Uso del workflow manager ecFlow al CRMA dell'ARPA FVG

Descrizione generale e breve corso all'uso

Seminario interno di ARPA FVG

Palmanova
07 giugno 2017

Dario B. Giaiotti

ARPA FVG – CRMA
Centro Regionale di Modellistica Ambientale
crma@arpa.fvg.it

Riprendiamo da dove eravamo rimasti

Il 21 aprile 2015 al termine della presentazione che introduceva ecFlow tra gli strumenti di gestione dei flussi computazionali al CRMA, citavo la celebre frase pronunciata dall'amico Stelio Montebugnoli a proposito delle luci di Hessdalen (https://it.wikipedia.org/wiki/Luci_di_Hessdalen), prima della campagna di misura con il mulinello di campo elettrico.

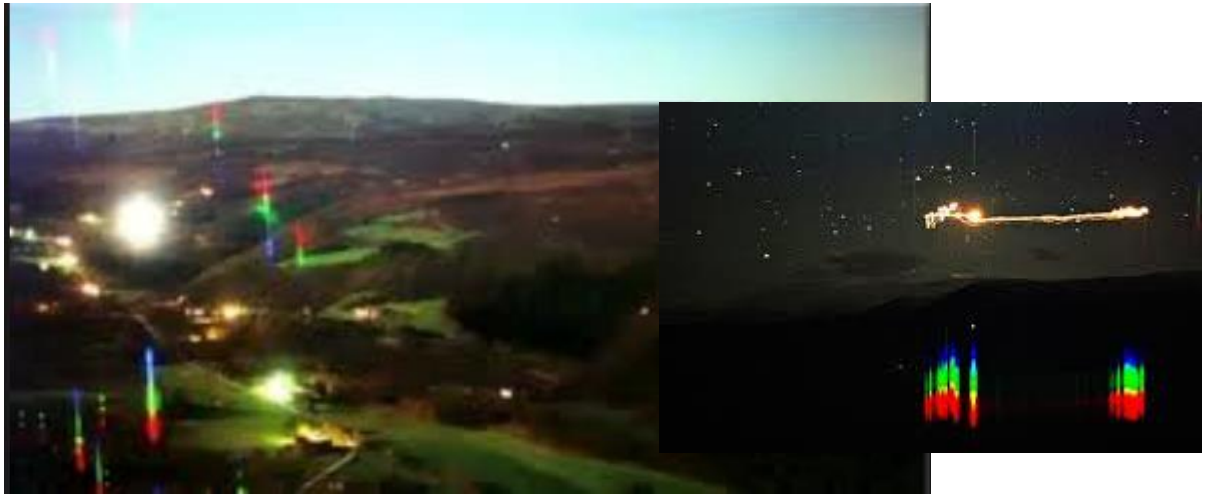
ecFlow è uno degli elementi (ambiente software) per:

- ❑ Sviluppo di catene (suite) modellistiche
- ❑ Sviluppo di catene (suite) di elaborazioni intensive di dati
- ❑ Sviluppo di catene (suite) per la diffusione dei dati
- ❑ Gestione dell'operatività

CRMA

Questa volta li prendiamo …. e li abbiamo presi.

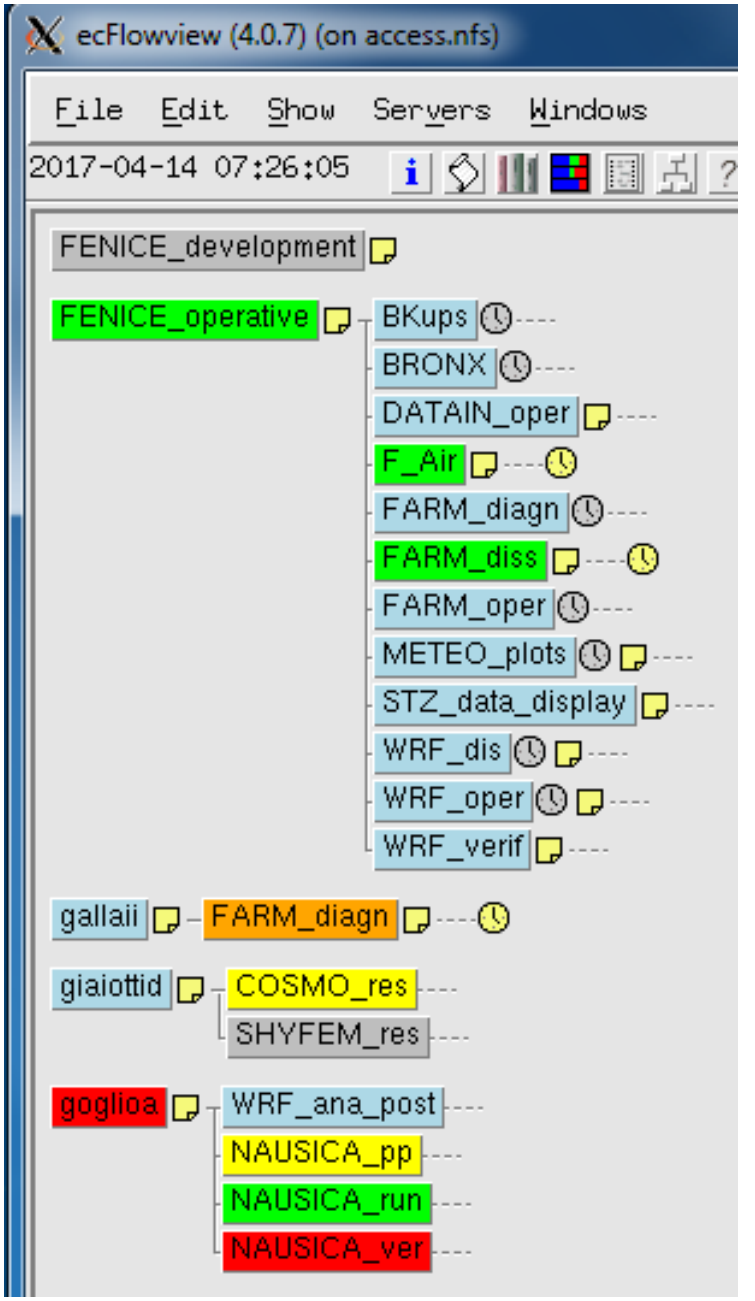
Norway's Hessdalen Lights.



