Bosun Documentation

Release 1

Luiz Irber, Guilherme Castelão, Léo Siqueira

July 23, 2015

Contents

Bosun is a runtime environment for BESM and associated models. It is used to document and automate the tasks related to running a climate model (prepare inputs, compile the code, run the model, archive the results). It is based on Fabric and can be used as a library or CLI.

Contents:

CHAPTER 1

Design principles

This section explains the philosophy behind Bosun development and how we can fit a model to it.

1.1 Concepts

Bosun tasks acts over data sources and sinks. Tasks can be divided in four categories: Prepare, Compile, Run and Archive. Data sources and sinks can be Experiments, Code, Artifacts and Storage.

1.1.1 Execution

In a very high level executing a model can be divided in four parts:

- Prepare

  This step can include data manipulation (download, munging, copying) and directory creation.

- Compile

  Code checkout from repositories, instrumentation and compilation are good fits for this step.

- Run

  How to run the model: batch system submission, consistency checks, status and automatic restarts.

- Archive

1.1.2 Sources and sinks

- Experiment
- Code
- Artifacts
- Storage

1.2 Small tasks, composable tasks

Tasks should be small, and complex tasks should be broken in smaller ones.

CHAPTER 2

Bosun Tutorial

2.1 Introduction

In this manual $EXP_REPOS is the local directory where you want to keep the model experiments (local repository) in your local machine. To name it exp_repos and place in your $HOME dir:

```
$ EXP_REPOS=~/exp_repos
```

2.2 Installing

First you need to install Mercurial, a version control system, if not previously installed. As first step, you should teach Mercurial your name (required). For that you open/create the file ~/.hgrc in your home directory with a text-editor and add the ui section (user interaction) with your username:

```
[ui]
username = name <user@mail.com>
```

In Mercurial, we do all of our work inside a repository which will be $EXP_REPOS set up in the Introduction. A repository is a directory that contains all of the source files that we want to keep history of, along with complete histories of those source files. To do this, we will use the clone command (hg clone). This makes a clone of a repository (remote in our case); it makes a complete copy of another repository so that we will have our own local, private one to work in. We will bring the experiments repository to your local machine after the installation is finished.

2.2.1 Installing dependencies and Bosun

We need to install a series of software dependencies for Bosun to work, although it already has built-in dependencies in itself that are installed automatically.

First, we install virtualenv which is a tool to create isolated Python environments. It creates a Python interpreter that uses an isolated environment in order to not create conflicts among different versions or libraries installed. Anytime you use that interpreter the libraries in that environment will be used. To install it:

```
$ curl -O https://raw.github.com/pypa/virtualenv/master/virtualenv.py
$ python virtualenv.py ~/.bosun_env
```

In order to use this recently created environment it is necessary to activate it. To do that we execute the following command:

```
$ source ~/.bosun_env/bin/activate
```

This command will change your $PATH so its first entry is the virtualenv's bin/ directory and your prompt is changed so that you can visually distinguish when this environment is activated (you have to use source because it changes your shell environment in-place.). The activate command only alters the $PATH and prompt of you working terminal, if you want to use Bosun in another terminal you will have to activate it in that new terminal too. If you want to undo the changes done with activate you execute the following command:

```
(environment) $ deactivate
```

Finally, install another dependency and Bosun:

```
$ pip install bosun
$ rm virtualenv.py
```

2.2.2 Preparing Bosun for first use

Before running Bosun for the first time, please configure your username, remote host and connection port, so that you don't need to type it every time you use it. This configuration is placed inside the file ~/.bosunrc (.bosunrc in your $HOME directory) and needs to be edited/created.

