ARPA FVG
Agenzia Regionale per la Protezione dell'Ambiente del Friuli Venezia Giulia

Art. Tec-Sci No. 001/2017 (07/06/2017)
ARPA FVG - CRMA

Elementi essenziali per la costruzione delle suite per l'uso dell'ecFlow

Dario Giaiotti[1]

Sommario
Realizzare e gestire il flusso di dati e di elaborazioni che danno una risposta ad un problema, in forma numerica o grafica per mezzo di infrastrutture computazionali, non è un compito semplice se ad essere coinvolti sono un numero grande di basi dati ed applicativi interagenti. Inoltre se la realizzazione del flusso implica il lavoro di più di una persona allora la complessità aumenta, così pure il rischio di introdurre errori o di impiegare il tempo nell'amministrare e controllare aspetti marginali ma funzionali al conseguimento della risposta. Per questo motivo il CRMA dell'ARPA FVG si è dotato di un software che svolge il compito di workflkow manager e realizza i flussi secondo alcune regole che rendono standard la procedura, agevolando la ripartizione dei compiti tra colleghi e semplificando le operazioni di controllo e l'analisi dei flussi.

Keywords
Workflow, tasks ecFlow, operatività

[1] *ARPA FVG - CRMA*
*__Autore di riferimento__: dario.giaiotti@arpa.fvg.it

Indice

1. Introduzione

I flussi di dati e di applicativi che svolgono compiti operativi o di ricerca e sviluppo del CRMA, nell'ambiente HPC, sono gestiti tramite il workflow manager ecFlow [1]. Il vantaggio dell'utilizzo di un workflow manager consiste nell'avere definito ed implementato numericamente il flusso computazionale e di conseguenza poterlo monitorare, eseguire diagnosi, individuare problemi, riprodurre o clonare esperimenti computazionali con uno sforzo minimo e con una bassissima probabilità di commettere errori; inoltre la portabilità dei flussi è notevolmente agevolata. Ovviamente tutti questi vantaggi non vengono *Gratis et amore Dei*, ma derivano dal lavoro svolto dai progettisti e sviluppatori del software che implementa il workflow manager, l'ecFlow [1] nel nostro caso specifico, e dall'impegno di chi progetta e realizza il flusso, cioè i modellisti del CRMA.

Infatti un flusso ben progettato e realizzato tramite un workflow manager dalle avanzate funzionalità gestionali, permette di eseguire un numero di compiti elevatissimo, anche molto diversi gli uni dagli altri, richiedendo un impegno minimo da parte dell'utilizzatore. Quindi è estremamente importante seguire due requisiti:

- il realizzatore dei flussi conosca le caratteristiche del workflow manager;
- che progetti i suoi flussi con logica, lungimiranza e chiara visione dei risultati che intende conseguire.

Questi requisiti necessitano un significativo impegno iniziale. Per il primo requisito si suggerisce lo studio del manuale per l'utente e l'utilizzo dei tutorial per l'ecFlow [1], oltre che alla letteratura introduttiva ai workflow manager [2],[3],[4]. Per il secondo è possibile procedere secondo attitudine anche se l'adozione di adeguati linguaggi, strumenti concettuali e documentali per la progettazione possono risultare di grande aiuto [5], [6]. La scelta del CRMA di dotarsi del software ecFlow consegue da uno studio delle caratteristiche dei workflow manager disponibili in modalità *free*[7][8][9][10] negli anni 2013-2014 e sulla base delle esperienze derivanti dai flussi computazionali operativi che, a quel tempo, erano già stati implementati dai modellisti del CRMA, prevalentemente tramite scripting BASH. I requisiti essenziali soddisfatti da ecFlow sono riassumibili in:

- autonomia gestionale del software da parte del CRMA, sia per quanto riguarda l'installazione, la gestione e gli aggiornamenti;
- certezza che il software sia mantenuto aggiornato da un supporto informatico competente, il quale fornisca anche assistenza, per un periodo di anni sufficientemente lungo da essere confrontabile con i tempi evolutivi della modalità di gestione dei flussi computazionali del CRMA;

- il software sia adottato operativamente almeno da una struttura tecnico scientifica che si occupa di modellistica ambientale ad alto livello (ECMWF)[11];

- i tipi di flussi per i quali è stato progettato il software siano simili, se non uguali, a quelli che il CRMA gestisce e gestirà in futuro.

Ciò premesso, qui di seguito vengono date le indicazioni minimali che il progettista dei flussi di dati e di applicativi del CRMA è opportuno segua. Tramite ecFlow il flusso viene descritto ed implemento da una **suite** (vedi il manuale per la definizione [1]). Nel seguito saranno dati per scontati alcuni concetti e la terminologia essenziale usata per la costruzione dei flussi tramite ecFlow.

2. Costruzione della suite

Ogni suite, da utilizzare tramite workflow manager ecFlow [1], deve essere costruita secondo i seguenti criteri.

