HTCondor come batch system in un sito Grid

Configurazione, interazione con il middleware, gestione

HTCondor – Milano T2

- In uso da diversi anni (prima di CREAM)
  - Introdotto per problemi di scalabilità con PBS
    - … e per la presenza di know how locale
  - Inizialmente affiancato a PBS
    - Insiemi disgiunti di WN
  - Nessun problema osservato con HTCondor
    - …dopo varie patch in diversi componenti del m/w grid
  - Phase out progressivo di PBS in circa un anno
    - Nuovi WN aggiunti solo al CE Condor, dismissione dei WN obsoleti

HTCondor – Milano T2

- Configurazione attuale
  - 2 CREAM CE su VM
  - 1 Central Manager
    - 2 Xeon E5506 @ 2.13GHz
    - 24 GB RAM
    - Ampiamente sottoutilizzato…
  - 88 WN
    - 1928 CPU logiche
    - Numero di job slot variabile

HTCondor – Milano T2

- Configurazione dei CE
  - Nessun demone condor
    - Solo macchina di sottomissione, configurazione minimale
    - …In realtà gira *condor_master* ma non realmente necessario
    - File extra in /etc/condor/config.d per l'aggiunta di valori di accounting

```
# file: /etc/condor/condor_config.local

CONDOR_HOST = t2-ce-03-125.mi.infn.it

CONDOR_IDS = 9991.999
DAEMON_LIST = MASTER

SEC_DEFAULT_AUTHENTICATION_METHODS = FS_REMOTE
FS_REMOTE_DIR = /var/lib/condor/auth
DELEGATE_JOB_GSI_CREDENTIALS = False
```

```
# file: /etc/condor/config.d/blah

# DGAS information to be inserted in the job classad
JobType = "grid"
ceCertificateSubject =
"/C=IT/O=INFN/OU=Host/L=Milano/CN=t2-ce-04.mi.infn.it"
SiteName = "INFN-MILANO-ATLASC"
SUBMIT_EXPRS = JobType ceCertificateSubject SiteName
```

- Configurazione del CM (1/3)

```
# file: condor_config.local

CONDOR_HOST = $(FULL_HOSTNAME)

SEC_DEFAULT_AUTHENTICATION_METHODS = FS, FS_REMOTE
FS_REMOTE_DIR = /var/lib/condor/auth
DELEGATE_JOB_GSI_CREDENTIALS = False
BIND_ALL_INTERFACES = True
ALLOW_WRITE = *.mi.infn.it
ALLOW_ADMINISTRATOR = $(ALLOW_ADMINISTRATOR), $(FULL_HOSTNAME)
UID_DOMAIN = mi.infn.it
FILESYSTEM_DOMAIN = $(FULL_HOSTNAME)

COLLECTOR_NAME = INFN-MILANO-ATLASC
COLLECTOR_UPDATE_INTERVAL = 60

CONDOR_IDS = 9991.999

MAX_HISTORY_LOG = 83886080
MAX_HISTORY_ROTATIONS = 60

DAEMON_LIST = $(DAEMON_LIST), COLLECTOR, NEGOTIATOR, GANGLIAD
MASTER.USE_PROCD = True

NEGOTIATOR.CLASSAD_LOG_STRICT_PARSING = False
#NEGOTIATOR_SLOT_CONSTRAINT =  Machine != "t2-wn-29.mi.infn.it"
```

- Configurazione del CM (2/3)
  - Disabilitazione della preemprion

```
# Disable preemption by machine activity.
SUSPEND = FALSE
KILL = FALSE
PREEMPT = False
# Disable preemption by user priority.
PREEMPTION_REQUIREMENTS = False
# Disable preemption by machine RANK by ranking all jobs equally.
RANK = 0
# Since we are disabling claim preemption, we
# may as well optimize negotiation for this case:
NEGOTIATOR_CONSIDER_PREEMPTION = False
# after 20 minutes, schedd must renegotiate to run
# additional jobs on the machine
CLAIM_WORKLIFE = 1200
```

- Configurazione del CM (3/3)
  - *Schedd* – un file per coda

```
# file: /etc/condor/config.d/q_atlas
SCHEDDATLAS = $(SCHEDD)
SCHEDDATLAS_ARGS = -f -local-name atlas
SCHEDDATLAS_LOG = $(LOG)/ScheddLog.atlas
SCHEDDATLAS_ENVIRONMENT = "_CONDOR_SPOOL=$(SPOOL)/atlas"
SCHEDD.atlas.SCHEDD_NAME = atlas@t2cm
SCHEDD.atlas.SCHEDD_LOG = $(SCHEDDATLAS_LOG)
SCHEDD.atlas.SCHEDD_ADDRESS_FILE = $(SPOOL)/.schedd_address
SCHEDD.atlas.SCHEDD_DAEMON_AD_FILE = $(SPOOL)/.schedd_classad
SCHEDD.atlas.HISTORY = $(SPOOL)/history

SCHEDD.atlas.MaxCPUTime = 172800
SCHEDD.atlas.MaxWallclockTime = 345600
SCHEDD.atlas.GlueStatus = "Production"

DAEMON_LIST = $(DAEMON_LIST), SCHEDDATLAS
VALID_SPOOL_FILES = $(VALID_SPOOL_FILES), atlas
SCHEDD_EXPRS = MaxCPUTime, MaxWallclockTime, GlueStatus
```

