View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Scientific Open-access Literature Archive and Repository

CHIMBO-2020

Fabio Massimo Grasso, Tony Christian Landi, Oxana Drofa, Fabrizio Roccato

Maggio 2021

Contents

1 Presentazione **2**

2 Uso di Singularity **4**
2.1 Installazione 4
2.2 Creazione di un container 5
2.2.1 Scrittura della *recipe* 5
2.2.2 Building del container 8
2.2.3 Modifica della sandbox 9

3 Installazione e manutenzione della catena CHIMBO-2020 **11**
3.1 Installazione 11
3.2 Diagrammi operativi delle applicazioni 14

1 Presentazione

Questo *technical report* costituisce una guida per la creazione e manutenzione della catena modellistica CHIMBO-2020 sviluppata da ISAC a partire dal 2017 [2]. Le corse *operative* attualmente attive e generate quotidianamente con la attuale versione di questo sistema modellistico sono presenti ai due links seguenti:

- CHIMBO @ CNR ISAC

- CHIMBO @ ARPA Sicilia

Come riportato nelle pagine web sopra indicate, la catena CHIMBO calcola i campi 3D delle concentrazioni degli aerosols e dei principali gas presenti in atmosfera. Le condizioni iniziali ed al contorno per la parte meteorologica sono fornite gratuitamente della NOAA/NCEP che distribuisce i dati generati dal modello Global Forecastes System (GFS), mentre quelle relative alla parte di composizione atmosferica vengono scaricate gratuitamente dalla ECMWF nell′ ambito della iniziativa internazionale Copernicus. Gli elementi principali che costituiscono la catena sono:

- GFS: dowload dei dati GFS alla risoluzione orizzontale sull'area globale di 0.5 x 0.5 scaricati gratuitamente dall ftp: ftpprd.ncep.noaa.gov .
- IBC: download dei dati CAMS tramite il servizio ftp di ECMWF e loro preparazione del formato per essere ingerito dal Chemical Transport Model (CTM) CHIMERE (https://www.lmd.polytechnique.fr/chimere/)
- PreBOLAM: step propedeutico al run di BOLAM utile per la preparazione dei dati di *input* per il modello BOLAM.
- BOLAM: previsione meteo eseguita con il modello BOLAM sviluppato di ISAC.
- B2C (BOLAM to CHIMERE): Trasformazione degli outputs meteorologici da dare in *input* al modello CHIMERE per la simulazione di composizione atmosferica.
- CHIMERE: simulazione multi scala dei fenomeni di trasporto chimico tramite il modello fotochimico CHIMEREv2014b.

La tecnologia di supporto utilizzata è il programma di containerizzazione Singularity [1]; questo consente di astrarre lo strato sottostante la catena modellistica, vale a dire il sistema operativo e le librerie di supporto, con la finalità di ottenere la replicabilità del setup che a sua volta permette di ottenere:

- facilitazione delle operazioni di **manutenzione** poichè l'ambiente di produzione è praticamente implementato *from scratch (da zero)*, di fatto sono installate solo le librerie strettamente necessarie alla catena e nelle modalità standard del sistema operativo, senza particolari adattamenti;

- **modularità**: questa architettura si presta particolarmente all'implementazione della catena CHIMBO in cui l'interazione dei componenti è prettamente sequenziale, quindi la modifica di un singolo anello non implica necessariamente un aggiustamento dell'intero sistema;
- **sicurezza**: l'intera catena non è eseguita con i privilegi di amministratore di sistema dunque lascia al sistema operativo la normale gestione delle risorse dell'host che non le competono e non può (in linea di principio) interagire con esse.

2 Uso di Singularity

2.1 Installazione

Singularity si installa comodamente su sistemi Linux, le librerie propedeutiche sono:

```
sudo apt-get update && sudo apt-get install -y \
build-essential \
libssl-dev \
uuid-dev \
libgpgme11-dev \
squashfs-tools
```