Dr. Stelio Montebugnoli

Oggi (03/2017) tutti i flussi operativi CRMA sono in ecFlow

- **12** flussi operativi (suite) costantemente eseguite una o più volte al giorno
- **N** flussi personali



I flussi operativi svolgono i seguenti compiti

- Backups ed archiviazioni
- Analisi e display misure qualità dell'aria
- Analisi e display misure meteo
- Analisi e display misure marine
- Acquisizione dati
- Simulazioni meteorologiche prognosi
- Simulazioni meteorologiche diagnosi
- Simulazioni qualità dell'aria prognosi
- Simulazioni qualità dell'aria diagnosi
- Post elaborazioni dati modelli
- Verifica qualità simulazioni modellistiche
- Distribuzione dati e grafici su aree FTP
- Pubblicazione dati e grafici sul web agenziale

Esecuzione di codici HPC, semplici script e applicativi di diversa complessità usando nodi di calcolo e nodo di accesso.

Sommario del seminario

Parte I - Per tutti (45')

- Descrizione dei workflow manager
- Benefici e costi dell'uso del workflow manager
- Descrizione del workflow manager ecFlow
- Descrizione dell'implementazione e dell'uso di ecFlow al CRMA

Pausa

Parte II - Per gli utenti della FENICE interessati all'uso (60')

- Dettagli tecnici su come realizzare una suite ecFlow ed eseguirla sulla FENICE
- Esempi di sviluppo ed esecuzione di suite

Parte I

- Descrizione dei workflow manager
- Benefici e costi dell'uso del workflow manager
- Descrizione del workflow manager ecFlow
- Descrizione dell'implementazione e dell'uso di ecFlow al CRMA

Cos'è un workflow manager per scopi scientifici

Il concetto nasce in ambito gestionale (gestione dei processi aziendali)

La teoria e le applicazioni del **workflow management** considerano la gestione dei **gruppi di lavoro collaborativi** secondo il *workflow model*, **modello processuale**. Un processo consiste in una o più **attività** ognuna delle quali **rappresenta un lavoro da svolgere** per giungere a un obiettivo comune. Il workflow management sostiene l'organizzazione del processo di lavoro mediante l'utilizzo di software specifici. Le attività possono essere svolte dai partecipanti o da applicazioni informatiche.

(fonte Wikipedia http://it.wikipedia.org/wiki/Workflow_management)

L'applicazione per scopi scientifici mutua l'esperienza in ambito gestionale

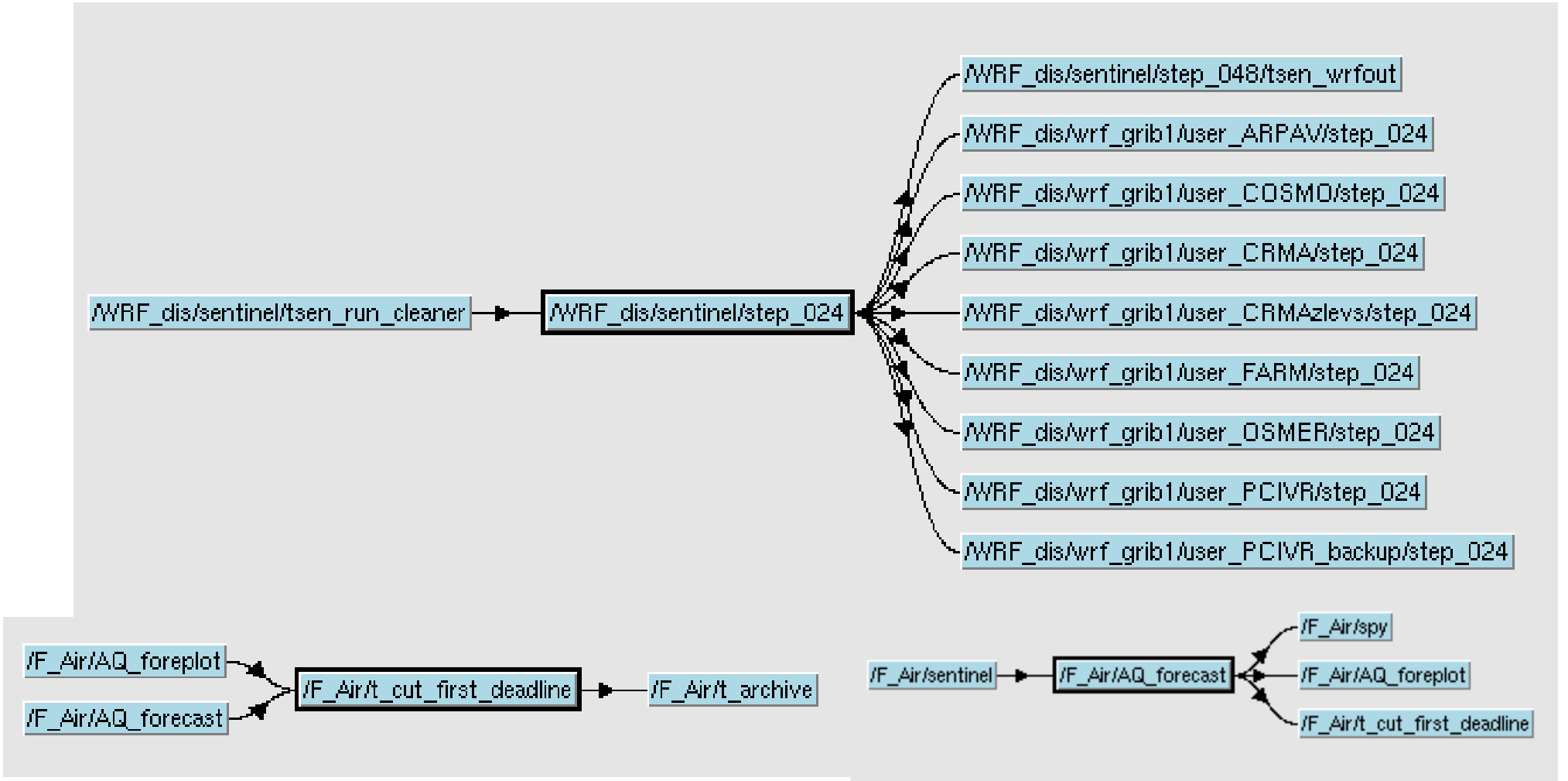
I workflow manager sono software che gestiscono l'esecuzione di applicativi e i flussi di dati secondo la logica ed i principi del **workflow management**, nell'era della data-centric scientific research(1) e del products-services generation(2)

(1) *The Impact of Workflow Tools on Data-centric Research in "The Fourth Paradigm: Data-Intensive Scientific Discovery",* S. Tansley, K. M. Tolle, Microsoft Research, 2009

(2) ARPA FVG - Centro Regionale di Modellistica Ambientale - from working expertise

Quando usare un workflow manager

Eseguire **ripetutamente compiti** di acquisizioni dati, validazioni, elaborazioni, esecuzione di applicativi, archiviazioni, pulizie di aree disco, ecc. che sono **dipendenti gli uni dagli altri e dal manifestarsi di particolari eventi**.



