WIM Description.

T.D. Williams
timothy.williams@nersc.no

Nansen Environmental and Remote Sensing Center
Thormøhlensgate 47
5006 Bergen
Norway

April 29, 2015

Contents

Chapter 1

Introduction

Text.

Chapter 2

Mathematical and physical background

Text.

2.1 Scattering model

Text.

2.1.1 Two-dimensional model: a single floe

Text. ?

Chapter 3

Cron and Crontab

In this chapter, after a brief introduction on *Cron*, the user will learn how to check and modify a *Crontab* file.

3.1 Introduction

Cron is a software utility time-based for **Unix-like** operating systems. *Crontab* is a text file that guides *Cron* providing the time and nature of its operations; it is possible for every user to have a personal *Crontab*. Every **minute** *Cron* checks the uploaded *Cronfile* comparing it to the **system** time. These tasks will run regardless of whether the user is actually logged into the system.

3.2 Crontab editing

The layout of the *Crontab* file (fig. 3.1) provides 5 inputs for time and 1 for the command:

```
# * * * * *  command to execute
# │ │ │ │ │
# │ │ │ │ │
# │ │ │ │ └──── day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the same as 0)
# │ │ │ └────── month (1 - 12)
# │ │ └──────── day of month (1 - 31)
# │ └────────── hour (0 - 23)
# └──────────── min (0 - 59)
```

Figure 3.1: *Crontab* file's layout with explanation

```
# This is an example of a Crontab file
* * * * * /home/user/prog1.sh    # Every minute
15 * * * * /home/user/prog2.sh   # Every hour at min 15
# Comments don't interfere with the tab
12 18 * * * /home/user/prog3.sh # Every day at 18:12
*/5 * * 1 * /home/user/prog4.sh # Every 5 minutes of January
```

NOTE - *\ will reiterate that command

To modify the *Crontab* a previously written file can be uploaded (*i.e. mycrontab*) or it can be manually modified (this will change the current user's).

```
$ crontab mycrontab      # Uploading a personal Crontab
$ crontab -e             # Manually modifying the Crontab
```

3.3 Crontab on Hexagon

Cron is working on **Hexagon** as well but since is a very busy and heavily stressed system there are a few precautions that should be remembered before setting up a *Cron* job.

First of all you should choose a *node* in which set up your *Crontab*. A *node* is a connection, redistribution or communication point for a network. For more information please visit (`http://docs.hpc.uib.no/wiki/Main_Page`).
Another precaution is to use unique *Crontab* file's names so that different set-ups will not be uploaded by mistake.

Here is shown a complete procedure for a *Crontab* setup in **Hexagon**:

```
$ ssh hexagon            # Personal login on Hexagon

# Now a Cron-job is appended to a unique personal tab
$ echo "* * * * * /home/user/prog.sh" >> prog_cron

$ ssh login1             # Login node1 (change number to change node)
$ crontab prog_cron      # Uploading unique Crontab
```

Now every minute *Cron* will execute *prog.sh* from *node1*.

Remember that the outputs of *Cron* are shown in a virtual display hence are lost if not saved. It is possible to save the outputs by manually inserting a log, saving stdr.out or setting up an automatic e-mail service that will send to a specified address the outputs by adding the following line in the *Crontab* file (**NB** before the jobs):

```
$ MAILTO=user@domain.org
```

Chapter 4

Model's Inputs - last modification April 29, 2015

In this chapter the user will be instructed on how to control the scripts used to download and storage the input files that will be used as initial conditions for the WIM model.
There are two products that are updated and/or stored from every week to every hour:

- TOPAZ - Weekly Restart Files
- WAMNSEA - Daily Waves

4.1 TOPAZ

4.1.1 Introduction

Towards an **O**perational **P**rediction system for the North **A**tlantic European coastal **Z**ones, simply known as TOPAZ is a coupled ocean-sea ice data assimilation system for the North Atlantic Ocean and Arctic. It is the only operational, large-scale ocean data assimilation system that uses the ensemble Kalman filter. This means that TOPAZ features a time-evolving, state-dependent estimate of the state error covariance.

In the regional Barents and Kara Sea forecast system (**BS1**), the TOPAZ (**TP4**) is used as an outer model in the nested system. Locally the TP4 model runs once a week for an 11 days period, with 9 days forecast, and produce initial and boundary conditions for the regional model BS1.