Essendo scritto in Go, per essere compilato necessita della sua installazione; ecco come effettuare le due cose in sequenza:

```
#!/bin/bash
#####
#
#  Script per installare GO e Singularity
#
#  Attenzione !
#    *  Prima di lanciarlo pulire il proprio ~/.bashrc
#    *  Valutare di aggiornare l'environment di tutti gli utenti
#
#####
export SV=3.5.3
export SG=1.14
sudo rm -r /usr/local/go
sudo rm -r /opt/singularity-${SV}
sudo mkdir /opt/singularity-${SV}
sudo chmod -Rv 777 /opt/singularity-${SV}

export VERSION=${SG} OS=linux ARCH=amd64 && \
    wget https://dl.google.com/go/go$VERSION.$OS-$ARCH.tar.gz && \
    sudo tar -C /usr/local -xzvf go$VERSION.$OS-$ARCH.tar.gz && \
    rm go$VERSION.$OS-$ARCH.tar.gz

echo 'export GOPATH=${HOME}/go' >> ~/.bashrc && \
    echo 'export PATH=/usr/local/go/bin:${PATH}:${GOPATH}/bin' >>
        ~/.bashrc && \
    source ~/.bashrc

go get -u github.com/golang/dep/cmd/dep

export VERSION=${SV} && # adjust this as necessary \
    mkdir -p $GOPATH/src/github.com/sylabs && \
    cd $GOPATH/src/github.com/sylabs && \
    wget https://github.com/sylabs/singularity/releases/download/v${
        VERSION}/singularity-${VERSION}.tar.gz && \
    tar -xzf singularity-${VERSION}.tar.gz && \
    cd ./singularity && \
    ./mconfig --prefix=/opt/singularity-${VERSION}
    make -C ./builddir
    sudo make -C ./builddir install
```

```
echo 'export PATH=${PATH}:/opt/singularity-${VERSION}/bin' >> ~/.
    bashrc && \
    source ~/.bashrc
```

2.2 Creazione di un container

2.2.1 Scrittura della *recipe*

La recipe è un file di *meta-istruzioni* interpretato da Singularity per generare il **container** che è un oggetto che ingloba il sistema operativo e il software che scegliamo che contenga, disaccoppiandosi dall'host su cui è eseguito, dunque anche dall'hardware e dal sistema operativo dell'host. Di seguito la recipe di CHIMBO-2020, in cui possiamo notare queste sezioni:

- bootstrap: sta ad indicare a Singularity che il contantainer deve essere costruito partendo dal container debian-buster presente in libreria remota. Nel caso in cui si volesse partire da uno già esistente, di seguito chiamato *step1.sif*, andrebbe indicato:

  ```
  BootStrap:localimage
  From:step1.sif
  ```

- setup: in questa fase il focus è sull'host e non nell'embrione del container per cui si approfitta per indicare al sistema di utilizzare la cartella in cui risiederà il container anche per la scrittura dei file temporanei, in modo da non mandare in crisi il sistema operativo; ciò perchè CHIMBO-2020 è un'applicazione relativamente pesante e si potrebbe riempire tutto il volume del sistema operativo.

- files: copia dall'host al container dei sorgenti che verranno poi compilati nella fase *post* e dei dati "eterni" di CHIMBO-2020.

- post: installazione dei programmi tramite repository debian e compilazione di quelli propedeutici e costituenti la catena CHIMBO-2020.