Componenti fondamentali di un workflow manager

- Piattaforma esecutiva (un sistema client-server)
- Strumenti di sviluppo (applicativi, funzioni e un linguaggio di definizione del flusso)
- Interfaccia grafica gestionale (non sempre disponibile)

Esempi di software che svolge il compito di workflow manager sono:

ecFlow	https://software.ecmwf.int/wiki/display/ECFLOW/
Kepler	https://kepler-project.org/
Pegasus	https://pegasus.isi.edu/
Taverna	http://www.taverna.org.uk/
Makeflow	http://ccl.cse.nd.edu/software/makeflow/
AiiDA	http://www.aiida.net/feature/data-provenance/
Quartz	http://www.quartz-scheduler.org/

Benefici nell'uso di workflow manager per scopi scientifici

Uso del flusso

- Rieseguire lo stesso flusso di compiti su basi dati e con applicativi diversi.
- Clonare il flusso e adattarlo a nuove esigenze computazionali.
- Condividere il flusso, o parti di flusso, tra utenti.
- Verificare lo stato di avanzamento del flusso su interfaccia grafica (visivamente).
- Agire sul flusso tramite interfaccia grafica.
- Risolvere problemi di esecuzione tramite interfaccia grafica.
- Tracciare ogni flusso e documentarlo, ovvero garantire la data provenance (1).

Efficienza nell'implementare flussi di compiti ed eseguirli

Riduzione dei tecnicismi e concentrazione sui compiti

Realizzazione del flusso

- Partecipazione di più sviluppatori allo sviluppo di un medesimo flusso (sistema collaborativo).
- Se abbinato ad un applicativo per il versionamento dei software (Git), tracciabilità degli sviluppi e delle clonazioni del flusso.
- Se abbinato con un sistema collaborativo di sviluppo (TRAC), tracciabilità della progettazione e dello sviluppo del flusso.
- Se abbinato con un sistema di verifica automatica (Jenkins), standardizzazione dei flussi e riduzione del rischio di fallimento per motivi tecnici.

(1) Data provenance è la capacità di ricostruire la storia completa di tutti i passaggi di un flusso di calcolo e della gestione di dati che producono un risultato scientifico.

Costi dell'uso di workflow manager per scopi scientifici

Uso del flusso

- Apprendimento sull'installazione e la manutenzione del software
- Apprendimento della logica e della sintassi nell'uso a comando di linea
- Apprendimento delle funzionalità dell'interfaccia grafica
- Apprendimento del formato e delle tecniche usate per la data provenance

Tempo impiegato in formazione

Realizzazione del flusso

- Progettazione del flusso secondo la logica del workflow manager.
- Coordinamento tra gli sviluppatori alla stesura di un medesimo flusso.
- Se abbinato ad un applicativo per il versionamento dei software (Git), versionamento regolare delle modifiche apportate al flusso.
- Se abbinato con un sistema collaborativo di sviluppo (TRAC), documentazione regolare delle modifiche apportate al flusso.
- Se abbinato con un sistema di verifica automatica (Jenkins), implementazione dei test.

Tempo impiegato per la realizzazione

Tempo impiegato a supporto della condivisione e della verifica

Motivi che hanno portato alla scelta di ecFlow

Valutazione iniziata nel 2013-2014 con questi criteri da soddisfare

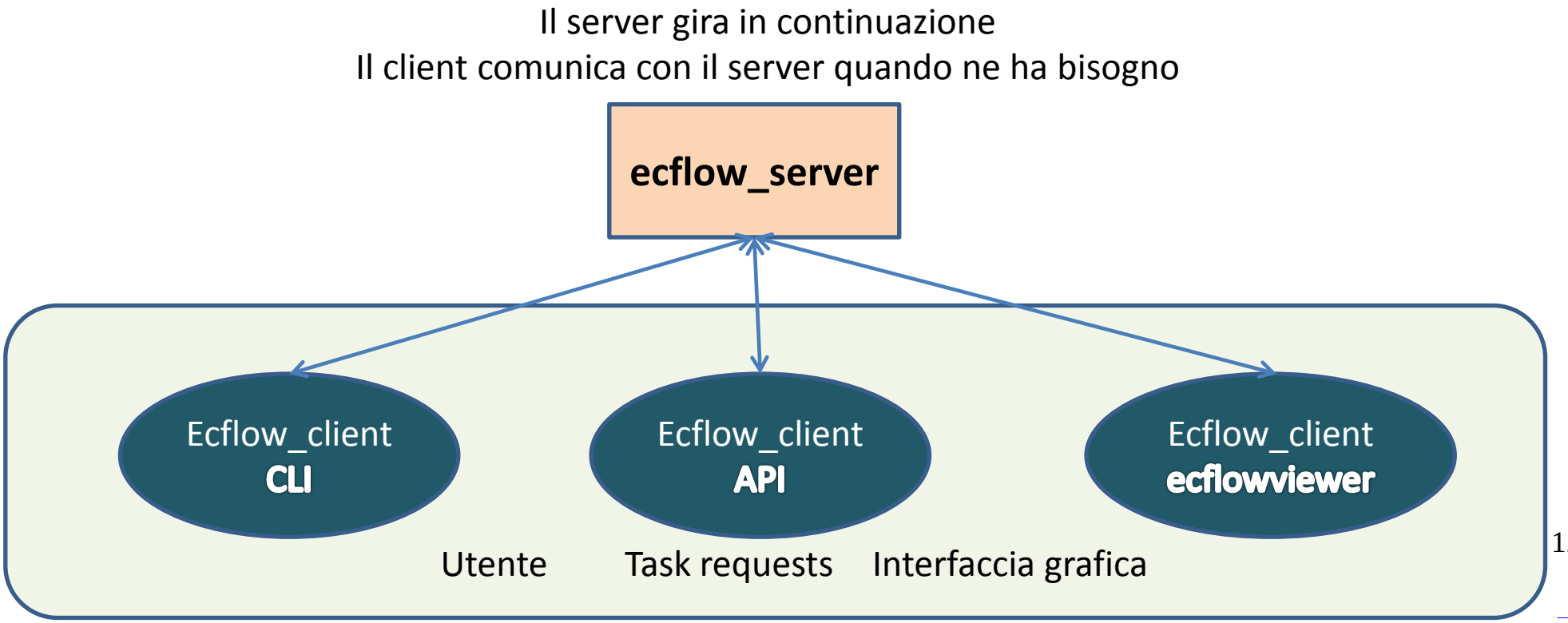
- Software libero, non a pagamento.
- Certezza del mantenimento informatico del software.
- Supporto informatico competente per un periodo di anni sufficientemente lungo da essere confrontabile con i tempi evolutivi della modalità di gestione dei flussi computazionali del CRMA;
- Software adottato operativamente almeno da una struttura tecnico scientifica che si occupa di modellistica ambientale ad alto livello su tipi di flussi simili, se non uguali, a quelli che il CRMA gestisce e gestirà in futuro.
- Autonomia gestionale del software: installazione, gestione e aggiornamenti.