4.1.2 Data Format

The restart product consist of 3 different files that will provide initial conditions:

- **TP4restart *YYYY_ddd_hh*_mem001.a** - [**data**] initial conditions of the system.
- **TP4restart *YYYY_ddd_hh*_mem001.b** - [**text**] information and instructions for the first file.

- **TP4restart *YYYY_ddd_hh*ICE.uf** - [**data**] informations about sea ice (localization, height, etc..).

Where **Y** is the Year, **d** is the day and **h** is the hour of the restarts release.

NB - file names could change depending on the service.

These files are weekly uploaded to the main server of the forecast system Hexagon. They will be temporary stored in a working directory that is cleared every two weeks (or else depending on the needs of the server) hence the need of a script that will daily check and if needed archive and transfer the new data (see 5.1.1).

4.2 WAMNSEA

Chapter 5

Scripts Protocol - April 29, 2015

For the daily forecast product, its storage and online delivery, the algorithm follows 3 main phases:

- **Inputs Download and Archiviation**
- **Model Run**
- **Post-processing and Storing**

NB All the scripts can be found on the *GIT-HUB* directory (https://github.com/nansencenter/SWARP-routines).

The following diagram shows the work directory:

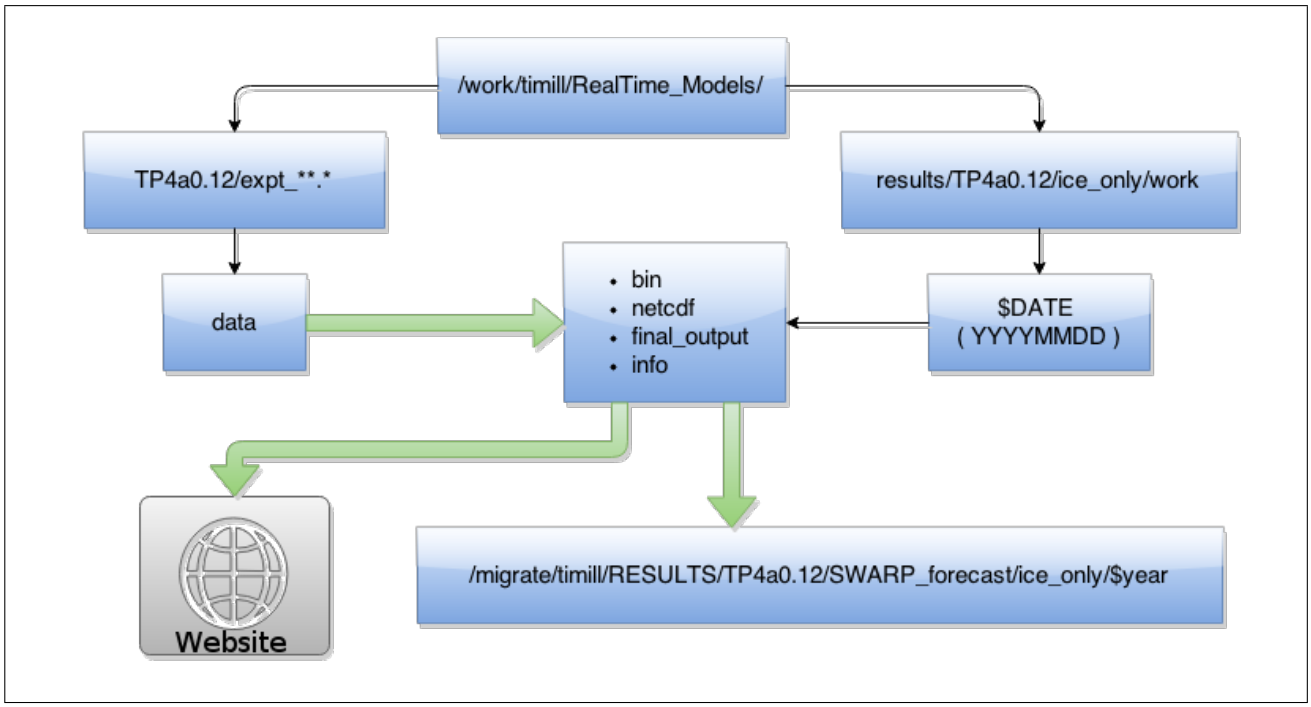

Figure 5.1: ****.*** indicates the experiment ran; **$DATE** is the daily date (*YYYYMMDD*) and the **green** arrows are operations done by the scripts described later in this chapter.