- help: risposta al comando *singularity run-help CHIMBO_sandbox*.

```
BootStrap: library
From: debian:buster

%setup
  _pwd=$(pwd)
  mkdir -p ${_pwd}/tmp
  export SINGULARITY_TMPDIR=${_pwd}/tmp

  cat > ${_pwd}/INSTALL.sh << EOF
#!/bin/bash
singularity exec -H ${_pwd} ${_pwd}/chimbo_sandbox bash -c "cd /opt
    && ./install.sh"
EOF
```

```
  chmod +x ${_pwd}/INSTALL.sh

%files
  SOURCES/automake-1.16.tar.gz /opt
  SOURCES/openmpi-3.1.6.tar.gz /opt
  SOURCES/hdf5-1.8.21.tar /opt
  SOURCES/netcdf-c-4.7.3.tar.gz /opt
  SOURCES/netcdf-fortran-4.5.2.tar.gz /opt
  SOURCES/cmake-3.17.1.tar.gz /opt
  SOURCES/jasper-1.900.1.tar.gz /opt
  SOURCES/eccodes-2.17.0-Source.tar.gz /opt
  SOURCES/gfs_data_download /opt
  SOURCES/ctm /opt
  SOURCES/ibc /opt
  SOURCES/pacchetto_meteo /opt
  SOURCES/install.sh /opt
  SOURCES/b2c /opt
  SOURCES/bf /opt
  SOURCES/geo_dataset /opt
  SOURCES/ibc/spec_info /opt/ibc
  SOURCES/ibc/static_data /opt/ibc
  SOURCES/script /opt

%environment

%runscript

%startscript

%test

%labels
%help
  debian:buster
  automake-1.16
  openmpi-3.1.6
  hdf5-1.8.21
  netcdf-c-4.7.3
  netcdf-fortran-4.5.2
  eccodes-2.17.0

. Per creare il container:
singularity build --fakeroot --sandbox chimbo_sandbox CHIMBO.def
. Per installare la catena:
./INSTALL.sh
. Se si volesse usare crontab per automatizzare l azionamento degli
    script, aggiungervi:
SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/
    opt/singularity-3.5.3/bin

%post
  echo "***** Hello from inside the container *****"
  apt-get update
  apt-get install -y texinfo vim dialog locales libtool libtool-bin
      autoconf git flex gawk build-essential gfortran g++ zlib1g-dev
       csh libcurl4-gnutls-dev libssl-dev wget subversion nco cdo
```

```
    locate python

strip --remove-section=.note.ABI-tag /usr/lib/x86_64-linux-gnu/
    libQt5Core.so.5

chmod -Rv 777 /opt/gfs_data_download

cd /opt
tar xzvf automake-1.16.tar.gz
rm automake-1.16.tar.gz
    cd /opt/automake-1.16
    sed -i 's:/\\\${:/\\\$\\{:' bin/automake.in
    autoreconf -i -f
    ./configure --prefix=/usr/local --docdir=/usr/share/doc/
        automake-1.16
    make
    make install

cd /opt
tar xzvf jasper-1.900.1.tar.gz
rm jasper-1.900.1.tar.gz
cd /opt/jasper-1.900.1
./configure --prefix=/usr/local
make
make install
ldconfig

cd /opt
tar xzvf openmpi-3.1.6.tar.gz
# rm openmpi-3.1.6.tar.gz
cd /opt/openmpi-3.1.6
./configure --prefix=/usr/local --enable-static
    make all install
    ldconfig

cd /opt
    tar xvf hdf5-1.8.21.tar
rm hdf5-1.8.21.tar
    cd /opt/hdf5-1.8.21
    CC=/usr/local/bin/mpicc ./configure --prefix=/usr/local --
        enable-fortran --enable-cxx
    make  && make check && make install
    ldconfig

cd /opt
tar xvf netcdf-c-4.7.3.tar.gz
rm netcdf-c-4.7.3.tar.gz
cd netcdf-c-4.7.3
CC=mpicc  CFLAGS='-fPIC' LDFLAGS=-L/usr/local/lib CPPFLAGS=-I/usr/
    local/include ./configure --prefix=/usr/local --disable-dap-
    remote-tests
make
make check
make install
ldconfig

cd /opt
```

```
tar xzvf netcdf-fortran-4.5.2.tar.gz
rm netcdf-fortran-4.5.2.tar.gz
cd /opt/netcdf-fortran-4.5.2
CPPFLAGS="-I/usr/local/include -DgFortran" LDFLAGS=-L/usr/local/
    lib FC=mpif90 F77=mpif90 CC=mpicc F77FLAGS="-O3 -fno-second-
    underscore" FCFLAGS="-O3 -fno-second-underscore" FFLAGS="-O3 -
    fno-second-underscore" ./configure --prefix=/usr/local --
    enable-shared
make check install
ldconfig

cd /opt
tar xzvf cmake-3.17.1.tar.gz
rm cmake-3.17.1.tar.gz
cd cmake-3.17.1
./bootstrap
make
make install

cd /opt
tar xzvf eccodes-2.17.0-Source.tar.gz
rm eccodes-2.17.0-Source.tar.gz
mkdir build
cd build
cmake ../eccodes-2.17.0-Source
make
ctest
make install
ldconfig

cd /opt/pacchetto_meteo/rad-ecmwf-new/package
./makeall

cd /opt/pacchetto_meteo
make
```

2.2.2 Building del container

Un container può essere creato in due formati:

- **binario**: è un file monolitico, non modificabile ed *eseguibile*.
- **sandbox**: è un albero di cartelle che ricalca quelle di un sistema, così come creato nella recipe, è *scrivibile* e dunque utile per la testing. Nel caso specifico di CHIMBO-2020, poichè alcuni anelli della catena scrivono file di controllo in directory non convenzionali, come ad esempio la cartella stessa in cui sono presenti gli eseguibili o gli script, è necessario che *CHIMBO-2020 venga buildato in questo formato*.