Ex:1) If you are inside CPTEC/INPE:

```
user: your_user_at_tupa
hosts: tupa.cptec.inpe.br
```

2.

```
user: your_user_at_tupa
hosts: lmo-f
port: 6000
```

2.3 Updating

In order to update Bosun without installing all the built-in dependencies again:

```
$ pip install -U --no-deps bosun==dev
```

2.4 Downloading experiments repository to your local machine

Bring the experiments repository to your local machine using:

```
$ hg clone ssh://tupa//stornext/online2/ocean/exp_repos $EXP_REPOS
```

When you clone a repository, the new repository becomes an exact copy of the existing one at the time of the clone, but subsequent changes in either one will not show up in the other unless you explicitly transfer them, by either pulling (hg pull -u) or pushing (hg push) using Mercurial.

2.5 Creating a new experiment

In order to create a new experiment, copy the directory expbase to a new exp dir, e.g. expnew:

```
$ cd $EXP_REPOS/exp
$ cp -R expbase expnew
$ cd expnew
```

The new experiment directory will have the following structure: 1. MODELIN 1. AGCM model configuration 1. input.nml 1. OGCM and coupler configuration. 1. instrument_coupler.apa 1. Instrumentation configuration. 1. namelist.yaml Configurations for running the experiment.

1. runscripts/ 1. Files for compiling, executing, and post-processing. You should not need to modify these. 1. tables/ 1. field_table 2. diag_table 3. data_table 1. 1. data_override

Go on and edit namelist.yaml which contains the experiment name, start/end dates, etc:

```
$ vi namelist.yaml
```

Change name to expnew and save.

```
$ cd tables
```

Edit diag_table for selecting diagnostic variables for the OGCM model outputs:

```
$ vi diag_table
$ cd $EXP_REPOS/exp
```

Update new exp in the repository:

```
$ hg add expnew
```

Check for changes:

```
$ hg pull -u
```

Merge if necessary:

```
$ hg merge
```

The act of creating a changeset in the main repository is called committing it. We perform a commit using the commit command. The hg commit command has a nice short alias: ci (“check in”), so we can use that:

```
$ hg ci
```

This drops us into an editor, and presents us with a few cryptic lines of text. The first line is empty and the lines that follow identify the user, branch name and the files that will go into this changeset. Edit the text in the first line briefly mentioning your changes so it can be understood by yourself and other users of the same repository/branch.

Upload your changes to the main repository (remote) using:

```
$ hg push
```

2.6 Running Tasks

Before running a task with Bosun, please check if your virtualenv is activated:

```
$ source ~/.bosun_env/bin/activate
```

Bosun has several tasks that can be performed. To list the available tasks to be performed:

```
$ bosun --list
```

To detail the task dependencies and which variables need to be set in the namelists respective to a specific task from the list:

```
$ bosun -d taskname
```

2.6.1 Run Experiment from scratch

In order to run a full model cycle, i.e. prepare (create all directories, and copy the necessary inputs to the right place), compile and run: $ bosun deploy_and_run:name=expnew The three tasks performed in deploy_and_run can also be achieved if the user runs the tasks: prepare, compilation, and run in this specific order. This will create directories in: /scratchin/grupos/ocean/home/${USER}/${EXP}/ containing source files for compilation, namelists, and executables for this specific experiment. It will also create directories in: /scratchout/grupos/ocean/home/${USER}/${EXP}/${TYPE}/ where ${TYPE} is the type of run: ocean model only (mom4p1_falsecoupled) or coupled ocean/atmosphere run (coupled). It contains the input, outputs, and restart files. If running a coupled run, the atmospheric outputs are located in: /scratchout/grupos/ocean/home/${USER}/${EXP}/coupled/model/dataout/TQ0062L028/

2.6.2 Restart an experiment

Assuming you've successfully ran the model, you now have the model restart files written to your RESTART directory. For safety, rename the restart directory to save the data intact, before starting a new run: $ cp RESTART RESTART_yyyymmddhh In order to restart and existing experiment: $ bosun restart:name=expnew,restart=yyyymmddhh,finish=yyyymmddhh This task will check the current model time in /RESTART/coupler.res against the given date and return an error message if they do not match. Observation: If restarting again, don't forget to rename the newly formed RESTART directory if you plan to keep those files!