Viene creata una directory con il nome della suite nella directory radice che ospita tutte le suite; negli esempi che seguiranno si farà riferimento alle suite gestite dall'utente operative. Per gli scopi operativi, sul cluster di calcolo FENICE, si usa la directory $HOME/src/operative_workflows/**operative**. Inoltre esiste una directory radice dove sono presenti le suite non operative che sono in via di definizione finale o sottoposto a test; tale directory è la $HOME/src/operative_workflows/**develop**.

Alla suite va attribuito un nome costituito al massimo da 15 caratteri, per esempio:

WRF_oper oppure **11B181B0B1_2007**.

A partire da questo nome si genera il file che definisce la suite, avente lo stesso nome della suite più l'estensione **.def**. La corrispondente directory ha lo stesso nome della suite ed entrambe esistono allo stesso livello gerarchico, ovvero nella directory radice; si veda figura 1.

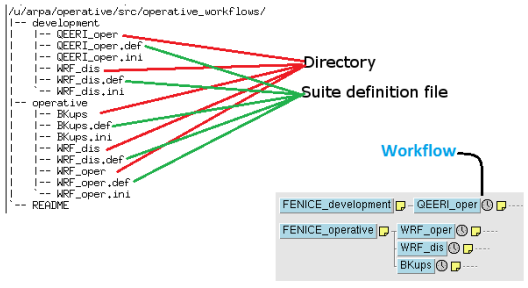

Figura 1. L'immagine mostra l'arborescenza delle directory, con i corrispondenti file di definizione dei workflow, e la visualizzazione dei workflow tramite l'interfaccia grafica del workflow manager ecFlow

All'interno della directory di suite è opportuno creare alcune sottodirectory che saranno utilizzate per depositare files di rilevanza comune. In particolare la directory **include**, il cui path sarà utilizzato per la ricerca dei file da includere durante il preprocessing dei task. Nella include conviene depositare i file di interesse comune per tutta la suite, per esempio i file **head.h** e **tail.h** che iniziano e terminano ogni task, mentre è opportuno creare delle specifiche sottodirectory, come la **man** che contiene i manuali, i quali sono uno per task quindi potenzialmente molti. In questo modo si può includere il manuale nel task, con l'operazione di preprocessing %include, indicando il path relativo che dalla directory radice **include** individua il file del manuale da includere nel task. Vedi figura 2.

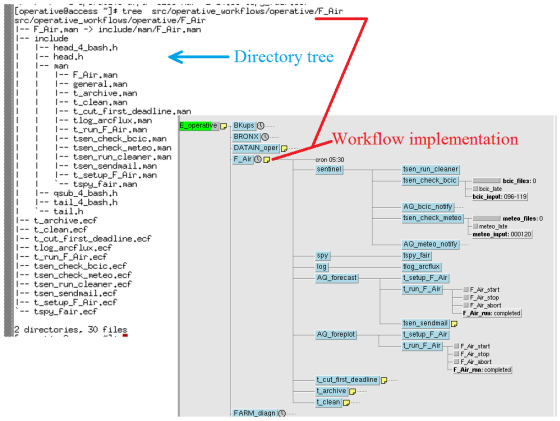

Figura 2. Arborescenza della directory dedicata al workflow e sua implementazione e visualizzazione trame ecFlow.

Ogni suite deve essere composta da **famiglie** di **task**. Il raggruppamento dei task in famiglie ha lo scopo di rendere ordinato e congruente il flusso all'interno della famiglia. Tutti gli script che implementano i task è opportuno risiedano nella directory radice della suite. Infatti, secondo la logica con cui è stato realizzato il workflow manager, gli script vengono cercati ricorsivamente nell'alberatura della directory della suite, fino al livello più alto, cioè alla radice. Qualora siano necessari dei controlli che attivino dei trigger o che producano eventi a seguito dei quali l'evoluzione delle famiglie e dei task cambia, i task di controllo debbono appartenere ad una speciale famiglia della suite, detta famiglia delle sentinelle; tale famiglia si chiamerà sempre **sentinel** e non c'è rischio di confusione in quanto risiede all'interno della suite ed è unica. Nella famiglia sentinel, all'occorrenza può risiedere l'elenco dei limiti che vengono imposti, con il comando di ecFlow ***limit***, all'esecuzione contemporanea dei nodi della suite. Di prassi ogni suite ha la propria sentinel family. Analogamente ogni suite dovrebbe essere dotata di una famiglia di task che segnalano l'andamento ed eseguono il monitoraggio dei task della suite. Tale famiglia si chiamerà **spy** risiede all'interno della suite ed è unica. La suite avrà anche sempre una family **log** che conterrà almeno un task che archivia i log file e i job file di tutta la suite. L'archivio sarà un file tar compresso che verrà spostato in un'opportuna cartella per il seguente backup. Qualora una o più delle tre famiglie, che debbono essere presenti di prassi, ovvero sentinel, spy e log, non venisse popolata da task è