Per realizzare il container bisogna usare il comando *singularity build* con i privilegi di root; successivamente il container può essere copiato sull'host di produzione su cui sarà eseguito. Nel caso in cui non si disponesse dei privilegi di root sull'host in cui deve essere buildata la recipe, sarebbe necessario che il sistemista impostasse:

```
sudo sysctl -w kernel.unprivileged_userns_clone=1
```

In questo modo l'utente può eseguire il building utilizzando l'opzione – *fakeroot*:

```
singularity build --fakeroot --sandbox chimbo_sandbox  CHIMBO.def
```

In fase di building viene usata la cartella /tmp dell'host che potrebbe non contenere il container per cui si deve specificare una cartella temporanea differente andando a impostare la variabile d'ambiente *SINGULARITY_TMPDIR*:

```
export SINGULARITY_TMPDIR=/home/grasso/cantiere_CHIMBO/tmp
```

Ciò andrebbe fatto prima di lanciare il build oppure nella fase *setup* del building.

2.2.3 Modifica della sandbox

La peculiarità del formato *sandbox* è che consente di navigare nel file system virtuale e modificarlo per adattarlo ad esigenze di sviluppo. Ovviamente le modifiche apportate in questo modo non hanno un corrispettivo nella recipe che ha generato il container per cui andrebbe aggiornata per assicurarsi che il container sia riproducibile. Per poter accedere alla sandbox bisogna dare il comando:

```
singularity shell -H working_directory sandbox_folder
```

L'utente entra nella shell con il suo stesso nome e privilegi e la sua home e quella indicata con l'opzione *-H*.

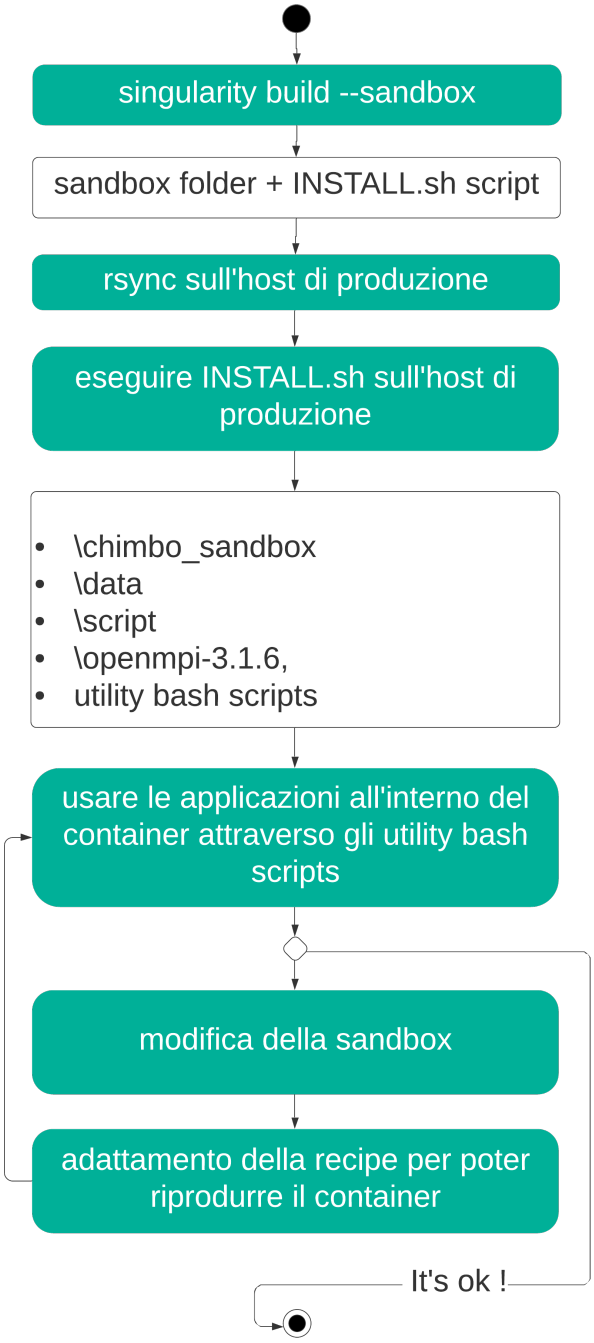

Figure 1: Flusso di lavoro

3 Installazione e manutenzione della catena CHIMBO-2020

3.1 Installazione

Nell'istanza Gitlab CE ospitata all'url **https://git.isac.cnr.it** e riservata ai dipendenti ISAC, sono stati creati due repository:

- **sviluppo**: tutto ciò che è necessario per ricreare il container: la recipe e i sorgenti degli applicativi. Chi volesse evolvere la catena CHIMBO-2020 potrebbe:
  - clonare il repository:

    ```
    git clone ssh://git@git.isac.cnr.it:2224/grasso/CHIMBO
        -2020.git
    ```

    ed effettuare il build del container, così come illustrato nei paragrafi precedenti;
  - creare un nuovo branch: i branch ('ramificazioni') sono utilizzati per sviluppare features che sono isolate l'una dall'altra. Il branch master è quello di default, è possibile usare altri branch per lo sviluppo ed infine incorporarli ('merge') nel master branch una volta completati. Si crea un nuovo branch chiamato "feature-x" e si passa ad esso usando:

    ```
    git checkout -b feature_x
    ```
  - sviluppare e testare in locale il nuovo branch;
  - inviare sul server remoto l'evoluzione del branch, in modo che si possa valutare il *merge* di queste modifiche con il branch principale, vale a dire il *master*.

    ```
    git push origin <branch>
    ```

- **produzione**: questo repository è pensato per porre subito in essere una istanza di CHIMBO-2020 operativa, senza preoccuparsi del suo miglioramento. Per quanto sia possibile evolvere la catena anche partendo dalla sandbox, le modifiche che non trovano corrispondenza nella recipe fanno si che si perda la possibilità di ricreare il container e quindi in sostanza di tenere traccia delle modifiche stesse. Per scaricare la versione operativa della catena, è sufficiente dare il comando:

  ```
  git clone ssh://git@git.isac.cnr.it:2224/grasso/CHIMBO-2020-
      container.git
  ```

  e poi lanciare la procedura di installazione:

```
singularity exec -H /home/user/fold /home/user/fold/
    chimbo_sandbox bash -c "cd /opt && ./install.sh"
```

La procedura di installazione, che è nello script */opt/install.sh* interno al container, è un ponte fra l'ambiente dell'host e quello del container. Per quanto la filosofia del lavoro sia quella di avere un oggetto stateless e trasportabile, l'eterogeneità delle applicazioni e l'adattarsi al numero di processori dell'host ospitante fanno sì che si debba creare un legame fra host e container e che il focus passi da uno all'altro con cognizione delle variabili di ambiente. Detto ciò, la procedura di installazione colloca sull'host:

- gli script per il lancio delle applicazioni: gfs.sh, ibc.sh, prebolam.sh, bolam.sh, b2c.sh, CHIMERE1.sh e CHIMERE2.sh . Questi script possono essere usati in crontab per automatizzare l'uso della catena:

```
    SHELL=/bin/bash

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/
    bin:/opt/singularity-3.5.3/bin:$PATH

30 00 * * * /home/user/fold/gfs.sh>/home/user/fold/gfs.log 2>&1

00 01 * * * /home/user/fold/ibc.sh>/home/user/fold/ibc.log 2>&1

00 04 * * * /bin/bash -c "source /home/user/fold/prebolam.sh >/
    home/user/fold/prebolam.log 2>&1"

00 05 * * * /home/user/fold/bolam.sh >/home/user/fold/bolam.log

00 11 * * * /bin/bash -c "source /home/user/fold/b2c.sh >/home/
    user/fold/b2c.log 2>&1"

30 11 * * * /home/user/fold/chimere1.sh >/home/user/fold/
    CHIMERE1.log 2>&1

00 14 * * * /home/user/fold/chimere2.sh >/home/user/fold/
    CHIMERE2.log 2>&1
```

- la cartella 'script', che le applicazioni usano per scambiare il focus fra host e container.
- la compilazione di openmpi-3.1.6: è importante che il container e l'host abbiano la stessa versione di openmpi per cui compilarlo nella cartella di installazione aumenta l'affidabilità.
- i dati 'eterni', che non devono essere aggiornati e sono necessari all'operabilità della catena.
- le variabili di Bolam usate per parallelizzare:
  - NNODE: numero di nodi

