -p skx-dev -t 2:0:0
```

To run mpi process on the interactive node, the syntax is:

```
ibrun -n ${nprocs} ./mitgcmuv${forwadj}
```

Model Domain Configurations

There are five domain configurations covered in this note: (1) regional **ASTE1080**, (2) **global1080**, (3) regional **ASTE4320**, (4) **ASTE270**, and (5) **ASTE90** (miniASTE). The number "1080" in (1) and (2) refers to a grid we use that has 4*1080 points along the earth's equator (covering 360° longitudes at ∼110km/1°). This translates to ∼360°/(4*1080)*(110km/1°) = 9km horizontal grid spacing at the equator. For ASTE4320, ASTE270 and ASTE90, the horizontal grid spacings correspond to 5.3km, 36.7km and 110km

1 ASTE1080

1.1 Domain and domain decomposition

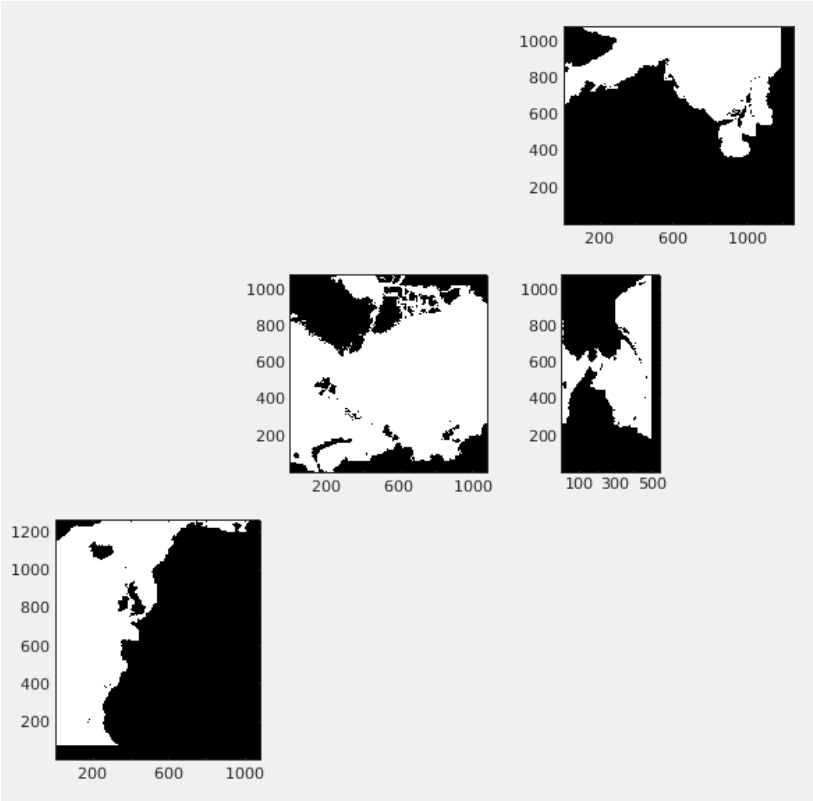

Figure 1: ASTE1080 domain

This is a regional domain, containing a total of 1080 x 4140 horizontal grid points and 90 vertical levels, yielding a total of 402,408,000 grid points. **Fig. 1** shows the full ASTE1080 domain. Note that the domain is **not** a rectangle of size 1080 x 4140 but rather is composed of several "faces" (four in **Fig. 1**) such that when re-assembled back into a rectangle the size is 1080 x 4140. The decomposition of the domain is done along the horizontal slice (non-continuous 1080 x 4140 grid points, see **Fig. 1**) via tile partitioning. An example of the decomposition into tiles of size 90 x 90 is shown in **Fig. 2**. Once decomposition is finished, the MITgcm allows for exclusion of "land" tiles, such that we only solve for "wet tiles", e.g., less than the maximum number of 1080x4140/90/90=552 tiles and less than the total 402,408,000 grid points.

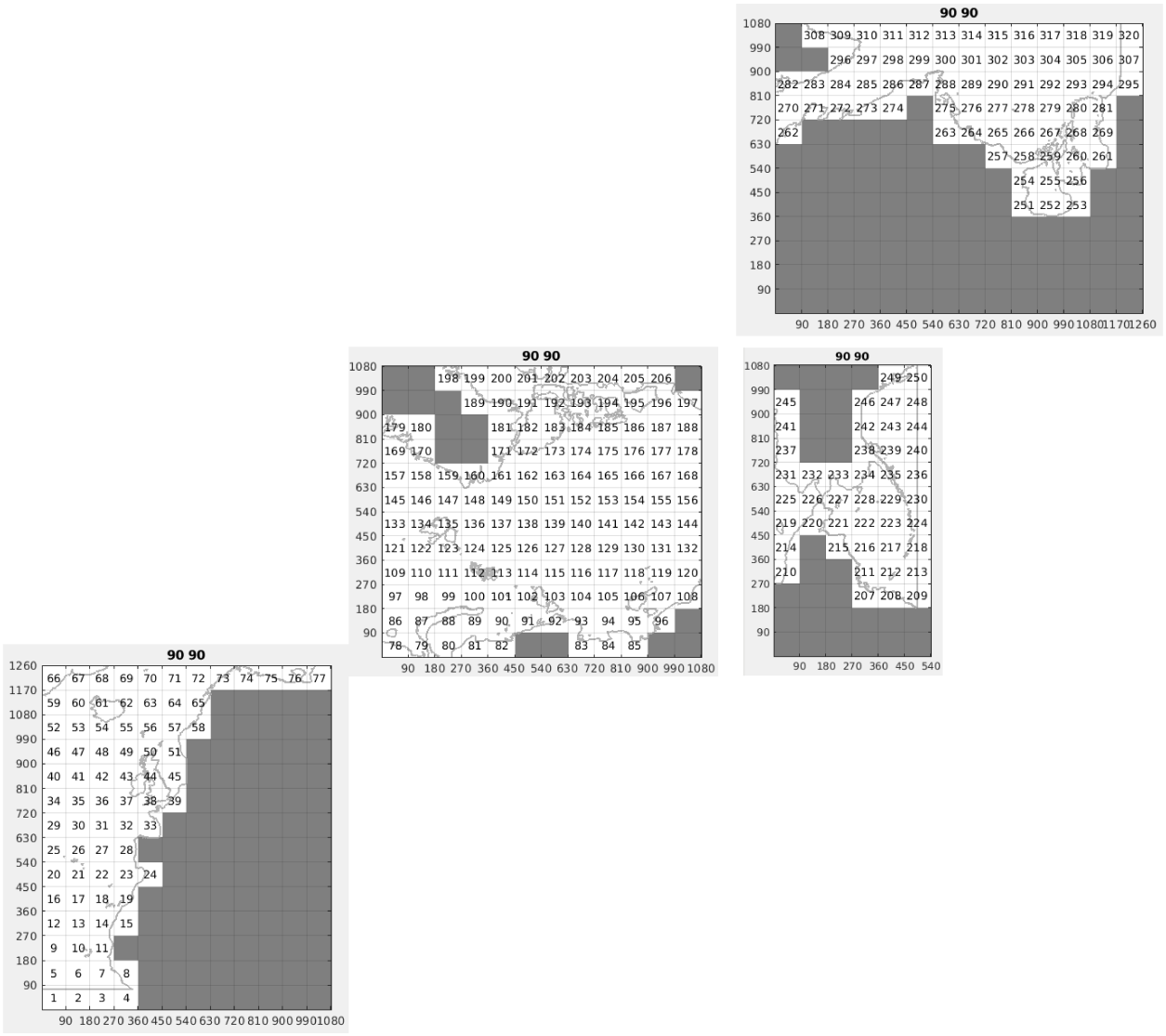

Figure 2: ASTE1080 decomposed into 90x90 tiles. The gray area is "blanked out" such that only the white area, a total 320 tiles (shown as incremental numbers, instead of 552 tiles), will be solved for. For this decomposition, the total number of grid points will be 90x90x320 x 90-vertical levels = 233,280,000 grid points, which is ∼57% of the total number of grid points for the entire domain. The smaller the tile size, e.g., 18x18 or 15x15, the more gray areas we can remove from the computation. With this 90x90 decomposition, a total of 320 tiles translates to 320 CPUs is needed, and that each CPU will solve for 90x90=8100 grid points in the horizontal (plus grid overlaps, and also to be multiplied with number of vertical levels). This yields a max of 8100 x 90-vertical levels = 729,000 grid points per CPU, which can be taxing for the CPU memory if the state is stored. Memory demands can be reduced by using smaller tile size, with the trade-off being the increased communications between CPUs and nodes.

Table 1 shows some of the possible decompositions and associated total number of CPUs required. For a forward calculation, tile sizes ~30x30 or larger are OK. For the adjoint calculation, due to the memory required to store the state, tile sizes 15x15, 18x18, or 20x20 are needed.

sNx	sNy	No. of tiles = No. of CPUs (nProc)
90	90	320
90	60	463
60	60	659
45	60	860
45	45	1149
sqrt(512 45	36	1430
36	36	1784
36	30	2099
30	30	2484
20	30	3671
20	20	5438
20	18	6027
18	18	6695
15	15	9405

Table 1: Decomposition of the ASTE1080 regional domain into various tile sizes. sN[x,y] are the number of grid points in the [x,y]-dir per tile. Note that the tiles need not be squared, but likely for optimal communication/exchange with edges of different tile, i.e. different CPUs, it's desirable to keep the tiles close to a square. There are more possibilities that what's listed here, but these are some samples for testing. Horizontal lines are only drawn to improve ease of reading.

1.2 Location of files

The main directory for this configuration is at:

```
/work/03901/atnguyen/MITgcm_c67/MITgcm/
/work/03901/atnguyen/MITgcm_c67/mysetups/aste_1080x1260x540x90/
    code_tides_mds_prof_sp1/
    build_tides_mds_prof_sp1/
    input_tides/
    aste1080_tides.bash
/work/03901/atnguyen/BitBucket/computing/optfiles/
    linux_amd64_ifort+mpi_stampede2_aste
    linux_amd64_ifort+mpi_frontera_aste1
```

To compile in your own directory, copy all of the above to the appropriate location. Below, I use an example with **$LOCALDIR** being **/scratch/03901/atnguyen/localdir**:

```
  export atnwork='/work/03901/atnguyen/'
  export LOCALDIR='/scratch/03901/atnguyen/localdir/'
  cd $LOCALDIR
```

```
    cp -rp $atnwork/MITgcm_c67/MITgcm/ ./MITgcm
    cp -rp $atnwork/MITgcm_c67/mysetups/aste_1080x1260x540x90/code_tides_mds_prof_sp1 ./code
    cp -rp $atnwork/MITgcm_c67/mysetups/aste_1080x1260x540x90/input_tides ./NAMELISTS
    cp -p $atnwork/MITgcm_c67/mysetups/aste_1080x1260x540x90/aste1080_tides.bash ./aste1080.bash
    cp -p  $atnwork/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_stampede2_aste ./
    cp -p  $atnwork/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_frontera_aste1 ./
    cp -p /home1/03901/atnguyen/.bashrc_stampede2 ./bashrc_stampede2
s
```

1.3 Instruction to compile and run the model

1.3.1 Compiling:

The compilation of the code requires the main-branch **MITgcm** directory and the specific **code** for the ASTE1080 domain, as well as the appropriate optfile depending on whether we're using stampede2 or frontera,

```
    linux_amd64_ifort+mpi_stampede2_aste
    linux_amd64_ifort+mpi_frontera_aste1
```

Before compiling, choose the tile decomposition by reading off **Table. 1** and pick the row you want, for example, **90x90x320**. The first manual step is to cp the **SIZE.h** file associated with this tile set in **code/**:

```
    cd $LOCALDIR/code
    cp SIZE_h_90x90x320 SIZE.h
```

Next, create the **build** directory and compile the MITgcm:

```
    cd $LOCALDIR
    mkdir build
    cd build
  ../MITgcm/tools/genmake2 -mpi -of=../linux_amd64_ifort+mpi_stampede2_aste -mods=../code -rd=../MITgcm
    make depend
    make -j 4
```

where the **"genmake2"** command creates the file **Makefile** in the directory which allows you to check all the module lists and compare to mine, and the last "make" command creates a **"mitgcmuv"** executable and the "-j 4" will compile in parallel to speed things up, using 4 CPUs in this example. One last step I do is to move this executable to match the chosen tile decomposition:

```
    mv mitgcmuv mitgcmuv_90x90x320
```

NOTE: if you have to **re-compile**, in the same **build** dir, there are two levels of clearing you need:
a) If all modules are correct, and you're only changing **SIZE.h** to recompile, then the file **Makefile** can be re-used as follows:

```
    cd $LOCALDIR/build
    make CLEAN
    make makefile (note small case ‘‘m’’ in ‘‘makefile’’)
    make depend
    make -j 4
```

b) If your modules are incorrect, you cannot use the existing **Makefile** that was created in the **build/** dir and will simply need to empty the dir and recompile from scratch using **genmake2**.

1.3.2 Prepare and submit jobscript:

Next, edit the jobscript file **aste1080.bash**. Line 7 to choose the account:

```
 7 #SBATCH -A atn-startup
```

Next, un-comment the correct option associated with the choice **90x90x320** in this example:

```
10 #SBATCH -N 7
11 #SBATCH -n 336
59 nprocs=320
60 snx=90
61 sny=90
```

Finally, edit the line to point to where your $LOCALDIR is:

```
89 scratchdir=/scratch/03901/atnguyen/localdir
```

Now submit the job:

```
sbatch aste1080.bash
```

I've a successfully created all of the above steps and have an example run using **90x90x320** at:

```
/scratch/03901/atnguyen/localdir/
  linux_amd64_ifort+mpi_frontera_aste1
  linux_amd64_ifort+mpi_stampede2_aste
  MITgcm/
  code/
  build/
  NAMELISTS/
  aste1080.bash
  run_hourly_pk0001577880/ <-- the succesful run
```

1.4 Expected results

In the run directory, you should be able to see progress and monitored physical oceanographic statistics and any warnings in:

```
/scratch/03901/atnguyen/localdir/run_hourly_pk0001577880/
   STDOUT.0000, STDERR.0000
```

The run is set to output at quite high frequency, with the output files dumped in the subdirectory

```
/scratch/03901/atnguyen/localdir/run_hourly_pk0001577880/; diags/
```

2 Global1080

2.1 Domain and domain decomposition

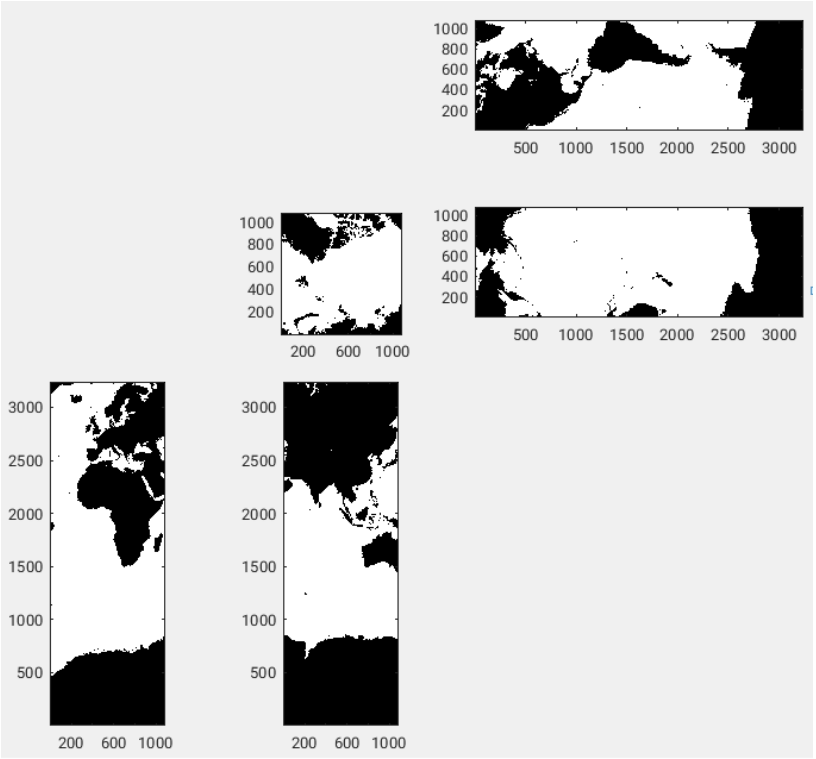

Figure 3: global1080 domain

This is a global domain, containing a total of 1080 x (1080x13) horizontal grid points and 90 vertical levels, yielding a total of 15,163,200*90=1,364,688,000 grid points. **Fig. 3** shows the full global1080 domain. Note that the domain is composed of five "faces" of sizes 1080x(1080x3), 1080x(1080x3), 1080x1080, (1080x3)*1080, (1080*3)*1080, such that when re-assembled back into a rectangle the size is 1080 x (1080x13). The decomposition of this domain is similar to that for the **ASTE1080**, and **Table. 2** shows the number of CPUs required for various sizes of tiles for **global1080**.

The number of nodes and syntax for requesting resources (last 3 columns of **Table. 2**) are calculated as follows. Assume X is the number of cpus per node (X=48 on stampede2, X=56 on frontera), we get, for the example of 120x120x757:

```
on stampede2:
757/X = 15.77
16*X = 768 = total number of nodes (16) and cpus (768) requested
768-757 = 11 = offset in the first node to allow for more memory on cpu 0

on frontera:
757/X = 13.52
14*X = 784 = total number of nodes (14) and cpus (784) requested
784-757= 27 = offset in the first node to allow for more memory on cpu 0
```

		No. of tiles	Frontera(minimum)			Frontera at 48CPU/node		
sNx	sNy	= No. of CPUs (nProc)	#node	#cpus asked	offset	#node	#cpus asked	offset
120	120	757	14	784	27	16	896	11
90	90	1,300	24	1344	44	28	1568	44
90	72	1,597	29	1624	27	34	1904	35
90	60	1,900	35	1925	25	40	2240	20
72	72	1,979	36	2016	37	42	2352	37
72	60	2,349	43	2365	27	49	2744	3
60	60	2,787	50	2800	21	59	3304	45
60	45	3,667	66	3696	29	77	4312	29
45	45	4,831	87	4872	41	101	5656	17
45	36	5,982	107	5992	10	125	7000	18
36	36	7,418	133	7448	30	155	8680	22
36	30	8,837	158	8848	11	185	10360	43
30	30	10,540	189	10584	44	220	12320	20
20	30	15,625	280	15680	55	326	18256	23
20	20	23,135	414	23184	49	482	26992	1
20	18	25,631	458	25648	17	534	29904	1
18	18	28,407	508	28448	41	592	33152	9
18	15	33,967	607	33992	25	708	39648	17
15	15	40,570	725	40600	30	846	47376	38
12	15	50,477	902	50512	35	1052	58912	19
12	12	62,818	1122	62832	14	1309	73304	14
12	9	83,321	1488	83328	7	1736	97216	7
9	9	110,551	1975	110600	49	2304	129024	41
9	6	164,943	2946	164976	33	3437	192472	33
6	6	246,058	4394	246064	6	5127	287112	38

Table 2: Decomposition of the **global1080** regional domain into various tile sizes. sN[x,y] are the number of grid points in the [x,y]-dir per tile. It is desirable to keep the tiles close to a square. **NOTE: Toward the bottom of this Table, as the No. of tiles increases, in code/W2_EXCH2_SIZE.h, one needs to ensure the variable W2_maxNbTiles ≥ Number of CPUs required. If not, the model will crash during run-time with the error reporting this problem in run/STDERR.0000.** For Frontera, #node = round_up(#CPUs / 56), #cpus asked = #node * 56, and offset = #cpus asked minus #cpus required. Syntax for jobscript will be: #SBATCH -N #node; #SBATCH -n #cpus asked; ibrun -n #cpus -o offset mitgcmuv . Thus for the example of **120x120x757**, the syntax is #SBATCH -N 14; #SBATCH -n 784; ibrun -n 757 -o 27 mitgcmuv .

A= CPUs/node desired	N=ceil(nProc/A) (No. of nodes requested)	o=N*A-nProc Offset in the 1st node	No. CPUs requested = N*56	Frontera syntax
56	14	27	784	-N 14 -n 784 -o 27
52	15	23	840	-N 15 -n 840 -o 23
48	16	11	896	-N 16 -n 896 -o 11
44	18	35	1008	-N 18 -n 1008 -o 35
40	19	3	1064	-N 19 -n 1064 -o 3
36	22	35	1232	-N 22 -n 1232 -o 35
32	24	11	1344	-N 24 -n 1344 -o 11
28	28	27	1568	-N 28 -n 1568 -o 27
24	32	11	1792	-N 32 -n 1792 -o 11

Table 3: Breakdown of what to ask for -N, -n, and -o for **frontera** with 56 cpus / node, assuming we're using the tile decomposition **sNx by sNy = 120x120** and requiring **nProc=757** (first row in Table. 2).

2.2 Location of files

For the **global1080**, running can be either on frontera or stampede2. As both machines can see **/work/** , the input binaries for the configuration are stored there. For that reason, we need to copy them, about 170GB total, to **/scratch** (on stampede2) or **/scratch1** (on frontera) prior to copying the **code** and **NAMELISTS**:

```
export atnwork='/work/03901/atnguyen/'
export LOCALDIR='/scratch/03901/atnguyen/global_llc1080/'
cd $LOCALDIR
mkdir jra55 jra55_do
cp -p $atnwork/jra55/*_2010 ./jra55
cp -p $atnwork/jra55/*_2011 ./jra55
cp -p $atnwork/jra55_do/*_2010 ./jra55_do
cp -p $atnwork/jra55_do/*_2011 ./jra55_do
cp -rp $atnwork/llc1080/global/run_template ./
```

Next we proceed with copying the **code**, **NAMELISTS**, optfiles, jobscript:

```
cd $LOCALDIR
cp -rp $atnwork/MITgcm_c67/MITgcm/ ./MITgcm
cp -rp $atnwork/MITgcm_c67/mysetups/global_llc1080/code_tides_mds_prof_sp1 ./code
cp -rp $atnwork/MITgcm_c67/mysetups/global_llc1080/input_tides ./NAMELISTS
cp -p $atnwork/MITgcm_c67/mysetups/global/glob1080_*.bash ./
cp -p  $atnwork/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_stampede2_aste ./
cp -p  $atnwork/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_frontera_aste1 ./

cp -p /home1/03901/atnguyen/.bashrc_stampede2 ./bashrc_stampede2
OR
cp -p /home1/03901/atnguyen/.bashrc_frontera ./bashrc_frontera
```

2.2.1 Compiling:

The compilation of the code requires the main-branch **MITgcm** directory and the specific **code** for the glob1080 domain, as well as the appropriate optfile depending on whether we're using stampede2 or frontera,

```
linux_amd64_ifort+mpi_stampede2_aste
linux_amd64_ifort+mpi_frontera_aste1
```

Before compiling, we need to choose the tile decomposition. Read off **Table. 2** and pick the row you want, for example, **120x120x757**. The first manual step is to cp the **SIZE.h** file associated with this tile set in **code/**:

```
cd $LOCALDIR/code
cp SIZE_h_120x120x757 SIZE.h
```

Next, check to make sure in code/W2_EXCH2_SIZE.h the variable W2_maxNbTiles $\geq$ Number of CPUs required.