opportuno sia comunque presente nel flusso e le sarà attribuito lo status di completata a priori tramite il comando defstatus complete.
Ogni file che definisce la suite (file avente estensione .def) deve contenere **un'intestazione di commenti**, ovvero righe che iniziano con il carattere "#" nella quale, oltre a commenti sono presenti alcune **parole chiave** che saranno usate per la descrizione e l'analisi automatica dei flussi di programmi. Ogni parola chiave deve iniziare con i sei caratteri SUITE_, e proseguire con caratteri maiuscoli. La parola chiave termina con il carattere ":", due punti, ed è seguita dal valore attribuito alla parola chiave che è la stringa seguente fino al termine della linea o all'incontro del carattere "#". Per esempio SUITE_AUTHOR: Cesare Augusto. Ulteriori estensioni dell'intestazione dovranno, a parte il primo carattere #, essere formattate secondo la sintassi YAML [12]
Ci sono **alcune parole chiave che debbono essere presenti di default** e alle quali vanno attribuiti dei valori, esse sono:

 # SUITE_NAME: Nome attribuito alla suite
 # SUITE_OWNER: l'ente proprietario; per esempio ARPA FVG CRMA
 # SUITE_AUTHORS: l'autore della suite; per esempio Dario B. Giaiotti
 # SUITE_DESCRIPTION: una breve descrizione; ad esempio Operational run of WRF atmospheric model
 # SUITE_CREATION-DATE: la data di creazione espressa con il formato YYYYMMDD
 # SUITE_LAST_MODIFIED: la data di ultima modifica espressa con il formato YYYYMMDD
 # SUITE_TYPE: tipo di suite tra i seguenti:operational, development, research or user

La figura 3 mostra l'inizio di una suite operativa.

```
[operative@access tmp]$ more  /u/arpa/operative/src/operative_workflows/operative/WRF_oper.def
# +-------------------------------------------------------------------------+
#
# This is the ecFlow suite definition driving the operational run of wrf model
# on the Friuli Venezia Giulia domain for operational purposes
#
#
#
# SUITE_NAME:           WRF_oper
# SUITE_OWNER:          ARPA FVG - CRMA
# SUITE_AUTORS:         Dario B. Giaiotti
# SUITE_DESCRIPTION:    Operational run of WRF atmospheric model for ARPA FVG operational domain
# SUITE_CREATION-DATE:  20151102        # Date expressed as YYYYMMDD
# SUITE_LAST_MODIFIED:  20151102        # Date expressed as YYYYMMDD
# SUITE_TYPE:           operative       # May be operative, development,
#                                       # research or user
#
# +-------------------------------------------------------------------------+
# | Remember to replace the "development" word with the "operative" one when |
# | this suite is moved from one repository to the other. Use global replace-|
# | ment in this suite definition file.                                      |
# +-------------------------------------------------------------------------+
#
#
suite WRF_oper
  #
  #   ----> D E F I N I T I O N      S E C T I O N <----
  #
  #   CLOCK DEFINITION
  #
  clock real       #This suite user the real clock
  #
  #   ENVIRONMENTAL VARIABLES DEFINITION
  #
  # Directory. It is used as a prefix portion of the path of the job files created by ecFlow server
  #
    edit ECF_HOME    /lustre/arpa/operative/scratch/ecflow_suites/operative
  #
  # Directory. It is used to find the ecFlow files
  #
    edit ECF_FILES   /u/arpa/operative/src/operative_workflows/operative
  #
  # Directory: It is used to find the files to be included in the nodes
  #
    edit ECF_INCLUDE /u/arpa/operative/src/operative_workflows/operative/$SUITE/include
  #
  # Number of times a job should be rerun if it aborts (default is 2)
  #
    edit ECF_TRIES  2
  #
--More--(47%)
```

Figura 3. L'inizio del file che definisce una suite operativa

Ogni suite deve avere la sua **definizione di clock**, che può essere **real** o **hybrid**. Pur essendo il clock di tipo real assunto di default nel caso non venga definito esplicitamente nella suite, si caldeggia la sua definizione anche nel caso real per motivi di chiarezza.

Nella suite vanno sempre definite le variabili d'ambiente che saranno ereditate da tutte le famiglie e i task. Si ricordi che nel file di definizione, la variabile **$SUITE** assume il valore del nome della suite.

ECF_HOME è la directory che viene utilizzata per lo svolgimento dei compiti della suite. Qui potranno essere generati eventuali file prodotti dalla suite durante l'esecuzione, per esempio i job file e i corrispondenti output file. Per gli scopi operativi si usa una directory temporanea /lustre/arpa/operative/scratch/ecflow_suites/operative/ e all'interno di questa directory, ecFlow genera una sottodirectory **#SUITE** nella quale saranno depositati i files. Sono generate anche parent directory, se nella definizione di ECF_HOME mancano per completare il path definito.