HTCondor – Milano T2

- Configurazione dei WN
  - 193 slot statiche da 8 core
  - 360 slot statiche da 1 core
  - 2 slot partizionabili da 16 core
    - Sperimentali – utilizzati per testare `condor_defrag`
    - Normalmente eseguono job da 1 core, possono eseguire quelli da 8 in mancanza dei primi
    - Necessario approfondire semantica di Glue2 (come si pubblicano degli slot dinamici?)

HTCondor – Milano T2

- Configurazione dei WN: slot partizionabili
  - Utilizzati per fare *dynamic provisioning* delle risorse
  - Da una slot partizionabile vengono ricavate slot dinamiche "su misura" per le richieste del job
  - http://research.cs.wisc.edu/htcondor/manual/v8.4/3_5Policy_Configuration.html#36689
- Direttive aggiunte in /etc/condor/config.d:

```
# file:/etc/condor/config.d/partitionable
SLOT_TYPE_2 = cpus=100%, ram=100%
SLOT_TYPE_2_PARTITIONABLE = True
NUM_SLOTS_TYPE_2 = 1
```

HTCondor – Milano T2

- Configurazione dei WN: slot statiche da 1 core
  - Il file di configurazione è lo stesso del CM, salvo alcune macro specifiche per lo *startd*

```
DAEMON_LIST = MASTER, STARTD
HOSTALLOW_CONFIG = $(CONDOR_HOST), $(FULL_HOSTNAME)
HOSTALLOW_ADMINISTRATOR = $(HOSTALLOW_CONFIG)
HOSTALLOW_NEGOTIATOR = $(CONDOR_HOST), \
                       t2-ce-03-125.mi.infn.it, t2-ce-03-127.mi.infn.it

USER_JOB_WRAPPER = /etc/condor/condor_starter.sh

BIND_ALL_INTERFACES = True
MAXJOBRETIREMENTTIME=$(HOUR)*100
CLAIM_WORKLIFE = 1200
```

- Configurazione dei WN: slot statiche da 8 core
  - Direttive aggiunte in /etc/condor/config.d:

```
# file: /etc/condor/config.d/mc_node
# Static 8-core slots
SLOT_TYPE_1 = cpus=8
NUM_SLOTS_TYPE_1 = 3
SLOT_TYPE_1_PARTITIONABLE = false
```

HTCondor – Milano T2

- Integrazione con CREAM a livello “artigianale”
  - Soluzione adottata da pochi siti
    - …ma interesse crescente durante l’ultimo anno
  - Use case di Milano estremamente semplice
    - Supporto di una singola VO (ATLAS): fair share banale
    - Solo pilot jobs: BDII sostanzialmente inutilizzato

HTCondor – Milano T2

- Componenti sviluppate
  - Information provider
  - Accounting
- Sviluppo fortemente legato ad un aspetto critico: la simulazione delle code

HTCondor e le code

- HTCondor non prevede il concetto di code
  - Il demone *schedd* mantiene tutti i job in un'unica coda
- Una semantica analoga alle code è comunque ottenibile con (almeno) due approcci
  - Job-attribute: inserendo un attributo ad hoc nella descrizione del job (es. "BatchQueue")
  - Multi-schedd: utilizzando diversi *schedd* (uno per coda)
- Provati entrambi a Milano, attualmente in uso il secondo

HTCondor e le code: Job-attribute

- In fase di sottomissione del job, un attributo “Batchqueue” viene inserito nel classad
- L’attributo può essere utilizzato facilmente per gestire il job, es. in espressioni come `PERIODIC_REMOVE`
- Pro
  - Approccio più in spirito Condor
  - Lo *schedd* può girare indifferentemente sul CE o sul CM
- Contro
  - Manca un posto ‘naturale’ dove conservare le informazioni sulla coda
  - La configurazione deve essere replicata per tutti gli schedd

HTCondor e le code: Job-attribute

- Implementazione
  - Attributo “pseudoqueue” inserito dallo script di blahpd `condor_submit.sh`
  - Perl script `condor_pseudoqueue_config.pl`
    - Invocato da condor per generare la configurazione, legge il file condor_config.local e aggiunge direttive di `PERIODIC_REMOVAL` per l’enforcing dei limiti
    - Invocato con differenti parametri dal ‘resource BDII’ per generare le informazioni da pubblicare