```
cd $LOCALDIR/code
grep -n 'W2_maxNbTiles = ' W2_EXCH2_SIZE.h

  35:C      W2_maxNbTiles = Nb of active tiles (=nSx*nSy*nPx*nPy) + Max_Nb_BlankTiles
  43:       PARAMETER ( W2_maxNbTiles = nSx*nSy*nPx*nPy * 2 + 1200 )
```

For **120x120x757**, we must ensure **W2_maxNbTiles = nSx*nSy*nPx*nPy * 2 + 1200 $\geq$ 757**. If not, the model will crash during run-time with the error reporting this problem in **run/STDERR.0000**.

Next, create the **build** directory and compile the MITgcm (note choose correct optfile depending on the machine, stampede2 in this example):

```
cd $LOCALDIR
mkdir build
cd build
../MITgcm/tools/genmake2 -mpi -of=../linux_amd64_ifort+mpi_stampede2_aste -mods=../code -rd=../MITgcm
make depend
make -j 4
```

where the **"genmake2"** command creates the file **Makefile** in the **build/** directory which allows you to check all the module lists and compare to mine, and the last "make" command creates a **"mitgcmuv"** executable and the "-j 4" will compile in parallel to speed things up, using 4 CPUs in this example. One last step I do is to move this executable to match the chosen tile decomposition:

```
mv mitgcmuv mitgcmuv_120x120x757
```

NOTE: if you have to **re-compile**, in the same **build** dir, there are two levels of clearing you need:
a) If all modules are correct, and you're only changing **SIZE.h** to recompile, then the file **Makefile** can be re-used as follows:

```
cd $LOCALDIR/build
make CLEAN
make makefile (note small case ``m'' in ``makefile'')
make depend
make -j 4
```

2.2.2 Prepare and submit jobscript:

Next, edit the jobscript file **glob1080_stampede2.bash** (or **glob1080_fr1.bash** if running on frontera). Line 7 to choose the account:

```
7 #SBATCH -A atn-startup
```

Choose the node type, **skx-normal** for stampede2, and **normal** for frontera, and how long the job will run:

```
6 #SBATCH -t 8:00:00
```

Next, un-comment the correct option associated with the choice **120x120x757** in this example:

```
15 #SBATCH -N 16
16 #SBATCH -n 768

35 nprocs=757
36 snx=120
37 sny=120
```

Finally, edit the line to point to where your $LOCALDIR is:

```
51 scratchdir=/scratch/03901/atnguyen/global_llc1080
```

Now submit the job (example for stampede2):

```
sbatch glob1080_stampede2.bash
```

I've a successfully created all of the above steps and finished a short (4-hour wall time) example run using **120x120x757** at:

```
/scratch/03901/atnguyen/global_llc1080/
  jra55/
  jra55_do/
  run_template/
  linux_amd64_ifort+mpi_frontera_aste1
  linux_amd64_ifort+mpi_stampede2_aste
  MITgcm/
  code/
  build/
  NAMELISTS/
  glob1080_stampede2.bash
  glob1080_fr1.bash
  run_hourly_pk0001577880/
```

3 ASTE4320

3.1 Domain and decomposition

To resolve the high frequency processes in the Arctic, we need a configuration based on the llc4320 grid (e.g., ∼1km in the Arctic). The **ASTE4320** domain is configured to include the entire North Atlantic Ocean above ∼5°N (**Fig. 4**). The decomposition of this domain is shown in **Table. 4**. This is a configuration that **has never been tested** due to lack of resources, both the computation as well as storage. Specifically, to spin up a configuration such as this would require access to enough nodes at efficient time to start from scratch with minimum time-step, the ramp up to a stable configuration. A typical start-up and spin-up will require at least 10-15 runs to address all instabilities/issues.

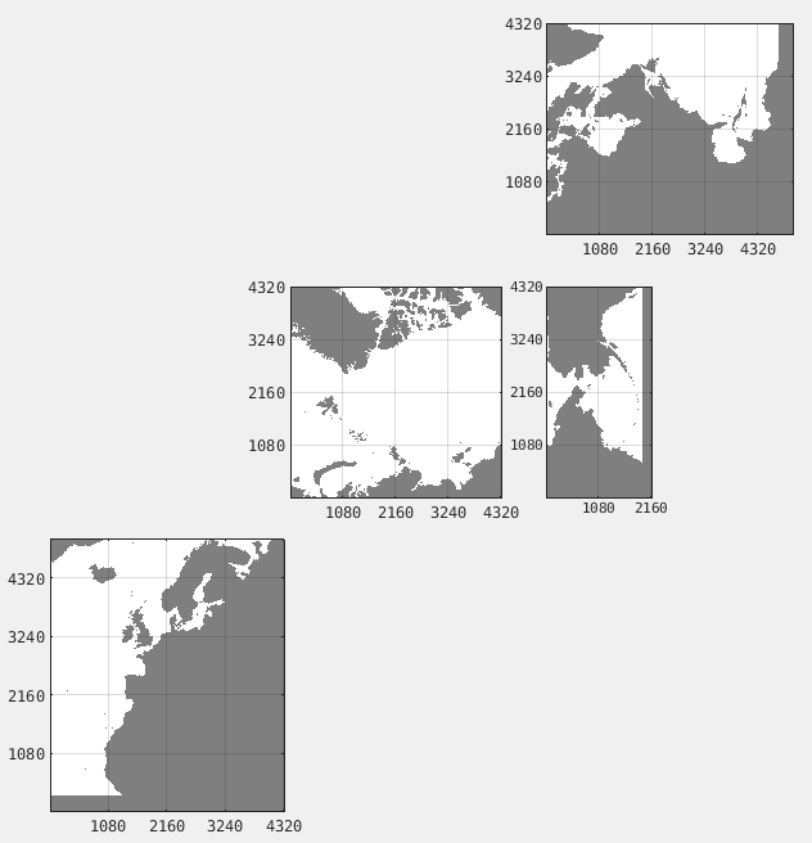

Figure 4: ASTE4320 domain

sNx	sNy	No. of tiles = No. of CPUs (nProc)	Frontera #node	Frontera #cpus asked	Frontera offset
120	120	2812	51	2856	44
90	90	4810	86	4816	6
90	60	7075	127	7112	37
60	60	10428	187	10472	44
45	60	13746	246	13776	30
45	45	18139	324	18144	5
36	36	27874	498	27888	14
36	30	33251	594	33264	13
30	30	39660	709	39704	44
30	24	49345	882	49392	47
24	24	61426	1097	61432	6
20	24	73276	1309	73304	28
20	20	87541	1564	87584	43
18	20	97044	1733	97048	4
18	18	107651	1923	107688	37
18	15	128655	2298	128688	33
15	15	153825	2747	153832	7
12	15	191600	3422	191632	32
12	12	238624	4262	238672	48

Table 4: Decomposition of the **ASTE4320** regional domain into various tile sizes. sN[x,y] are the number of grid points in the [x,y]-dir per tile. It is desirable to keep the tiles close to a square. **NOTE: Toward the bottom of this Table, as the No. of tiles increases, in code/W2_EXCH2_SIZE.h, one needs to ensure the variable W2_maxNbTiles $\geq$ Number of CPUs required. If not, the model will crash during run-time with the error reporting this problem in run/STDERR.0000.** For Frontera, #node = round_up(#CPUs / 56), #cpus asked = #node * 56, and offset = #cpus asked minus #cpus required. Syntax for jobscript will be: #SBATCH -N #node; #SBATCH -n #cpus asked; ibrun -n #cpus -o offset mitgcmuv . Thus for the example of **120x120x2812**, the syntax is #SBATCH -N 51; #SBATCH -n 2856; ibrun -n 2812 -o 44 mitgcmuv .

4 ASTE270

4.1 Domain and decomposition

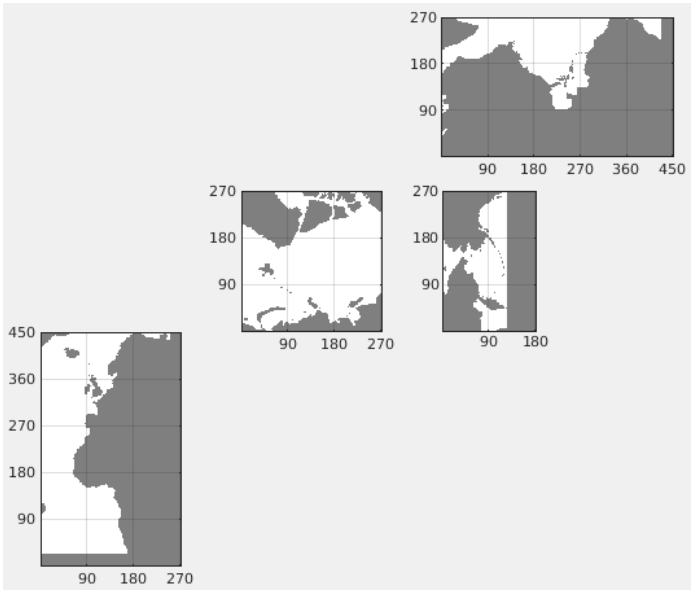

Figure 5: ASTE270 domain

sNx	sNy	No. of tiles = No. of CPUs (nProc)
90	90	36
45	45	117
30	30	242
18	18	580

Table 5: Decomposition of the ASTE270 regional domain into various tile sizes. sN[x,y] are the number of grid points in the [x,y]-dir per tile.

4.2 Location of files