ECF_FILES è la directory che viene utilizzata per tutti i nodi della suite che sono definiti mediante il file .def, in particolare i task file aventi estensione .ecf. Per gli scopi operativi si usa la directory $HOME/src/operative_workflows/operative/. Si ricordi che lo script del nodo viene ricercato a partire da questa directory esplorando tutto l'albero delle famiglie coinvolte nella suite.

ECF_INCLUDE è la directory che viene utilizzata per tutti i file che saranno inclusi tramite il comando include. Per gli scopi operativi si usa la directory $HOME/src/operative_workflows/operative/$SUITE/include. In questo caso è possibile includere **$SUITE** nel path.

SLEEP Definisce i secondi che si intende far trascorrere alla fine di ciascun task. Non necessario, ma suggerito. Se si intende definirlo si provi con il valore 1. Il valore di attesa deve essere incluso nei task con il comando bash sleep associandolo al valore assunto dalla variabile SLEEP che viene ereditato dai nodi della suite. Per esempio si può aggiungere il comando sleep %SLEEP% nella tail che viene inclusa in ciascun task.

ECF_TRIES Definisce il numero di volte che il job deve essere rieseguito automaticamente se abortisce. Conviene fissarlo al valore di default che è 2, ma se si pensa che potrebbe essere il caso di ritentare più volte, per esempio per una suite che esegue degli FTP, che possono fallire, conviene aumentare il numero. Il numero di tentativi automatici, in combinazione con l'uso della variabile di suite %ECF_TRYNO% accessibile in ciascun task, permette la deviazione automatica del flusso in caso di problemi.

E' generalmente utile definire una variabile ambientale di suite che individua un file di inizializzazione, il quale viene poi utilizzato per personalizzare l'ambiente dei task, tramite il comando source della SHELL. Tale variabile assume un nome ben preciso SUITE_INI_FILE

SUITE_INI_FILE Viene definito al path del file di inizializzazione comune a tutti i task della suite

Un esempio di suite definition file è riportata nella figura 4, oppure possono essere prese come prototipo le suite opera-

tive clonando il repository Git remoto: ssh://git@grid1.mercuriofvg.it/operative_workflows.git

```
            # +-------------------------------------------------------------------+
            #
# This is the ecFlow suite definition driving the operational run of wrf model
#
# SUITE_NAME:        wrfop
# SUITE_OWNER:       ARPA FVG - CRMA
# SUITE_AUTORS:      Dario B. Giaiotti
# SUITE_DESCRIPTION:   Operational run of WRF atmospheric model
# SUITE_CREATION-DATE: 20151001       # Date expressed as YYYYMMDD
# SUITE_LAST_MODIFIED: 20151002       # Date expressed as YYYYMMDD
# SUITE_TYPE:        operational     # May be operational, development,
#                      # research or user
#
# +-------------------------------------------------------------------+
#
suite wrfop
            #
            # ---->D E F I N I T I O N    S E C T I O N<----
            #
            #  CLOCK DEFINITION
            #
            clock real    #This suite user the real clock
            #
            #  ENVIRONMENTAL VARIABLES DEFINITION
            #
            # Directory. It is used as a prefix portion of the path of the job files created by ecFlow server
            #
             edit ECF_HOME   /lustre/arpa/operative/scratch/ecflow_suites/develop
            #
            # Directory. It is used to find the ecFlow files
            #
             edit ECF_FILES  /u/arpa/operative/src/ecflow_suites/develop
            #
            # Directory. It is used to find the files to be included in the nodes
            #
             edit ECF_INCLUDE /u/arpa/operative/src/ecflow_suites/develop/$SUITE/include
            #
            # Number of times a job should be rerun if it aborts (default is 2)
            #
             edit ECF_TRIES 3
            #
            # Suite initialization file.
            # this is the path to the file storing environmental variables characterizing the whole suite
            #
             edit SUITE_INI_FILE  /u/arpa/operative/tmp/operative_workflows/development/QEERI_oper.ini
            #
            #
            # ---->F A M I L Y    S E C T I O N<----
            #
            #
endsuite
```

Figura 4. Esempio di file definition della suite

3. Caratteristiche comuni a tutte le famiglie

Ogni famiglia deve avere un nome che al meglio individui il ruolo che svolge all'interno del flusso. Fanno eccezione le famiglie **log**, **sentinel** e **spy** i cui nomi sono riservati ai compiti sopra descritti.