Informazioni e metodo usato per la scelta del workflow manager

- Esperienza derivante dai flussi computazionali operativi implementati dai modellisti del CRMA, sino al 2014, prevalentemente tramite scripting BASH.
- Osservazione dei workflow manager usati in altri enti tecnico-scientifici.
- Ricognizione sull'uso di workflow managers presso infrastrutture HPC – richieste dirette tramite e-mail o telefonate. (CINECA, ECMWF, NCAR-Wyoming Supercomputing Center, San Diego Supercomputing Center, PRACE-Partnership for Advanced Computing in Europe, Lawrence Livermore National Laboratory)

Com'è fatto ecFlow?

- È un software formato da un **server** e da un **client**
- Scritto in linguaggio C++
- Utilizzabile tramite:
  - CLI (Command Level Interface)
  - API (Application Programming Interface) in Python
  - Interfaccia grafica (ecflowviewer)



Installazione e gestione del server

L'installazione del server non è difficile, ma richiede un minimo di attenzione

Si deve individuare l'HOST, la porta, alcune variabili d'ambiente essenziali per la gestione ordinaria e di eventuali problemi.

Sulla FENICE, l'utente che si occupa dell'installazione e della manutenzione è **operative**.
Ci sono dei moduli che quando caricati permettono di utilizzare con facilità il client e l'interfaccia grafica. Attualmente sono previsti due server operativi e potenzialmente un server per utente :

Server name	module name	purposes
Operative	oper_ecflow/4.0.6(default)	solo **suite operative**
Development	devel_ecflow/4.0.6(default)	solo **suite operative in sviluppo**
Users	user_ecflow/4.0.6(default)	tutte le altre suite **non operative**

Dalla documentazione distribuita dagli sviluppatori di ecFlow:

- Designed for collaborative working so default access is open
- ecflow server can be protected with white list file
  - ecf.lists
- We use specific accounts for operations and research
- Some jobs are submitted for another user: careful with job-file owner, output file owner, rsh/ssh settings, queueing system permissions
- Never run as root!
- And finally
- Do not even think about running as root!

Dove trovare la documentazione

Tutto quello che serve, manuali, tutorial, ecc. è disponibile nell'apposita pagina web mantenuta da ECMWF: https://software.ecmwf.int/wiki/display/ECFLOW/Documentation

Inoltre ecflow ha dei chiari help da linea di comando che sono molto utili per l'utilizzo tramite CLI.

Per quanto riguarda le caratteristiche specifiche dell'installazione e dell'uso per gli scopi del CRMA, è disponibile:

- un'apposita **pagina wikiCRMA** http://ms05lxarpa.arpa.fvg.it/wiki/index.php/EcFlow_al_CRMA
- un **articolo tecnico-scientifico** Art. Tec-Sci No. 001/2017 (07/06/2017) ARPA FVG – CRMA (arpaweb)
- Il repository remoto Git **operative_workflows** dal quale attingere esempi

Cosa si è tenuti a sapere:

- **l'utente operative** deve sapere **tutto o quasi** sul server, sul client e sulle reciproche interazioni.
- un **utente che esegue e monitora** lavori deve conoscere i **comandi del client** e l'interfaccia grafica **ecflowviewer.**
- un **utente** che contribuisce alla **costruzione di task** da sottoporre ad ecflow, deve conoscere i **comandi del client** e l'interfaccia grafica **ecflowviewer,** inoltre deve seguire **le prescrizioni** per l'assolvimento del proprio compito, ovvero la scrittura di parti di codice.

Cosa può fare ecFlow

ECFLOW permette di eseguire un elevato numero di applicativi interdipendenti e dipendenti dal tempo in un ambiente computazionale controllato.

- Flexible inter-dependencies between tasks
  - triggering
- Complex automated scheduling
  - based on events, times, task status
  - Multiple users / platforms / queues
- Access to monitoring information via GUI and CLI
  - ecflowview and API
- Dynamic and interactive supervision in real time
- Good recovery - at task and ecflow level
- But … ecflow is not a queuing system, it is a scheduler

Elemento essenziale del flusso in ecFlow

Tutti i componenti del flusso che si occupano di eseguire azioni sono chiamati **nodi**. I nodi si distinguono in: **famiglie** e **task**
L'elemento atomico dei lavori da svolgere è il **task.** A ciascun **task** che corrisponde un file.
Le **famiglie** sono insiemi di nodi.

Il task:

- ❑ esegue un lavoro
- ❑ comunica al server il suo stato o eventuali problemi
- ❑ riceve dal server ordini su quanto eseguire il lavoro

File che identifica il task (.ecf)

Come è fatto un task

È uno script come quelli UNIX (shell)

- Il nome del file ha estensione **.ecf**

Prevede sempre:

- Una **header** (direttiva di pre processing)
- Una **tail** (direttive di post processing)
- Sezioni dedicate al **manuale** del task
- **Comandi di shell (BASH, KSH) ma potrebbe essere anche Python o altro**

/home/operative/tmp/ecflow_trial/oper_flow/family_four/y2.ecf

```
%manual
   Manual for task y2
   Operations: if this task fails, set it to complete and report next working day
   Analyst:    Check something ?
%end
%manual
   Here is something that you shoud read carefully:

   +-------------------------------------------+
   |  When viewed they are simply concatenated. |
   +-------------------------------------------+

%include <oper_flow.man>
%end

%include <head.h>

echo "I am part of a suite that lives in %ECF_HOME%  h1"
echo -e "\n\tI have inherited varibale MYNAME=%MYNAME%        \n\n"
echo "\t\tI AM TASK 2\n\n"
eval /home/operative/tmp/ecflow_trial/oper_flow/submit_sge-job.sh
EXIT_STAUS=$?
echo -e "\n\tExecution completed with EXIT_STAUS=${EXIT_STAUS}\n\n"

%include <tail.h>

%manual
   +-------------------------------------------+
   There can be multiple manual pages in the same file.
   When viewed they are simply concatenated.
   +-------------------------------------------+
%end
```

Elementi essenziali del prologo (header)

Serve per:

Definire il tipo di interprete usato

Gestire la comunicazione con il server

Gestire gli errori di esecuzione dello script

Rendere disponibili variabili d'ambiente al task

operative@nexus:~/tmp/ecflow_trial (on nexus.arpa.fvg.it)

```
#!/bin/ksh

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


# Define the path where to find ecflow_client
# make sure client and server use the *same* version.
# Important when there are multiple versions of ecFlow
export PATH=/home/operative/apps/ecflow/ecflow.4.0.6/bin/:$PATH


# 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

#
# +-------------------------------------------------+
# |                                                 |
# | HEADER COMPLETED - START THE TASK CORE HERE BELOW |
# |                                                 |
# +-------------------------------------------------+
#
```