2.6.3 Creating new grids

The generation of new grids is treated in the same way as a regular experiment in Bosun. However, it does require some editing of the grid specification data file (to be moved inside the namelist in the future):

```
$ cd $EXP_REPOS/exp
$ cp -R expbase expnewgrid
$ cd expnewgrid
$ vi runscripts/mom4_pre/ocean_grid_run.csh
```

Usually you will need to edit hgrid_nml, vgrid_nml and topog_nml inside this file to include your grid specifications and suitable topography file for the intended resolution before running.

Once you configured your new grid, you need to commit the changes and send to the remote machine:

```
$ hg add expnewgrid
$ hg pull -u
$ hg commit -m "New grid generation"
$ hg push
```

In order to create the grid and the exchange grid for the coupled model using Bosun:

```
$ bosun generate_grid:name="expnewgrid"
$ bosun make_xgrids:name="expnewgrid"
```

This tasks will generate the following file in the remote machine:

```
/scratchout/grupos/ocean/home/${USER}/${EXP}/coupled/gengrid/grid_spec_UNION.nc
```

The grid_spec_UNION.nc is the actual file that will be used when running the model. It is recommended to change the name of the file so that it includes the grid resolution, e.g. grid_spec_0.1.nc for a 1/10 of degree global regular grid.

2.6.4 Regridding required fields for new grids

The generation of regrid fields is also treated in the same way as a regular experiment in Bosun. Make sure that you have the destination grid available if it wasn't created using the previous section. Go on and edit namelist.yaml inside expnewgrid:

```
$ vi namelist.yaml
```

```
regrid_3d_src_file: /stornext/online2/ocean/database/your_source_file.nc
regrid_3d_dest_grid: ${gengrid_workdir}/your_destination_grid.nc
regrid_3d_output_filename: your_3D_field_regridded.nc
```

Make sure "regrid_3d_run_this_module" is set to "True" in order to run the regrid 3D module, then edit the above three lines in order to set the source file, e.g. with temperature and salt 3D fields (monthly climatology with 12 time steps), your model grid file, and the output file name.

Make sure the number of variables and their names in runscripts/mom4_pre/regrid_3d_run.csh match the names of the source fields in regrid_3d_src_file:

```
$ vi runscripts/mom4_pre/regrid_3d_run.csh
```

```
numfields = 2
src_field_name = 'temp','salt'
```

Once the above is done, run the task:

```
$ bosun regrid_3d:name="expnewgrid"
```

This will create the following regrid file in the directory:

```
/scratchout/grupos/ocean/home/${USER}/${EXP}/coupled/regrid_3d/your_3D_field_regridded.nc
```

Always check if the output file really contains the required fields with the correct numbers of grid points and time steps before running the model.

To start a run using a new grid, it is also required to have three other 2D fields regridded: temp, salt, and chlorophyll. Go on and edit namelist.yaml inside expnewgrid:

```
$ vi namelist.yaml
```

```
regrid_2d_namelist:
  file: ${expdir}/tables/regrid_2d.nml
  vars:
    regrid_2d_nml:
      numfields: 1
      src_file: /stornext/online2/ocean/database/levitus.nc
      src_field_name: temp
      dest_field_name: temp
      dest_grid: ${gengrid_workdir}/your_destination_grid.nc
      dest_file: temp_0.1regrid.nc
      dest_grid_type: T
      vector_field: False
```

Make sure "regrid_2d_run_this_module" is set to "True" in order to run the regrid 2D module, then edit the regrid_2d_namelist shown in the lines above. You'll need to edit this regrid_2d_namelist for each field and create individual files for temp, salt, and chlorophyll.

This will create the following regrid file in the directory:

```
/scratchout/grupos/ocean/home/${USER}/${EXP}/coupled/regrid_2d/your_2D_field_regridded.nc
```

CHAPTER 3

Developer tips

3.1 The env_options decorator

Why env_options is a decorator?

CHAPTER 4

Indices and tables

- genindex
- modindex
- search