4. Caratteristiche comuni a tutti i task

Ogni task deve avere un nome che inizia con i due caratteri **t_**, fanno eccezione i task delle famiglia sentinel che iniziano con i caratteri **tsen_**, quelli della famiglia log che iniziano per **tlog_** e quelli della famiglia spy che iniziano con i caratteri **tspy_**. Tutti i task includono una header, ovvero un preambolo di istruzioni, vedi figura 5, e una tail, ovvero una coda di istruzione, vedi figura 6, nelle quali sono inseriti sia le comunicazioni con il server del workflow manager, sia alcune informazioni utili, riguardanti gli scopi del task, che potrebbero essere utilizzate per la diagnosi ed il monitoraggio dello stato di avanzamento della suite con dei semplici grep.

```
Preliminary information about the run of this task
SUITE NAME: QEERI_oper
TASK NAME: t_run_wps
TASK FULL NAME: /QEERI_oper/t_run_wps
START TIME: 2015-10-21 09:38:20 UTC
```

Figura che mostra il preambolo di ciascun task

```
Summary of information about the run of this task
STOP TIME: 2015-10-21 09:38:22 UTC
TASK WALL TIME: 2 s
```

Figura che mostra la conclusione di ciascun task

La header e la tail sono incluse nel task al momento del pre processamento, ovvero della formazione del job che ecFlow esegue come task pertanto possono essere usate le variabili ambientali definite nella suite e quelle generate automaticamente da ecFlow, quest'ultime sono elencate nel manuale d'uso del workflow manger. Anche il manuale, che deve accompagnare ciascun task, viene incluso al momento del preprocessamento. Infine, per superare un limite derivante dal software che gestisce i job sottoposti alle code di calcolo sulla strutture HPC FENICE, è stata realizzata una apposita funzione che si occupa della sottomissione dei job alla coda e ne verifica lo status, facendo procedere il flusso solo al termine dell'esecuzione del lavoro. Tale funzione, qsub_4_bash.h, viene inclusa anch'essa in fase di pre processamento. La tipica sequenza di inclusioni di file da parte di ciascun task è riportata in figura 7.

5. Caratteristiche comuni a tutti i manuali

Tutti i task debbono essere dotati di un manuale che ne descrive il compito all'interno del workflow, oltre a riportare informazioni utili all'eventuale individuazione della fonte di problemi di esecuzione o alla soluzione degli stessi.

I testi dei file di manuale debbono necessariamente essere scritti entro le prime **80 colonne**. Non si tratta di una necessità connessa al loro utilizzo, piuttosto una prassi per standardizzare i contenuti e rendere agevole la lettura tramite ecFlowviewer.

Non esiste un formato standard per la costruzione del manuale. Operativamente il manuale è diviso in almeno due parti che vengono incluse separatamente. La prima riguarda informazioni generali sulla suite, la seconda è specifica del task. Comunque viene lasciato al progettista del task la scelta della forma migliore, per esempio, alcune sezioni del manuale potrebbero iniziare con dei caratteri speciali che indicano l'inizio e altrettanto carattere che ne indica la fine, per esempio / & o anche START e STOP.

La figura 8 mostra un esempio di manuale generale della suite, mentre le figure 9 e 10 mostrano i manuali specifici di due task. Il nome del file del manuale coincide con il nome del file del nodo salvo l'estensione .man, la quale è necessaria, almeno per i manuali delle famiglie e della suite. Per essere individuati e mostrati dall'applicativo ecflowview, i file contenenti i manuali delle famiglie o della suite debbono risiedere nella directory $ECF_FILES, mentre i manuali

dei task, essendo inclusi durante il preprocessing del nodo nel momento in cui viene caricato sul server, possono essere depositati ovunque. Generalmente sono depositati nella sottodirectory man della directory $ECF_INCLUDE. Nel caso di sotto famiglie di famiglie che hanno lo stesso nome, quindi che svolgono la stessa funzione ma in parti di flusso distinte, per esempio indicanti scadenze temporali, è possibile utilizzare le variabili generate dalla suite per caratterizzare il manuale utilizzando lo stesso file. Nello specifico le variabili %FAMILY% e %FAMILY1% presenti nel file step_048.man saranno sufficienti a personalizzare il manuale della famiglia modello_A/generazione/step_048 e quello della famiglia modello_B/graficazione/step_048 usando lo stesso file step_048.man depositato nella directory radice: $ECF_FILES.
L'utilizzo dei manuali è di fondamentale importanza per documentare il flusso, per guidare l'utente alla comprensione dello stesso e alla soluzione di eventuali problemi. Pertanto la stesura del manuale non deve essere considerata come un'attività accessoria da poter eventualmente trascurare.
Viene naturale raccogliere tutti i manuali nella sotto directory man della directory $ECF_INCLUDE. Sfortunatamente questa prassi necessita la costruzione di link simbolici che mettono a disposizione il file dei manuali delle famiglie e della suite nella directory $ECF_FILES. Non esiste altro modo che posizionare i file dei manuali delle famiglie e della suite nella $ECF_FILES affinché siano visualizzabili tramite ecflowview in corrispondenza dell'icona del nodo a cui si riferiscono.
Per evitare la selva dei link simbolici o perlomeno limitarla al solo caso del manuale generale della suite è possibile adottare la tecnica di includere i manuali di suite, famiglia ed eventuali sottofamiglie direttamente nel manuale del task, tramite l'operazione di include. In questo modo l'utente, qualsiasi sia il nodo di ultimo livello esplorato, avrà a disposizione i manuali delle famiglie e della suite a cui appartiene, secondo una logica a cascata da lui scelta nel momento in cui definisce l'inclusione dei manuali nella header del task. Lo svantaggio di questo approccio è che dall'interfaccia grafica ecflowview la suite e le famiglie paiono prive di manuale. Un compromesso potrebbe essere quello di linkare nella directory $ECF_FILES solo il manuale della suite e includere i manuali delle sue famiglie e sottofamiglie nei task ed è così che l'utente accede, alle informazioni generali della suite, nel punto logicamente corretto ed il numero di link presenti nella $ECF_FILES si riduce ad uno solo.