Elementi essenziali del epilogo (tail)

Serve per:

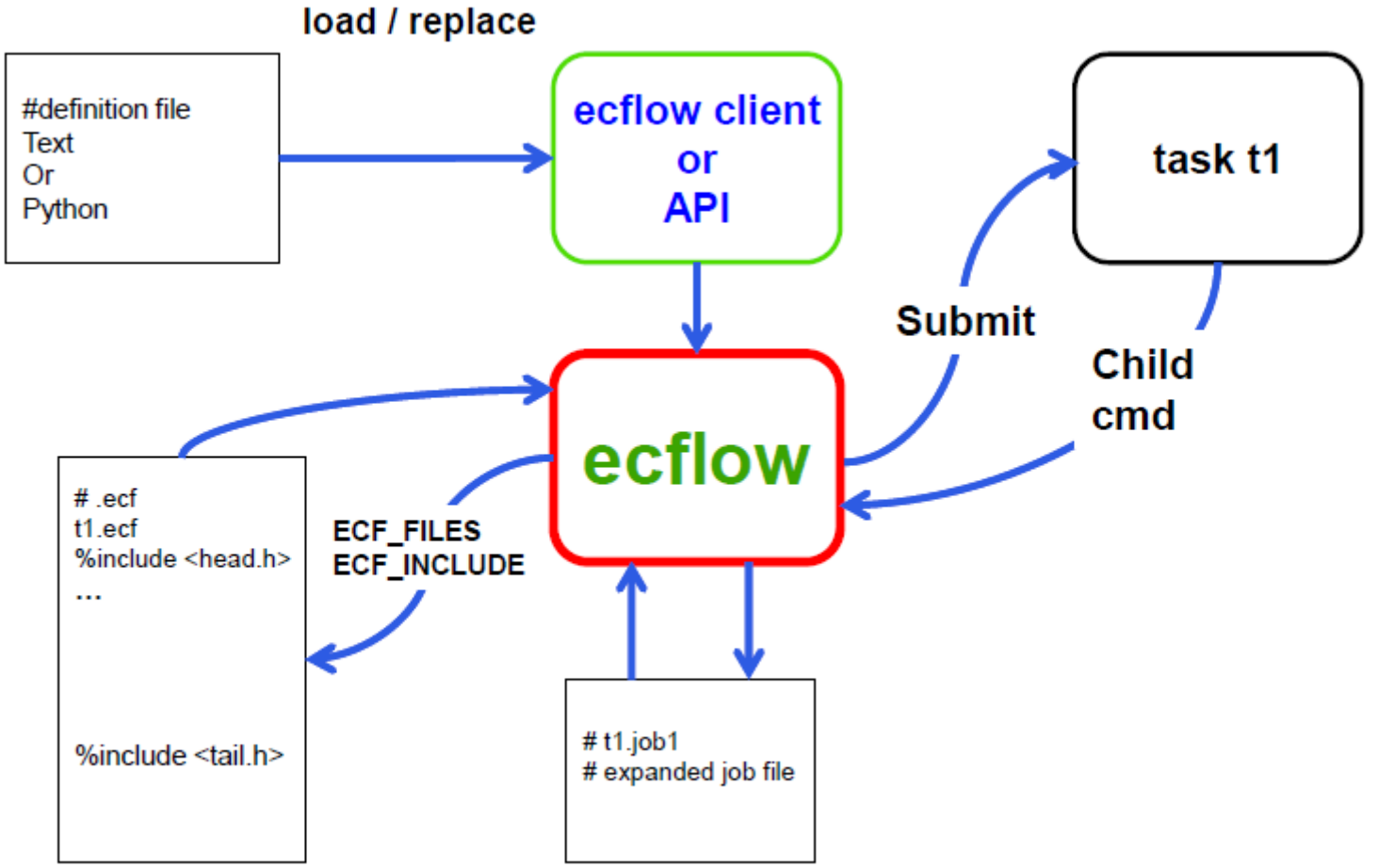
- Gestire la comunicazione con il server
- Diagnostica sull'esecuzione del compito
- Gestire gli errori di esecuzione dello script

```
#
# +-------------------------------------------------+
# |                                                 |
# | 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
```

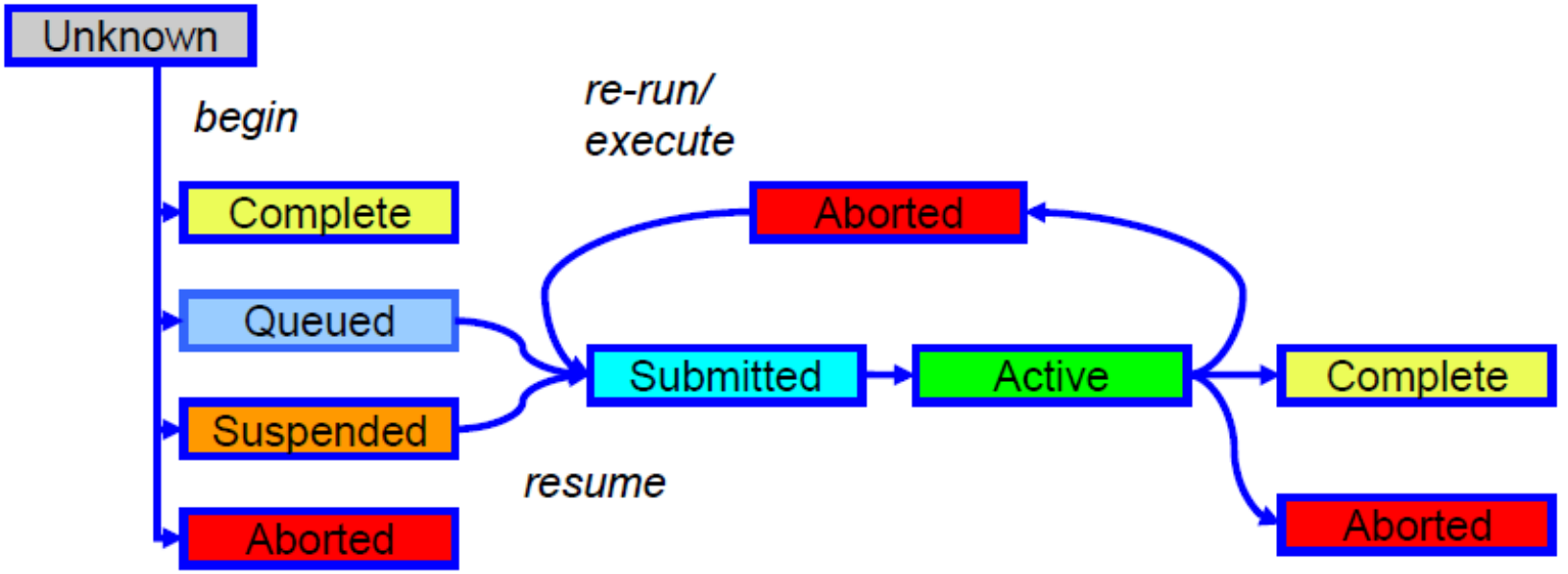
Le fasi della vita di un task

Quanto un task viene preso in carico da ecFlow accade quanto segue:

1) .ecf viene preprocessato (include, sostituzione ed export di variabili, ecc.)
2) il task viene trasformato in uno (SHELL) script da ecflow (.job)
3) Lo scripts, chiamato job, viene eseguito



Gli stati possibili di un nodo



Unknown	Least significant
Complete	
Queued	
Submitted	
Active	
Suspended	Will hide the underlying status
Aborted	Most important for a task, family or a suite
Shutdown	Only for Server
Halted	Most important, Only for Server

La suite, ovvero l'insieme dei task del workflow

I task appartengono sempre ad una suite che descrive il workflow e le interazioni tra i task.

La suite è:

- un file di testo;
- ha sintassi propria;
- può essere scritta con un editor oppure tramite le API Python, o altro (BASH);
- al file è associata una directory che contiene i tasks

```
# Definition of the suite test
suite test
   edit ECF_HOME /home/operative/tmp/ecflow_trial  # replace '$HOME' with the path to your home directory
   edit ECF_INCLUDE /home/operative/tmp/ecflow_trial/$SUITE/include  # The include default directory
   task t1
   task t2
        trigger /test/t3==complete
   task t3
endsuite
~
```

La suite viene caricata dal client e successivamente i task sono resi disponibili al server per le operazioni successive

ecflow_client --load=test.def

Alcuni aspetti importanti della definizione della suite

Nella definizione della suite sono contenute **tutte le caratteristiche del workflow**:

- Variabili da esportare a tutto il flusso
- Raggruppamenti di task in famiglie
- Dipendenze tra task e famiglie
- Dipendenze temporali dei task, delle famiglie e della suite
- Limiti all'esecuzione concorrenziale dei task
- Numero di volte di ri-esecuzione dei task abortiti
- Notifiche utili al monitoraggio visuale del flusso

```
# Definition of the suite test
suite oper_flow
   edit ECF_HOME     "/home/operative/tmp/ecflow_trial"
   edit ECF_INCLUDE  /home/operative/tmp/ecflow_trial/$SUITE/include
   edit ECF_FILES    /home/operative/tmp/ecflow_trial/test
   edit SLEEP '2'

   edit MYNAME "NEXUS OPERATIVE ecFlow"
#    cron 07:00 19:00 00:30
#
#  Define limits for tasks in suites
#
#
   limit f4 3  # Limit for tasks in family_four

   task s2
        trigger /oper_flow/family_one==complete
   task s3
         meter progress 1 20  15
   family family_one
          task f1
               trigger  /oper_flow/s3:progress==10
          task f2
               trigger  /oper_flow/s3:progress==18
   endfamily
   family family_two
          task g1
               label info "Some mews"
               event a
               event f1_ko
          task g2
               trigger g1:a
               complete g1:f1_ko
          task g3
               trigger g1:f1_ko
          task g4
               trigger /oper_flow/family_one==complete
   endfamily
#
#  This family is to test the labels, repeat
#
   family family_three
          repeat integer VALUE 1 5
          task h1
               repeat string DATE alpha beta gamma delta
               label goon ""
   endfamily
#
#  This family is to test the limiting action on tasks
#
   family family_four
          inlimit f4
          task y1
          task y2
          task y3
          task y4
          task y5
          task w1
          task y6
          task y7
          task y8
"oper_flow.def" 65L, 1835C
```

Come conviene lavorare con ecFlow

- Scrivere il flusso di task da eseguire e le loro dipendenze.
- Spezzare i compiti in task piccoli e quanto più generali o generalizzabili: applicare il metodo KISS (**K**eep **I**t **S**mall and **S**imple)
- Implementare i task con chiamate a SHELL script, applicativi, sottomissione di job alle code di calcolo i quali escono in modo controllato, 0, 1 (2)

0 - Eseguito come atteso | 1 - Errore inatteso e non gestibile | 2 - Errore atteso e gestibile

Designing an operational suite - considerations

- Critical path - minimise dependencies systems/file systems
- Documentation - man pages for suites/families/tasks
- Rerunnabililty of tasks
- Simplicity - KISS
- Keep runtimes under control
- Keep logfiles for support/optimisation
- Make/rebuild within suite plus admin tasks
- Allow for simple switching of systems
- Clean up

Alcune regole per la stesura dei task

Buona parte sarà scaricato sui wrapper: header e tail, oltre che nella definizione della suite