  - NPROC-NODE: processori logici per nodo

- le variabili di CHIMERE 2014b usate per parallelizzare:
  - NPROCS: numero di processi, usualmente è pari al numero di processi logici su cui è parallelizzato CHIMERE più uno, per cui su un host su cui si intende parallelizzare su 16 processori, va indicato 17.
  - NZDOMS nelle sue tre componenti
  - NMDOMS nelle sue tre componenti

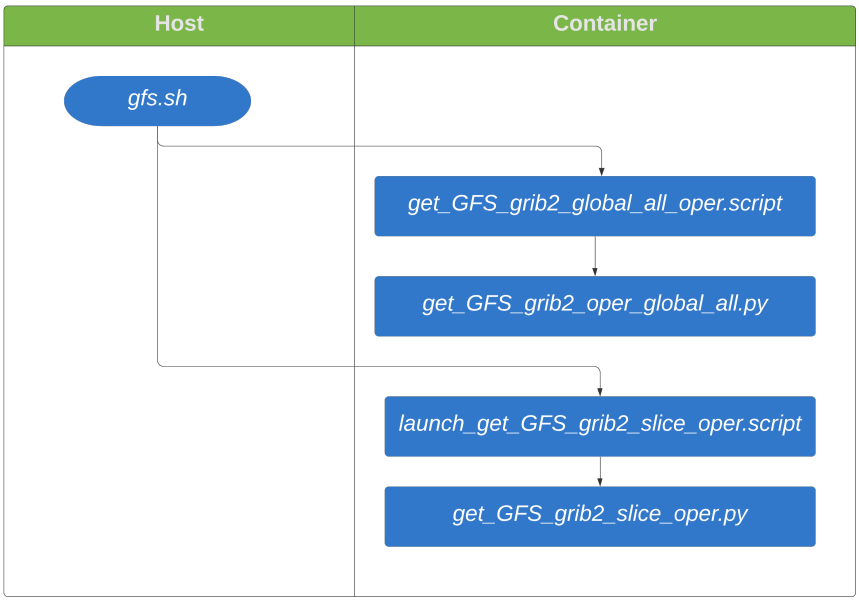

Figure 2: GFS

3.2 Diagrammi operativi delle applicazioni

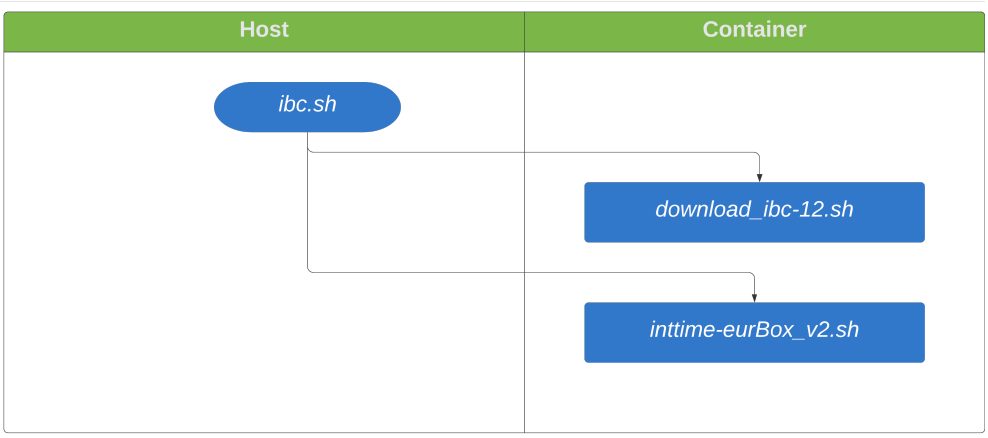

Figure 3: IBC

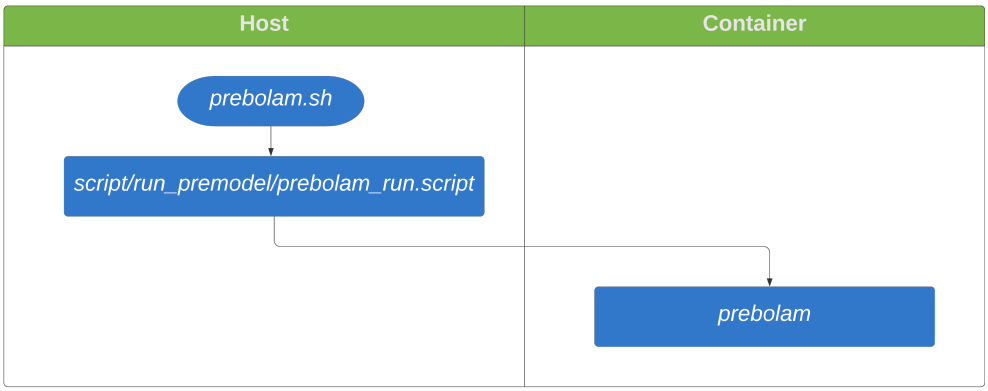

Figure 4: PreBOLAM

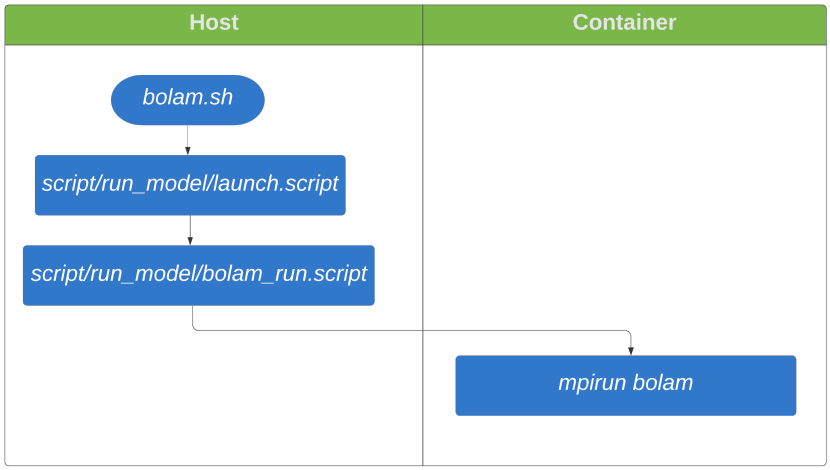

Figure 5: BOLAM

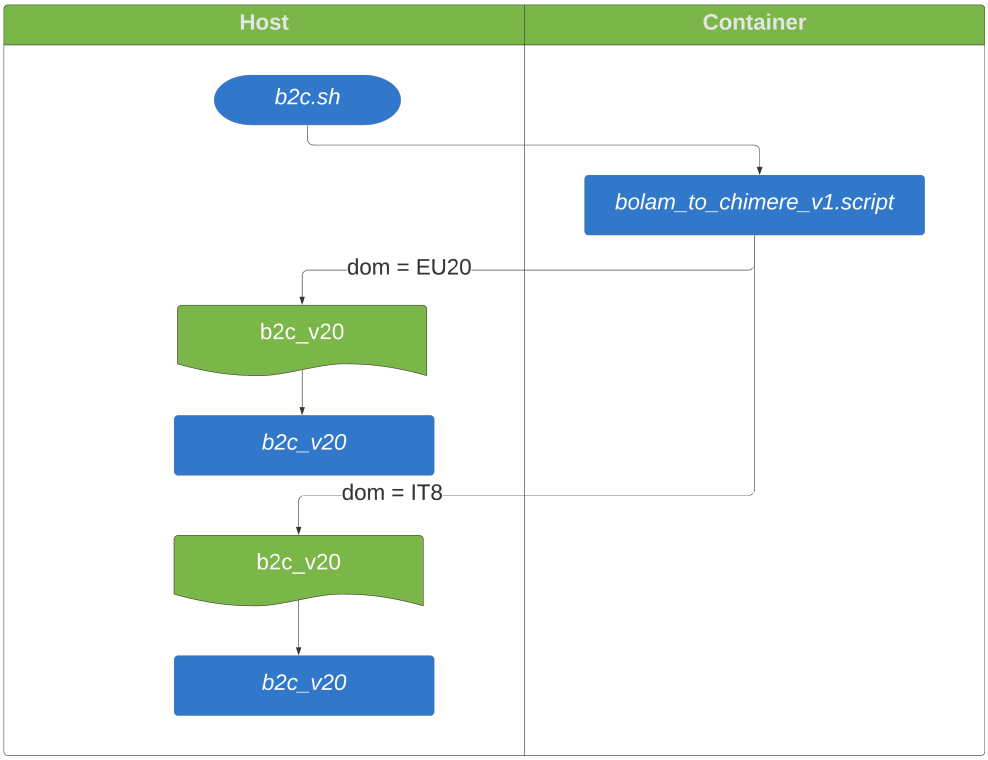

Figure 6: B2C

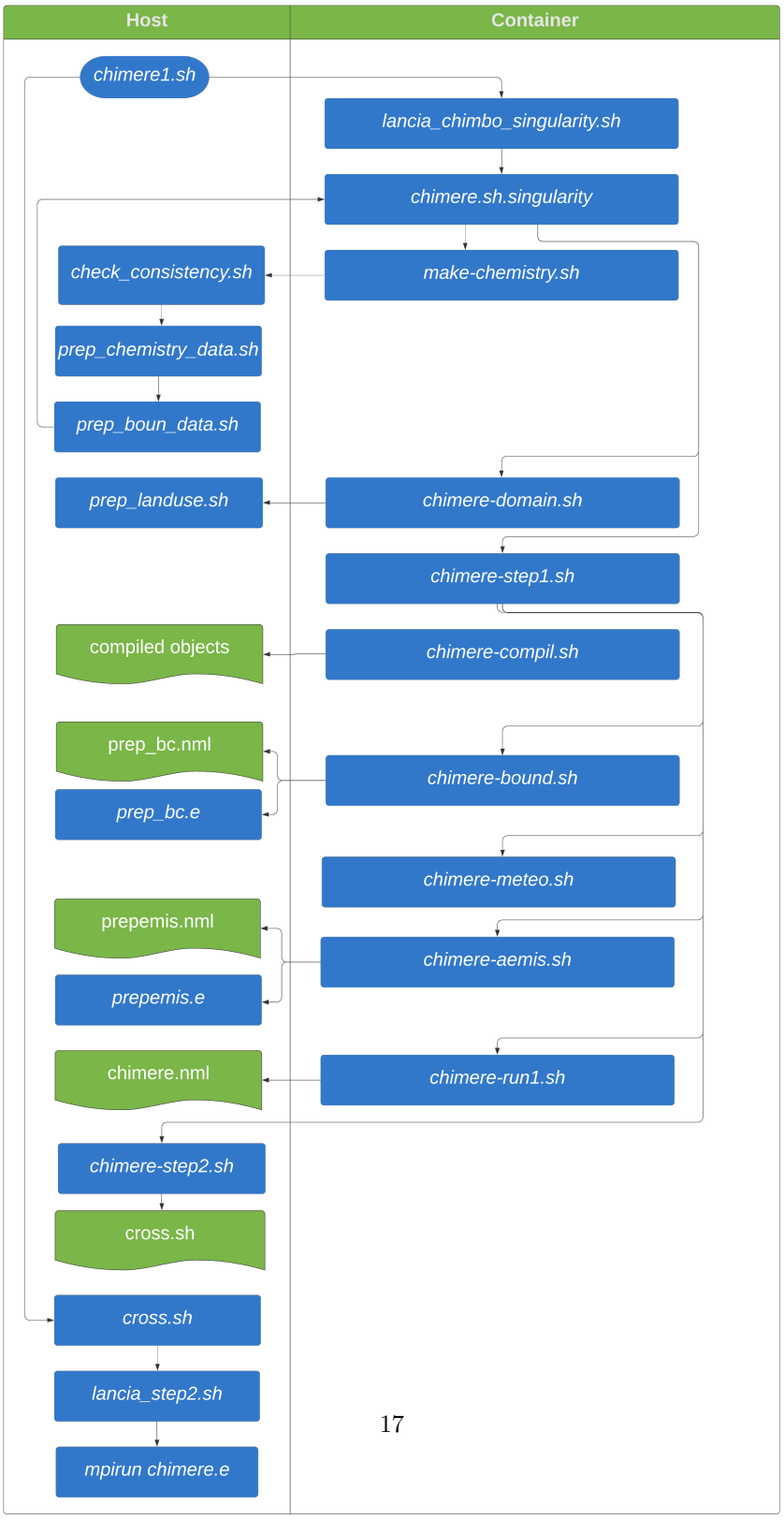

Figure 7: CHIMERE v2014b

References

[1] Fabio Massimo Grasso, Leopoldo Fazioli, Tony Landi, and Oxana Drofa. Architettura stateless con uso di singularity per il porting degli applicativi wrf-4.1.5 e bolam. Technical report, 2020.

[2] Tony Christian Landi and Piero Malguzzi. A new high-resolution air quality modeling chain implemented over italy. *ICAM 2019, Riva del Garda*, Conference Proceedings, 2019.