6. Glossario

BASH: interprete di comandi che permette all'utente di comunicare con il sistema operativo di un calcolatore attraverso funzioni predefinite o di eseguire programmi.

ECMWF: acronimo del Centro Europeo per le previsioni meteorologiche a medio termine, European Center for Medium-range Weather Forecasts.

FENICE: indica il sistema di calcolo ad alte performances di cui si tata ARPA FVG ed acronimo per Fvg ENhanced Infrastructure and Computational Environment.

HPC: indica strutture di calcolo ad alte performances High Performance Computing

YAML: linguaggio per la formattazione in forma seriale di dati che rettamente leggibile dalluomo (human readable) senza bisogno di codici che decodifichino i dati.

Sitografia e Bibliografia

[1] ECMWF. Workflow manager ecFlow. https://software.ecmwf.int/wiki/display/ECFLOW/Home.

[2] Goble C. and DeRoure D. The fourth paradigm. http://research.microsoft.com/en-us/collaboration-/fourthparadigm/ The impact of workflow tools on data-centric research, capitolo 3 pp. 137-154.

[3] Wikipedia. Workflow scientifici in WikiPedia. https://en.wikipedia.org/wiki/Scientific_workflow_system.

[4] D.Talia. Workflow Systems for Science: Concepts and Tools. http://www.hindawi.com/journals/isrn/2013/404525/ ISRN Software Engineering Volume (2013).

[5] UML. Unified Modeling Language. http://www.uml.org/.

[6] Violet UML Editor. http://alexdp.free.fr/violetumleditor-/page.php.

[7] Kepler. Kepler workflow manager. https://kepler-project.org/.

[8] PEGASUS WMS. Pegasus workflow manager system. https://pegasus.isi.edu/.

[9] Apache Taverna. Apache taverna. https://taverna.incubator.apache.org/.

[10] The Cooperative Computing Lab CCL. The makeflow workflow system. http://ccl.cse.nd.edu/software/makeflow/.

[11] ECMWF. European centre for medium-range weather forecasts. http://www.ecmwf.int/.

[12] YAML. Yaml. http://yaml.org/.

```
#!/bin/bash

set -e # stop the shell on first error
set -u # fail when using an undefined variable
#set -x # echo script lines as they are executed


# Defines the variables that are needed for any communication with ECF
export ECF_PORT=%ECF_PORT%    # The server port number
export ECF_NODE=%ECF_NODE%    # The name of ecf host that issued this task
export ECF_NAME=%ECF_NAME%    # The name of this current task
export ECF_PASS=%ECF_PASS%    # A unique password
export ECF_TRYNO=%ECF_TRYNO%  # Current try number of the task
export ECF_RID=$$             # record the process id. Also used for zombie detection


# Tell ecFlow we have started
ecflow_client --init=$$


# Define a error handler
ERROR() {
   set +e                     # Clear -e flag, so we don't fail
   wait                       # wait for background process to stop
   ecflow_client --abort=trap  # Notify ecFlow that something went wrong, using 'trap' as the reason
   trap 0                     # Remove the trap
   exit 0                     # End the script
}


# Trap any calls to exit and errors caught by the -e flag
trap ERROR 0


# Trap any signal that may cause the script to fail
trap '{ echo "Killed by a signal"; ERROR ; }' 1 2 3 4 5 6 7 8 10 12 13 15

# Get start time in a suitable format to be usefulf for task wall time computation
TASK_START_TIME=$(date -u +%%s)

# Some diagnostict at beginning of the task
echo -e "\n\t-------------------------------------------------------------------------------"
echo -e "\t|          Preliminary information about the run of this task           |"
echo -e "\t-------------------------------------------------------------------------------"
echo -e "\n\tSUITE NAME: %SUITE%\n\tTASK NAME: %TASK%\n\tTASK FULL NAME: %ECF_NAME%\n\tSTART TIME: $(date -u +%%F' '%%T' UTC')\n"
echo -e "\n\tSUITE DATE: %ECF_DATE%\n\tSUITE TIME: %ECF_TIME%\n\tSUITE CLOCK: %ECF_CLOCK%\n"
echo -e "\n\t-------------------------------------------------------------------------------"
echo -e "\t|                  Here below the stdout of this task                   |"
echo -e "\t-------------------------------------------------------------------------------"
#
# +-----------------------------------------------------+
# |                                                     |
# | HEADER COMPLETED - START THE TASK CORE HERE BELOW |
# |                                                     |
# +-----------------------------------------------------+
#
```