```
$/work/03901/atnguyen/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_stampede2_aste
$/work/03901/atnguyen/MITgcm_c65q/MITgcm/

export astedir='/work/03901/atnguyen/MITgcm_c65q/mysetups/aste_270x450x180/'
$astedir/run_RERUN_ASTE_R1_lfs_adv30_mombudg10day/
   code_ASTE_it62/
   code_adv7_bypass_tamc/
   code_HP_mombudg/
   NAMELISTS/
   script_ASTE_R1_mombudg.bash
```

```
/scratch/projects/ecco/
     llc270/aste_270x450x180/run_template/
     forcing/jra55
```

4.2.1 Compiling:

```
export LOCALDIR='/scratch/03901/atnguyen/aste_270x450x180/mombudg/'
mkdir $LOCALDIR
cd $LOCALDIR
cp -p $/work/03901/atnguyen/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_stampede2_aste ./
cp -rp /work/03901/atnguyen/MITgcm_c65q/MITgcm ./
cp -rp $astedir/code_ASTE_it62 ./
cp -rp $astedir/code_adv7_bypass_tamc ./
cp -rp $astedir/code_HP_mombudg ./
```

Refer to **Table. 5** to choose tile size. Here we use **90x90x36** as an example:

```
cd $LOCALDIR
cd code_ASTE_i62
cp SIZE_h_90x90x36_ol4 SIZE.h

cd $LOCALDIR
mkdir build
cd build

../MITgcm/tools/genmake2 -of=../linux_amd64_ifort+mpi_stampede2_aste \
'-mods=../code_ASTE_it62 ../code_adv7_bypass_tamc ../code_HP_mombudg' -rd=../MITgcm
make depend
make -j 4

mv mitgcmuv mitgcmuv_90x90x36
```

4.2.2 Prepare and submit jobscript:

```
cd $LOCALDIR
cp -rp $astedir/NAMELISTS ./
cp -p /work/03901/atnguyen/MITgcm_c65q/mysetups/aste_270x450x180/script_ASTE_R1_mombudg.bash ./
```

Edit **NAMELISTS/data** to comment out advection scheme 7 and choose advection scheme 30 (**make sure the hash sign # is at the first column**):

```
 38 ## code_adv7_bypass_tamc and set overlap ol[x,y]=8 in SIZE.h
 39 # tempAdvScheme=7,
 40 # saltAdvScheme=7,
 41 #
 42 ## if use adv30, only need overlap ol[x,y]=4
 43  tempAdvScheme=30,
 44  saltAdvScheme=30,
```

Edit **NAMELISTS/data.pkg** to turn off ecco and profiles to reduce computational cost as follows:

```
 10  useECCO        = .FALSE.,
 12  useProfiles    = .FALSE.,
```

Edit jobscript **script_ASTE_R1_mombudg.bash**. For the choice **90x90x36** the 36 CPUs fit within one node, so we can use the skx-dev node to test:

```
  6 #SBATCH -p skx-dev
  7 #SBATCH -t 1:00:00

 13 #SBATCH -N 1
 14 #SBATCH -n 48

 47 #SBATCH -A atn-startup

 63 nprocs=36
 64 snx=90
 65 sny=90

103   criosdir=/scratch/projects/ecco/llc270/aste_270x450x180/
104   scratchdir=/scratch/03901/atnguyen/aste_270x450x180/mombudg/
```

Now submit the job:

```
sbatch script_ASTE_R1_mombudg.bash
```

I've a successfully created all of the above steps and finished a very short example run using **90x90x36** at:

```
/scratch/03901/atnguyen/aste_270x450x180/mombudg/
  linux_amd64_ifort+mpi_stampede2_aste
  MITgcm/
  code_ASTE_it12/
  code_adv7_bypass_tamc/
  code_HP_mombudg/
  build/
  NAMELISTS/
  script_ASTE_R1_mombudg.bash
  run_ASTE_R1_mombudg_pk0000000007/  <-- mombudg outputs in diags/
```

5 ASTE90

5.1 Domain and decomposition

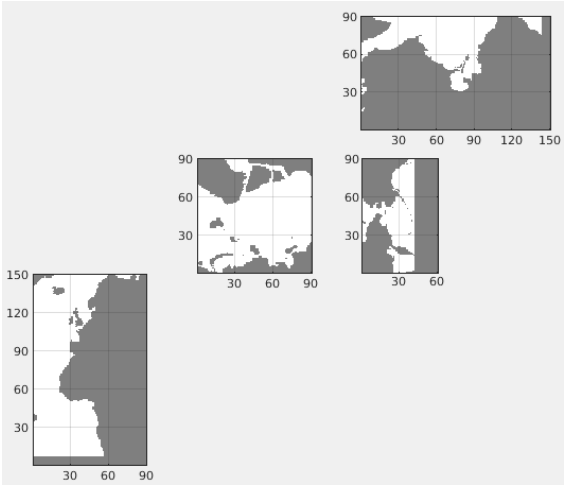

Figure 6: ASTE90 domain

sNx	sNy	No. of tiles = No. of CPUs (nProc)
30	30	36
15	30	64
15	15	117

Table 6: Decomposition of the ASTE90 regional domain into various tile sizes. sN[x,y] are the number of grid points in the [x,y]-dir per tile.

5.2 Location of files

```
/work/03901/atnguyen/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_stampede2_aste
/work/03901/atnguyen/MITgcm_c65q/MITgcm/

/work/03901/atnguyen/MITgcm_c65q/mysetups/aste_90x150x60/
   code_horflux_boxcont_FW5_logdiffkr_mod2_diffkh_fixedlogical_adxOFF
   code_HP_mombudg/
   input_boxcont_fw_beaufort_adxOFF_mombudg
   script_aste90_mombudg.bash

/scratch/projects/ecco/
     llc90_nontelescope/aste_90x150x90/run_template/
     forcing/jra55
     forcing/jra55_it12xx

/scratch/03901/atnguyen/aste_90x150x60/mombudg/
```

5.2.1 Compiling:

Note: Sometimes copy-and-pasting these lines results in an error due to the mistranslation of the apostrophe. Just in case, it is good to write each line out by hand.

```
export astedir='/work/03901/atnguyen/MITgcm_c65q/mysetups/aste_90x150x60/'
export LOCALDIR='/scratch/03901/atnguyen/aste_90x150x60/mombudg/'
mkdir $LOCALDIR
cd $LOCALDIR
cp -p /work/03901/atnguyen/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_stampede2_aste ./
cp -rp /work/03901/atnguyen/MITgcm_c65q/MITgcm ./
cp -rp $astedir/code_horflux_boxcont_FW5_logdiffkr_mod2_diffkh_fixedlogical_adxOFF ./code
cp -rp $astedir/code_HP_mombudg ./

cd $LOCALDIR/code
cp SIZE_h_30x30x36 SIZE.h

cd $LOCALDIR
mkdir build
cd build

../MITgcm/tools/genmake2 -of=../linux_amd64_ifort+mpi_stampede2_aste \
'-mods=../code ../code_HP_mombudg' -rd=../MITgcm
make depend
make -j 4
mv mitgcmuv mitgcmuv_30x30x36
```

5.2.2 Prepare and submit jobscript:

```
cd $LOCALDIR
cp -rp $astedir/input_boxcont_fw_beaufort_adxOFF_mombudg ./NAMELISTS
cp -p  $astedir/script_aste90_mombudg.bash ./
```

Edit **NAMELISTS/data.pkg** to make sure pkg ecco, ctrl, profiles, autodiff, and smooth are off to reduce computational cost as follows:

```
 10  useECCO        = .FALSE.,
 11  useCTRL        = .FALSE.,
 12  useProfiles    = .FALSE.,
 13  useAUTODIFF    = .FALSE.,
 14  useSMOOTH      = .FALSE.,
```

Edit jobscript **script_aste90_mombudg.bash**. For the choice **30x30x36** the 36 CPUs fit within one node, so we can use the skx-dev node to test:

```
 6 #SBATCH -p skx-dev
 7 #SBATCH -t 1:00:00
 9 #SBATCH -N 1
10 #SBATCH -n 36
```

```
      14 #SBATCH -A atn-startup
      30 nprocs=36
      31 snx=30
      32 sny=30
      50   criosdir=/scratch/projects/ecco/llc90_nontelescop/aste_90x150x60/
      51   scratchdir=/scratch/03901/atnguyen/aste_90x150x60/mombudg/
```

Now submit the job:

```
    sbatch script_aste90_mombudg.bash
```

I've a successfully created all of the above steps and finished a very short example run using **30x30x36** at:

```
  /scratch/03901/atnguyen/aste_90x150x60/mombudg/
    linux_amd64_ifort+mpi_stampede2_aste
    MITgcm/
    code/
    code_HP_mombudg/
    build/
    NAMELISTS/
    script_aste90_mombudg.bash
    run_aste90_mombudg_10days/  <-- mombudg outputs in diags/
```

6 EXTRA: LabSea

This domain is a simple small domain of size 20x16 in the horizontal and with 23 vertical levels.

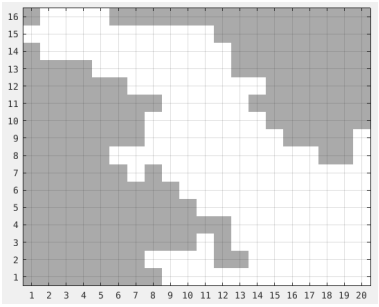

Figure 7: labsea domain. This entire domain fits into a single CPU.

6.1 Location of files

```
/work/03901/atnguyen/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_stampede2_aste
/work/03901/atnguyen/MITgcm_c65q/MITgcm/
/work/03901/atnguyen/MITgcm_c65q/mysetups/labsea/
   code_horflux_boxcont_FW5_logdiffkr_mod2_diffkh_fixedlogical_adxOFF/
   code_HP_mombudg/
   input_boxcont_fw_labsea_logdiffkr_adxOFF_mombudg/
   input_binaries/
   script_labsea_adxOFF_mombudg.bash
/scratch/03901/atnguyen/labsea/mombudg/
```

6.2 Compiling:

```
export atnwork='/work/03901/atnguyen/MITgcm_c65q/mysetups/labsea/'
export LOCALDIR='/scratch/03901/atnguyen/labsea/mombudg/'
cd $LOCALDIR
cp -rp /work/03901/atnguyen/MITgcm_c65q/MITgcm ./
cp -rp $atnwork/code_horflux_boxcont_FW5_logdiffkr_mod2_diffkh_fixedlogical_adxOFF ./code
cp -rp $atnwork/code_HP_mombudg/ ./
cp -rp $atnwork/input_boxcont_fw_labsea_logdiffkr_adxOFF_mombudg/ ./NAMELISTS
cp -rp $atnwork/input_binaries/ ./
cp -rp $atnwork/script_labsea_adxOFF_mombudg.bash ./
cp /work/03901/atnguyen/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_stampede2_aste ./
cp /work/03901/atnguyen/BitBucket/computing/optfiles/linux_amd64_ifort+mpi_stampede2_aste_knl ./
```

As we only need 1 CPU, we can choose between knight-landing (**KNL**) and sky-lake (**SKX**) node, where the former (KNL) is much slower than the latter (by a factor of 3 or more!!):

```
Building using skx node:
   mkdir $LOCALDIR/build_skx
   cd $LOCALDIR/build_skx
```

```
   ../MITgcm/tools/genmake2 -mpi -of=../linux_amd64_ifort+mpi_stampede2_aste \
      '-mods=../code ../code_HP_mombudg' -rd=../MITgcm
   make depend
   make -j 4

Building using knl node:
   ../MITgcm/tools/genmake2 -mpi -of=../linux_amd64_ifort+mpi_stampede2_aste_knl \
      '-mods=../code ../code_HP_mombudg' -rd=../MITgcm
   make depend
   make -j 4
```

Edit the jobscript **script_labsea_adxOFF_mombudg.bash**, where the development queue for KNL is **"development"** and for SKX **"skx-dev"**. Below is the example for KNL node:

```
     6 #SBATCH -p development
     9 #SBATCH -t 1:00:00
    10 #SBATCH -N 1
    11 #SBATCH -n 1
    12 #SBATCH -A atn-startup

    44 scratchdir=/scratch/03901/atnguyen/labsea/mombudg
    34 node="knl"
    47 builddir=$scratchdir/build_${node}
    49 workdir=$scratchdir/run_mombudg_${node}
```

Save and submit the job:

```
    sbatch script_labsea_adxOFF_mombudg.bash
```

There are two runs provided, only 10-time-steps, using knl and skx:

```
    /scratch/03901/atnguyen/labsea/mombudg/
    run_mombudg_knl
    run_mombudg_skx
```

A comparison of time can be seen to compare the time difference between knl and skx:

```
login1.stampede2(2085)$ more run_mombudg_knl/run.MITGCM.timing
    Mon Jun  7 01:09:17 CDT 2021
    Mon Jun  7 01:10:01 CDT 2021 <-- 44sec
login1.stampede2(2086)$ more run_mombudg_skx/run.MITGCM.timing
    Mon Jun  7 01:07:34 CDT 2021
    Mon Jun  7 01:07:50 CDT 2021 <-- 16sec!

    44/16=2.75
```

The longer we run, the potentially slower the run using knl nodes can become. However, knl nodes are readily available , unlike skx which is very limited.

6.2.1 Expected results:

```
/scratch/03901/atnguyen/labsea/mombudg/
run_mombudg_knl/diags/
run_mombudg_skx/diags/
```

6.2.2 Some tacc tips

How to keep a variable between login sessions: The best way is to create a shortcut in /.bashrc_stampede2

```
export $LOCALHOST='/scratch/07117/ni2kita/labsea/mombudg/'
save it, then source

source ~/.bashrc
```

Note about running on scratch and what to copy to work:

```
cd $LOCALDIR
tar czvf build_skx.tgz build_skx/ <-- "c" create, "z" zip, "v" verbose, "f" file
tar czvf build_knl.tgz build_knl/

mkdir $workdir/labsea/
mkdir $workdir/labsea/mombudg
cp build_skx.tgz $workdir/labsea/mombudg/
cp build_knl.tgz $workdir/labsea/mombudg/

cd $workdir/labsea/mombudg/
tar xzvf build_skx.tgz build_skx/mitgcmuv
tar xzvf build_skx.tgz build_skx/Makefile

tar xzvf build_knl.tgz build_knl/mitgcmuv
tar xzvf build_knl.tgz build_knl/Makefile
```