HTCondor e le code: Multi-schedd

- Viene fatto partire uno *schedd* per coda
- Tutti i CE devono sottomettere agli stessi *schedd*
- Pro
  - Concettualmente più simile alle code
  - Informazioni sulle code gestite in un unico punto
  - Code indipendenti: aggiungere una coda equivale ad inserire un file sul CM
- Contro
  - Le informazioni sui job (stato, history, etc.) sono distribuite tra i vari *schedd*
  - Non si può aumentare arbitrariamente il numero di *schedd*: possibili problemi di scala

HTCondor e le code: Multi-schedd

Esempio di implementazione: definizione della coda ATLAS

```
# file: /etc/condor/config.d/q_atlas
SCHEDDATLAS = $(SCHEDD)
SCHEDDATLAS_ARGS = -f -local-name atlas
SCHEDDATLAS_LOG = $(LOG)/ScheddLog.atlas
SCHEDDATLAS_ENVIRONMENT = "_CONDOR_SPOOL=$(SPOOL)/atlas"
SCHEDD.atlas.SCHEDD_NAME = atlas@t2cm
SCHEDD.atlas.SCHEDD_LOG = $(SCHEDDATLAS_LOG)
SCHEDD.atlas.SCHEDD_ADDRESS_FILE = $(SPOOL)/.schedd_address
SCHEDD.atlas.SCHEDD_DAEMON_AD_FILE = $(SPOOL)/.schedd_classad
SCHEDD.atlas.HISTORY = $(SPOOL)/history

SCHEDD.atlas.MaxCPUTime = 172800
SCHEDD.atlas.MaxWallclockTime = 345600
SCHEDD.atlas.GlueStatus = "Production"

DAEMON_LIST = $(DAEMON_LIST), SCHEDDATLAS
VALID_SPOOL_FILES = $(VALID_SPOOL_FILES), atlas
SCHEDD_EXPRS = MaxCPUTime, MaxWallclockTime, GlueStatus
```

HTCondor e Infosys

- Script custom:
  - glite-info-dynamic-condor
    - Utilizza i comandi `condor_q` e `condor_status` per recuperare informazioni dagli *schedd* e dal *negotiator*
    - Utilizza l'opzione `-format` per presentare le informazioni in formato Glue e Glue2
  - lrmsinfo-condor
    - Produce la lista di job per il calcolo di EWT ed ERT

HTCondor e Accounting

- Script custom:
  - htcondor_acc.sh
    - Legge il/i file di history di HTCondor e produce un file intermedio di usage records
    - Avvia il parser apel per raccogliere e trasmettere i dati di accounting
    - Alcune importanti differenze a seconda che si usi “Job-attribute” o “Multi-schedd”
  - htcondor.py
    - Implementa la classe HTCondorParser, utilizzata da apel per interpretare le righe prodotte dallo script precedente
  - Patch a /usr/bin/apelparser per l’import della classe HTCondorParser
- Il tutto dovrebbe essere ora incluso nella release ufficiale di apel

Gestione

- Drain dei nodi
  - Opzione `-peaceful` di `condor_off` e `condor_restart`
- Esclusione di nodi
  - NEGOTIATOR_SLOT_CONSTRAINT espressione booleana valutata sui classad degli *startd*, solo quelli per cui vale True vengono considerati dal *negotiator*
- Controllo dello stato del sistema
  - Versatilità di `-constraint`

Gestione

- Normalmente i WN non attivi non compaiono in condor (il *collector* li "dimentica"), problema per le info pubblicate nel BDII
- Risolvibile con alcune macro di configurazione
  - ABSENT_REQUIREMENTS
  - COLLECTOR_PERSISTENT_AD_LOG
  - ABSENT_EXPIRE_ADS_AFTER
    - Per i dettagli vedere "absent classads" http://research.cs.wisc.edu/htcondor/manual/v8.4/3_10Monitoring.html#SECTION00410200000000000000
- …e l'opzione `–absent` di `condor_status`

HTCondor - Milano T3

- HTCondor è utilizzato anche nella farm di calcolo del Tier3 di ATLAS e di alcuni gruppi di Fisica a Milano
- Ogni gruppo raggruppa le proprie risorse in un pool Condor
- Tutti i gruppi utilizzano slot partizionabili
- Obiettivo: condividere le risorse quando non utilizzate
- The Condor way: flocking

HTCondor - Milano T3

- Il flocking diretto richiede che ogni pool "conosca" tutti gli altri: problemi di management
- Approccio differente: **super-pool**
- La ricetta è ispirata a questa wiki page https://htcondor-wiki.cs.wisc.edu/index.cgi/wiki?p=HowToHaveExecuteMachines
- Gli *startd* sono pubblicati sia nel *collector* del proprio pool, sia in una macchina centrale (super-pool) che fa da *negotiator* per tutti i nodi
- Ai job di ogni pool è consentito il flocking verso il (solo) super-pool