Figura 5. Esempio di header file che in incluso in tutti i task.

```
#
# +-----------------------------------------------------+
# |                                                     |
# | START TAIL HERE BELOW  - THE TASK CORE HERE ABOVE |
# |                                                     |
# +-----------------------------------------------------+

sleep %SLEEP%              # Wait SLEEP number of seconds before to
                           # consider the task completed

# Get end  time in a suitable format to be usefulf for task wall time computation
TASK_END_TIME=$(date -u +%%s)

# Some diagnostict at end of the task
echo -e "\n\t-------------------------------------------------------------------------------"
echo -e "\t|           Summary of information about the run of this task           |"
echo -e "\t-------------------------------------------------------------------------------"
echo -e "\n\\tSTOP TIME: $(date -u +%%F' '%%T' UTC')"
echo -e "\tTASK WALL TIME: $((${TASK_END_TIME}-${TASK_START_TIME})) s\n"

wait                       # wait for background process to stop
ecflow_client --complete   # Notify ecFlow of a normal end
trap 0                     # Remove all traps
exit 0                     # End the shell
```

Figura 6. Esempio di tail file che in incluso in tutti i task.

```
%manual
%include <man/general.man>                      # General manual for all task in this suite
%include <man/wrf_grib1/t_wrf2grib1.man>        # Specific manual for this task
%end

%include <head.h>                               # Include header common to all tasks.
                                                # It manages general calls to ecflow client

%include <qsub_4_bash.h>                        # Include the function managing the jobs
                                                # submission to the queue

#-------------------------------------------------------------------------------
#                        MAIN SCRIPT HERE BELOW
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
#                        MAIN SCRIPT HERE ABOVE
#-------------------------------------------------------------------------------

%include <tail.h>                       # Include tail common to all tasks.
                                        # It manages general calls to ecflow client
```

Figura 7. Esempio di task con la parte che include il manuale la header e la tail.

```
[operative@access tmp]$ cat ~/src/operative_workflows/operative/WRF_dis/include/man/general.man
   --------------------------------

   This is the manual of WRF operational suite

   --------------------------------

   Manual for:
          TASK:   %TASK%
          FAMILY: %FAMILY%
          SUITE:  %SUITE%

   Manual version: 0.0.1
   Last change:    Nov 06, 2015
   Change by:      Dario B. Giaiotti

   SUITE GOAL IS:
   This suite interpolates and extracts subsets of WRF model outputs. The data
   are then disseminated to the users.

   EXTENDED SUITE DESCRIPTION
   This suite interpolates the WRF model outputs, that are in netCDF format,
   and write the results in the required format, i.e. GRIB.
   Fields and domains are selectected according ecFlow environmental variables
   and initialization files. There are several sub families in this family, each
   one having many tasks.
   The sub families are related to specific exctraction and dissemination
   purposes.
   Dissemination may include the transfer of files by means of ncftp application

   The suite was written by:
          Dario B. Giaiotti
          ARPA FVG - CRMA
          Centro Regionale di Modellistica Ambientale
          Via Cairoli, 14
          I-33057 Palmanova (UD)
          ITALY
          Room I/20/U
          Tel +39 0432 191 8048
          Fax:+39 0432 191 8120
          Certified e-mail - PEC arpa@certregione.fvg.it
          e-mail dario.giaiotti@arpa.fvg.it
```