Writing "operational" scripts - considerations for critical tasks

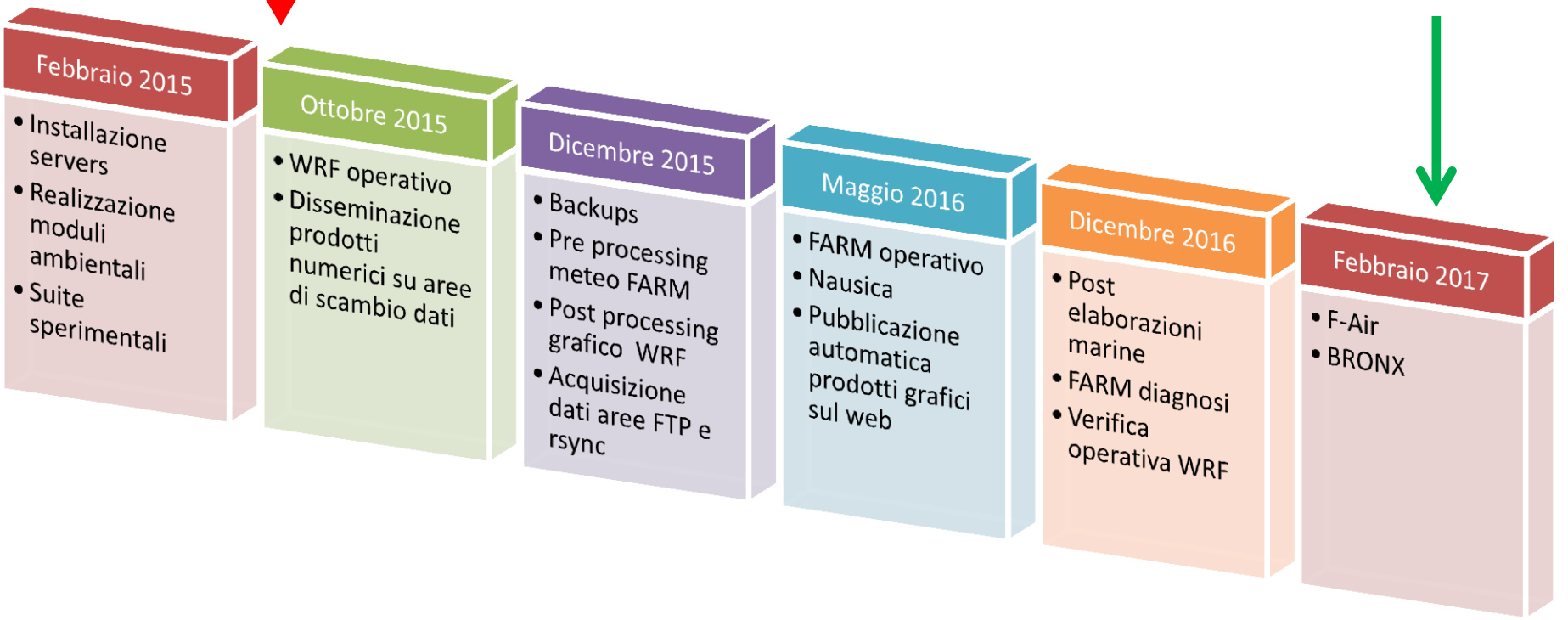
- Re-runnability
- Look after critical data - HA systems, backups
- Limit number of languages used
- Be careful with error trapping
- All variables should to be set (use default values %VAR:1%)
- Use a generic user - identify operations
- Works on multiple systems
  - ECF_JOB_CMD
- Design based on constraints
  - Staff criticality
- Avoid accessing off-line data in critical path
- Avoid NFS mounted files or unsafe file-systems (SCRATCH)
- Tasks can be serial or parallel
  - don't do serial things in parallel tasks
- Use generic directories to simplify cleaning and always clean up!
- Check task runtimes
- Keep output and job files
- Good to use a CM system and test
  - Test ecflow server/suites

Cronologia delle migrazioni dei flussi operativi

ecFlow è stato utilizzato sempre e solo sulla FENICE

Crash di NEXUS
06/08/2015

Tutta l'operatività sotto ecFlow + business continuity



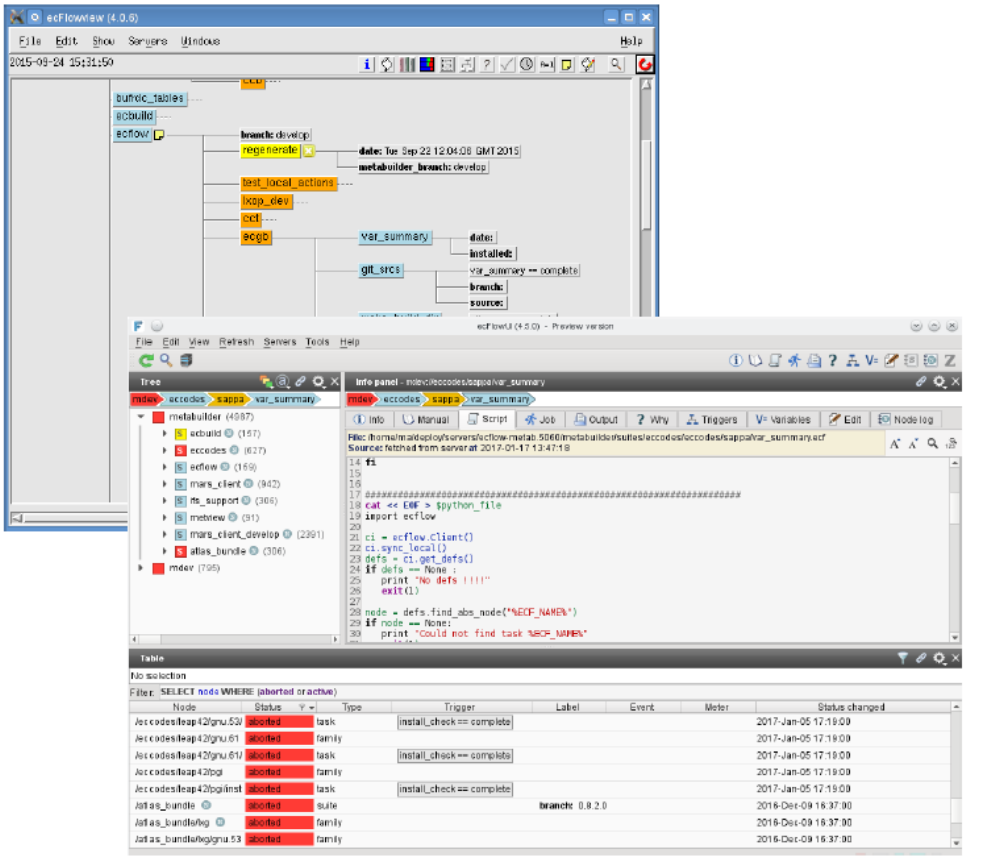
Dal 2017, ogni compito considerato operativo sarà sviluppato e gestito tramite ecFlow

Prossimi sviluppi al CRMA

- Aggiornamento alla versione 4.5.0 (attualmente 4.0.7 – siamo 5 versioni più vecchi)
- Introduzione di ecFlowUI
- Spegnimento e accensione automatica dei server con ricarica delle suite.
- Analisi con reportistica e backup automatici di tutte le informazioni relative al flusso
- Estensione, o sostituzione, della funzione ecflow_qsub con lo scopo anche, o solo, di monitorare le risorse usate (Dipende dall'upgrade del sistema di gestione delle code di calcolo della FENICE)

Transition from ecflowview

- **ecflowview** is the original user interface for ecFlow
- **ecFlowUI** is becoming its replacement
- Should be easier to use
- Should be more responsive
- Should have lower memory usage



Parte II

- Dettagli tecnici su come realizzare una suite ecFlow ed eseguirla sulla FENICE

- Esempi di sviluppo ed esecuzione di suite

Passo 1: il server

Prima cosa: avere un server ecFlow attivo

- Sulla FENICE esistono dei moduli ambientali che aiutano gli utenti

Per tutti gli utenti della FENICE: prima caricare la suite di moduli **arpa**, poi **user_ecflow**

```
[giaiottid@access ~]$ module load arpa
module path /opt/arpa/modules added, type 'module avail'
[giaiottid@access ~]$ module avail
```

```
---------- /opt/arpa/modules ----------
aria_reg/1.7.0              shyfem/6.1.62(default)
aria_reg/1.8.0(default)     user_ecflow/4.0.7(default)
aria_reg/1.9.0
```

Per l'utente operative sono disponibili tre moduli: **oper_ecflow**, **devel_ecflow**, **user_ecflow**

```
[operative@access tmp]$ module avail

---------- /u/arpa/operative/oper_mods ----------
devel_ecflow/4.0.7(default) oper_ecflow/4.0.7(default)  user_ecflow/4.0.7(default)
```

1. Caricare il modulo adatto

```
[giaiottid@access tmp]$ module load arpa user_ecflow
module path /opt/arpa/modules added, type 'module avail'


        The module: user_ecflow/4.0.7 is going to be loaded.
        Extended name: Workflow manager ecFlow
        Version: 4.0.7


        Environment suitable for: FOR USER SUITES

        You are the user: giaiottid
        Your ECF_PORT is: 2102
```

Porta unica per ciascun utente

Passo 1: il server (cont.)