NB It is possible that these directories will be changed and/or modified, please use caution and keep this log up to date if possible.

5.1 Inputs Download and Archiviation

The protocol consists in 2 scripts

- topaz_archive_restart.sh
- wamnsea_update.sh

5.1.1 topaz_archive_restart.sh

dir - forecast_scripts

The purpose of this script is to daily check the presence of new restarts for the Topaz model, collecting and store them. These are binary files produced by the *Nansen Environmental and Remote Sensor Center* weekly and can be found on the super computer **Hexagon**.

The logic behind the script is to find all the files with **TP4restart** in their file-name and check their presence in a list of the stored files. If the file is on the list it has already been archived and the script will end otherwise will automatically add the new files to the list and to the archive (using a *tar* compression protocol). Every operation will write a daily log that will be appended to a report delivered weekly to the forecast supervisor.

This script is regularly executed by **Cron** (see [3]) at 18:00 everyday.

5.1.2 wamnsea_update.sh

dir - forecast_scripts

The goal of this script is to constantly update the wave analysis and forecast for the North and Barents sea. These are Netcdf files produced by *met.no* and uploaded daily on **myocean**.

The script will download every WAM product (both daily analysis and forecasts) present on **myocean** using a *ncftp* protocol. If a product was already downloaded it will be substituted by the new one. The daily products are stored, a copy of them will be merged in a single file for every year.

This script is regularly executed by **Cron** (see [3]) every hour except for a time-out window (00:00/04:00) where it won't download any file because usually the files are uploaded after 03:00. If no new file is downloaded between 04:00 and 08:00 an email alert will be sent to the forecast supervisor.

5.2 run_forecast.sh

dir - forecast_scripts

The script is a collection of 3 scripts:

- **topaz_get_restart.sh**
- **make_infile4forecast.sh**
- **pbsjob.sh**

The last one will also launch the subscript **process_FCresults.sh** described in 5.2.4.

The following diagram shows the script's main structure:

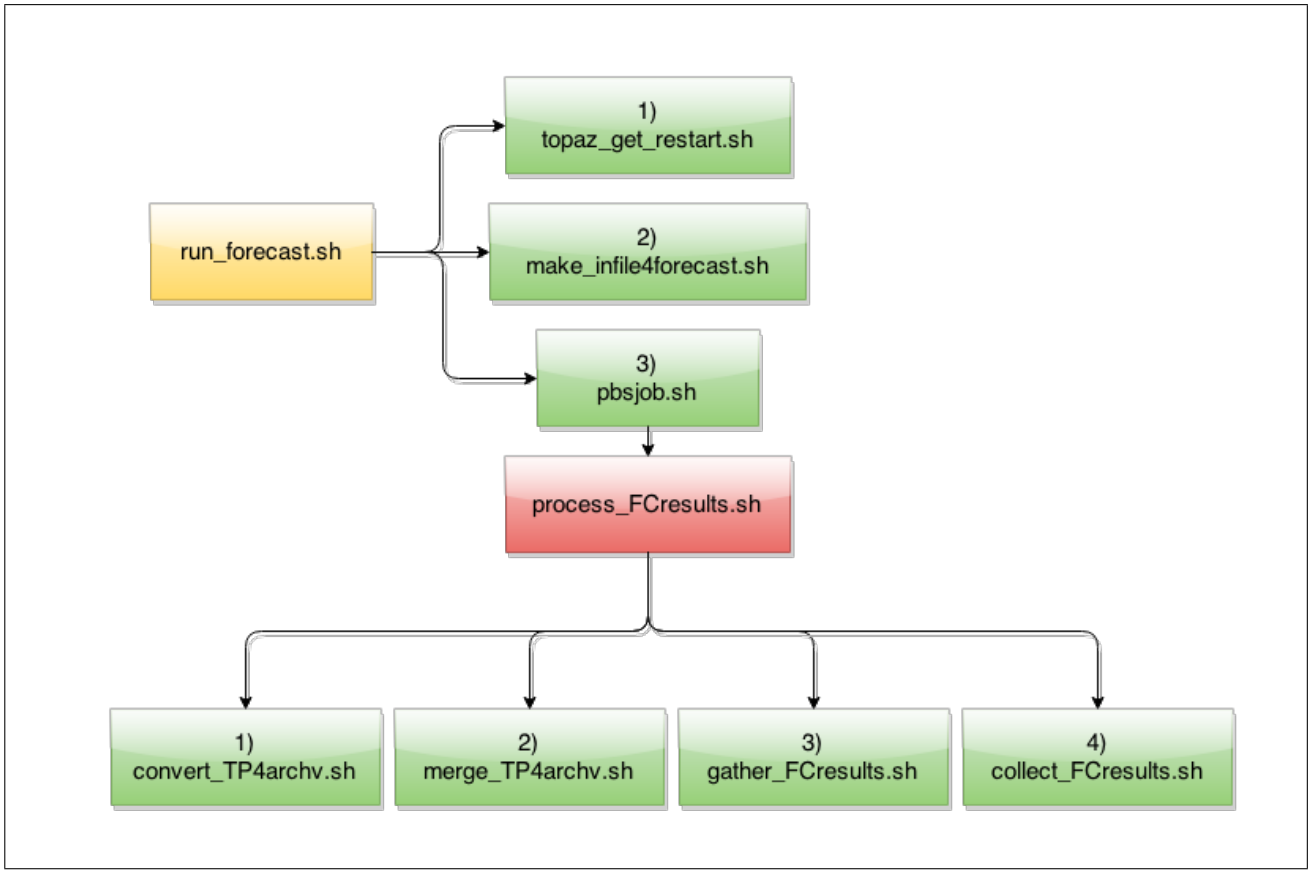

Figure 5.2:

First of all this scripts produces a text file named **datelist.txt** that contains all informations regarding the date of the run. It then launches the subscripts.

This script is regularly executed by **Cron** (see [3]) every day at 0200 - **NB** the first 2 subscripts will be done in a couple of minutes but **pbsjob.sh** will be queued in **Hexagon**. The queue depends on users' requests hence can last from 5 minutes to 5 hours. This means that **process_FCresults.sh** will be executed somewhere between 02:05 and 05:00.

5.2.1 topaz_get_restart.sh

dir - forecast_scripts

The goal of this script is to find, uncompress and relocate the newest restart files.

The logic is very simple; the script read through the archive list and reads which restart has been archived for last. It then checks in the *data* folder (fig.5.1) if the restart files have already been uncompressed if not it will proceed with the operation else it will exit.

If the list is missing, can't be found or if the restarts are older than 13 days an email with a warning will be sent to the forecast supervisor. In addition every Monday a weekly log will be sent.

5.2.2 make_infile4forecast.sh

dir - forecast_scripts

This script will produce a file (**infile.in**) with informations about the *Run Version* of the model, the *Reference year*, the *Starting day* and the *Last day* of the forecast.

A log is adjourned at every operation and it will be cleaned every Monday.

5.2.3 pbsjob.sh

dir - forecast_scripts/inputs/..

This script is the core of the model and will not be covered by this guide. For further information see the user manual for **HYCOM** model or contact Dr. Timothy Williams (timothy.williams@nersc.no).

The output consists of two kind of binary files. A daily mean file (*TP4DAILY**) for every day from the restart day to the final day of the run (running day plus forecast days) and a file every three hour (*TP4archv**) again from the restart day to the final day.

5.2.4 process_FCresults.sh

dir - forecast_scripts & netcdf_production

This scripts calls 4 different subscripts in the following order:

1. **gather_FCresults.sh** - Gathers all the information files and the binary files produced by the model and moves them in the proper **$DATE** directory (fig. 5.1). Email alert whether any file is missing.

2. **convert_TP4archv.sh** - Converts the three hour binary files (*TP4archv**) into *Netcdf* files. Email alert whether files are missing.

3. **merge_TP4archv.sh** - Merges the Netcdf files into the final product. The first binary file is produced for the day of the restart, the product has to start from the day the run was done. No alerts implemented.

4. **collect_FCresults.sh** - Collects and archive the **$DATE** directory, binary files, netcdf files, the final product and related information files. No alerts implemented.

5.3 copy2johansen.sh

dir - forecast_scripts

This script will transfer the final product from the supercomputer **Hexagon** to the *Nansen*'s server **Johansen** using a *secure copy protocol* (or *scp*). The script will look for the most recent final product starting from the present day and going

This script is regularly executed by **Cron** (see [3]) and will send an alert to the forecast supervisor in case the uploaded product is older than 2 days.