Figura 8. Esempio di manuale generale della suite

```
---------------------------------
| Specific manual for this task |
---------------------------------

Task manual version:        0.0.1
Task manual last change:    Nov 25, 2015
Change by:                  Dario B. Giaiotti

TASK GOAL IS:
To generate the user producs from WRF output files in GRIB1 format files.


IF THIS TASK FAILS, WHAT TO DO:
You must check the WRF ouput file existence, then the initialization files
concistency. There are two initialization files you have to look at:

  a) the suite initialization file, which is hosted in the root suite
     directory and its name is suite_name.ini

  b) the user initialization file, which is hosted in the etc directory
     of the WRF repository (i.e. WRF/WRF4CRMA/wrf4oper/etc/) and its name
     is wrf_4_%GRIB1_USER%_disse.ini, where %GRIB1_USER% is the
     identification name on the user.

The stdout of this task reports all the steps and the variables used to
get its objective, so going through it you should be able to find why it
has failed.

There are no automatic switches for the task in case of it fails to run.

EXTENDED TASK DESCRIPTION
The task performes the following actions:

1) it loads the suite initialization file. In this file there are the
   the features of the WRF output files identification, the PBS directives
   for jobs submission and the informatiomn and templates how to find the
   user initialization file;

2) it loads the user initialization file. In this file there are all the
   details of the products to be generated and to be disseminated;

3) it defines the template job suitable to run the unipost script
   (i.e. UNIPOST_JOB_TPL);
   it defines the working directory for the job;
   it defines the directory where to archive the GRIB outputs;
   it generates the crma-run_unipost.sh initialization file;
   it generates the job script to run crma-run_unipost.sh from template
   (i.e. UNIPOST_JOB_TPL);
   it submit the job script producing the user products ib GRIB1 format;
   it archives the GRIB1 files;

The task loops over the domains defined for the users and for each domain
the products are selected according to specific initialization files
which are defined in the user initialization file and are those expected by
unipost UPP software.

For each domain, in the user initialization file, there is a switch that
allows the GRIB fields to be reinterpolated in Lat/Lon regural grid or to
be left in the coordinate system of the original WRF outputs.
```

Figura 9. Esempio di manuale specifico di un task

```
---------------------------------
| Specific manual for this task |
---------------------------------

Task manual version:        0.0.1
Task manual last change:    Nov 25, 2015
Change by:                  Dario B. Giaiotti

TASK GOAL IS:
To transfer the user producs to the data exchange areas by means of
ncftp utility.


IF THIS TASK FAILS, WHAT TO DO:
You must check the WRF ouput file existence, then the initialization files
concistency. There are two initialization files you have to look at:

  a) the suite initialization file, which is hosted in the root suite
     directory and its name is suite_name.ini

  b) the user initialization file, which is hosted in the etc directory
     of the WRF repository (i.e. WRF/WRF4CRMA/wrf4oper/etc/) and its name
     is wrf_4_%GRIB1_USER%_disse.ini, where %GRIB1_USER% is the
     identification name on the user.

The stdout of this task reports all the steps and the variables used to
get its objective, so going through it you should be able to find why it
has failed.

There are no automatic switches for the task in case of it fails to run.

EXTENDED TASK DESCRIPTION
The task performes the following actions:

1) it gets the suite defined variables:
      GRIB1_USER --> the user identification string (e.g. 'OSMER')
      FTP_CLEAN  --> the flag for clening the remote area before the upload
                     'n' = No cleaning
                     'y' = Cleaning
      FTP_COMPL  --> the flag to allow the script to upload the FTP status
                     file. If the upload is allowed, then the file reports
                     the "in progress status" which is put in the file name
                     $FTP_STATUS_FILENAME of user initialization file,
                     otherwise it is pu in defatul file: ftp_status.txt
                     The status is in progress set just before to start the
                     upload, or after the cleaning, when the cleaning is
                     activated. The status is set completed after the
                     successfull file transfer.
                     'n' = No FTP status file upload
                     'y' = upload the FTP status


2) it checks the existence of the the initialization files and the it reads
   the variables and the parameters required for file transfer, i. e. user,
   passwd, area, etc.;


3) it looks for the files to be transferred making a list according the
   following criterium:
         ls ${ARCHIVE_DIR}${DIS_OUTPUT_TEMPLATE}*${FILE_EXTESION});
   at present the file extension FILE_EXTESION is fixed in the script
   each file is checked if already transferred. The already transferred file
   list is created when not available yet, or it is updated appending the
   list of files already successfully transferred. The file name is created
   in this task and it is:
       TRANS_LIST_FILE="${CHECK_JOB_LOG}/${GRIB1_USER}_transferred.list"
                                |
                                +---- This is the log directory
   The a temporary list of not already transfer files is generated;

4) each file in the temporary list is transferred by means of ncftpput
   utility. At the end, if the transfer was OK the file name is appended to
   already transferred file list; if the file was not successfully transfered
   the it is not added;

5) in case there are files not successfully transferred then the script goes
   again to point 3) and following the next point 4) arttempts to transfer
   the left files. The maximum number of trial before the leave files not
   transferred is set in the user initialization file $FTP_TRANSFER_RETRY_N.

Many other parameters are defined in the initialization files. Almost all of
then are written in the stdout at the beginning of this task execution, so
you can read them in the stdout log file.

This task was written by:
       Dario B. Giaiotti
       ARPA FVG - CRMA
       Centro Regionale di Modellistica Ambientale
       Via Cairoli, 14
       I-33057 Palmanova (UD)
       ITALY
       Room I/20/U
       Tel +39 0432 191 8048
       Fax:+39 0432 191 8120
       Certified e-mail - PEC arpa@certregione.fvg.it
       e-mail dario.giaiotti@arpa.fvg.it
```

Figura 10. Esempio di manuale specifico di un task

ARPA FVG-Aria-Elaborati tecnico-scientifici