2. Verificare se il server è attivo – la descrizione del modulo dà delle istruzioni
3. Se il server non è attivo, attivarlo

```
[giaiottid@access tmp]$ module show user_ecflow
-------------------------------------------------------------------
/opt/arpa/modules/user_ecflow/4.0.7:

                +-------------------------------------------------------------------+
                |HOW TO CHECK WHETHER THE SERVER IS RUNNING AND HOW TO START THE SERVER|
                +-------------------------------------------------------------------+

In order to use ecFlow, first check whether the ecflow_server is running.
Eexecute one of the following commands:

        ecflow_client --stats

        ecflow_client --ping --host=$ECF_NODE --port $ECF_PORT

        netstat -lnptv | grep ecflow  (only if server started with your user ID)

If the message sent to stdout shows errors in connecting to the server and there is no
mention to:  Status RUNNING, then you need to start the ecflow_server.

On a shared machine multiple users and ecFlow servers can coexist.


                !!!!!! HOW TO START AND STOP THE SERVER !!!!!!


If the server is not running, then start an ecflow_server : by typing the following

        ecflow-crma_start.sh  -d $ECF_HOME -p $ECF_PORT

        ecflow-crma_stop.sh   -p $ECF_PORT


                !!!!!! - - - - - - - - - - - - - - - - !!!!!!
```

Per la verifica sull'attività

Per l'attivazione o la disattivazione

Passo 2: realizzare la suite e caricarla sul server

- Seguire la sintassi ecFlow per realizzare il file di definizione della suite
- Seguire le indicazioni CRMA per le suite operative:
  - utilizzo del file di inizializzazione della suite .ini
  - strutturare le famiglie secondo le definizioni CRMA
  - informazioni disponibili nell'articolo tecnico-scientifico del CRMA:

    Elementi essenziali per la costruzione delle suite per l'uso dell'ecFlow.
    Arpaweb in aria/utilita/Documenti_e_presentazioni/tecnico_scientifiche_docs/**ats_2017_001.pdf**
- Caricamento della suite load
- Attivazione della suite begin

File e loro estensioni

Esistono delle estensioni nei nomi dei file che è importante ricordare

- <name>.def **Definition** file # Suite definition file, describes a suite
  - Expanded or high level
- <name>.ecf **Wrapper** task file # Describes the task (includes and %VAR%)
- <name>.jobN **job-file**
  - created by ecflow from the ecf-file
  - that is sent by ecflow to be executed
  - instance of a task
- <name>.usrN **alias-file**: created from user interaction with ecflowview
  - Test, debug, rerun without status side effects
  - Alias has an alias number and a job instance number

Le variabili di suite e loro ereditarietà

Il simbolo di percentuale (%) viene usato di default per individuare variabili di suite

- Variables # hosted in the server, substituted into a job
  - %VAR% # ECF_MICRO !
  - %VAR:default% # default value?
- There are four different kinds of inheritance in ecflow
- **Variable** inheritance
  - looks at the task first, then parents until it reaches ecflow itself
- **Status** inheritance
  - family status reflects most important status of its tasks
  - likewise for suites and ultimately for ecflow
- **Dependency** inheritance
  - dependencies on any level
  - for task to run, it must be free to run as well as its parents
  - Trigger dependencies may be “hidden” below, time dependencies are not!
- **Zombie handling** inheritance

Personalizzazione dell'uso di ecFlow al CRMA

File di inizializzazione della suite

Tutte le suite operative debbo essere dotate di un file di inizializzazione in formato ASCII che viene integrato nel task per mezzo del comando SHELL source. Questo file contiene variabili che caratterizzano il flusso e che rendono possibile la sua esecuzione in contesti diversi evitando si scaricare la suite dal server, modificarla ed infine ricaricarla per la nuova esecuzione

Funzione per il monitoraggio dei job sottoposti alle code di calcolo

Tutte le suite caricano nel preambolo, tramite direttiva %include, la funzione appositamente scritta per la sottomissione di job alla coda che garantisce al flusso di essere dipendente dall'esecuzione job e dal risultato dell'exit status del job.

Definizione delle famiglie essenziali alla suite

Il ruolo di log, sentinel, spy

Opportuna definizione di:

- ✓ ECF_HOME
- ✓ ECF_INCLUDE,

Ulteriori informazioni a riguardo di ecFlow al CRMA

- Inibito l'uso di cron e di at per tutti i compiti operativi, salvo per le operazioni di spegnimento ed accensione periodica dei server.

- Quali utenti usano regolarmente ecFlow:
  - ✓ operative,
  - ✓ gallaii,
  - ✓ giaiottid,
  - ✓ Goglioa

- Considerazione sui tempi e persone impiegati alla migrazione in ecFlow di flussi non nati in ecFlow
  - F-Air inizio 21/12/2016 fine 03/06/2017 – due persone
  - BRONX inizio 13/02/2017 fine 27/02/2017 – una persona

I tempi sono maggiori rispetto a quelli dello sviluppo *Ab imis fundamentis,* quindi conviene progettare e sviluppare il flusso direttamente secondo la logica ecFlow at CRMA

Data provenance, documentazione e business continuity

Data provenance

I log file dei flussi sono archiviati, così come i log file dei job sottoposti alle code di calcolo.

- Automazione del processo
  - Non ancora completa per tutti i flussi
- Archiviazione permanente dei log file
  - Backup dei flussi ed esportazione su dischi esterni.

Documentazione

Tutti i flussi di sviluppo ed operativi sono documentati e versionati

- Documentazione della progettazione e sviluppo tramite TRAC
- Monitoraggio tempi di realizzazione tramite TRAC
  - Non ancora automatizzato
- Versionamento tramite repository Git.
- Continuous integration tramite Jenkins
  - Non articolata, semplice solo sull'esistenza delle trappole

Business continuity

- Portabilità dei flussi (quasi) automatica su COSILT