- Configurazione del super-pool

```
# file: /etc/condor/condor_config.local

##  What machine is your central manager?
CONDOR_HOST = $(FULL_HOSTNAME)

## Pool's short description
COLLECTOR_NAME = Condor super-pool at $(FULL_HOSTNAME)

##  This macro determines what daemons the condor_master will start and keep its watchful
eyes on.
##  The list is a comma or space separated list of subsystem names
DAEMON_LIST = COLLECTOR, MASTER, NEGOTIATOR

# Insert NegotiatorMatchExprNegotiatorName="SuperPool" into matches
# that this negotiator makes.  This is used by the startd to give
# the local negotiator priority over the super negotiator.
NegotiatorName = "SuperPool"
NEGOTIATOR_MATCH_EXPRS = NegotiatorName

# Configure authorization settings to permit startds in sub-pools to join
# the super-pool and to allow submission of jobs from all appropriate
# places.

ALLOW_WRITE = $(ALLOW_WRITE),*.mi.infn.it,*.fisica.unimi.it
```

- Configurazione dei pool federati (1/2)

```
# Insert NegotiatorMatchExprNegotiatorName="<LocalPoolName>" into matches
# that this negotiator makes.  This is used by the startd to give
# the local negotiator priority over the super negotiator.
NegotiatorName = "magi-pool"
NEGOTIATOR_MATCH_EXPRS = NegotiatorName

# For advertising to super-pool
SUPER_COLLECTOR = condor-superpool.mi.infn.it
LOCAL_COLLECTOR = $(CONDOR_HOST)

# the local negotiator should only ever report to the local collector
NEGOTIATOR.COLLECTOR_HOST = $(LOCAL_COLLECTOR)
# startds should report to both collectors
STARTD.COLLECTOR_HOST = $(LOCAL_COLLECTOR),$(SUPER_COLLECTOR)

# trust both negotiators
#ALLOW_NEGOTIATOR=$(COLLECTOR_HOST)
ALLOW_NEGOTIATOR = $(LOCAL_COLLECTOR),$(SUPER_COLLECTOR)

# Ensure external users get big priority factor.
# If you don't have a uniform uid domain for all local users, then
# you will need to have some external process that updates priority
# factors using condor_userprio.
ACCOUNTANT_LOCAL_DOMAIN = $(UID_DOMAIN)
```

- Configurazione dei pool federati (2/2)

```
# Flocking to super-pool
FLOCK_TO = $(SUPER_COLLECTOR)

# Advertise in the machine ad the name of the negotiator that made the match
# for the job that is currently running.  We need this in SUPER_START.
CurJobPool = "$$(NegotiatorMatchExprNegotiatorName)"
SUBMIT_EXPRS = $(SUBMIT_EXPRS) CurJobPool
STARTD_JOB_EXPRS = $(STARTD_JOB_EXPRS) CurJobPool

# Turn PREEMPT on only for jobs comming from an external pool
PREEMPT = ($(PREEMPT)) && (MY.CurJobPool =!= $(NegotiatorName))

# We do not want the super-negotiator to preempt local-negotiator matches.
# Therefore, only match jobs if:
#      1. the new match is from the local pool
#   OR 2. the existing match is not from the local pool
SUPER_START = NegotiatorMatchExprNegotiatorName =?= $(NegotiatorName) || \
              MY.CurJobPool =!= $(NegotiatorName)

START = ($(START)) && ($(SUPER_START))
```

HTCondor - superpool

- Patch dello *schedd* (in schedd.cpp) (by F. Prelz)
  - Al vaglio degli sviluppatori di HTCondor
  - In uso a Milano da circa due mesi
- Risolve un problema di coesistenza tra “super-pool” e slot dinamici
  - Per distinguere e dare diversa priorità tra job locali e remoti viene usata l’espressione
    `CurJobPool = "$$(NegotiatorMatchExprNegotiatorName)”`
  - A causa di un bug, NegotiatorMatchExprNegotiatorName veniva definita solo per la prima slot dinamica
  - Le successive slot venivano comunque ricavate ma non erano in grado di far partire i job, che rimanevano in HOLD

HTCondor - superpool

- Patch del *dedicated scheduler* (dedicated_scheduler.cpp) by F. Prelz
  - Al vaglio degli sviluppatori di HTCondor
  - In uso a Milano da circa due mesi
- Problema di coesistenza tra slot dinamici e job dell'universo "parallel"
  - Race condition nella creazione e allocazione di slot dinamiche per job paralleli con alto numero di CPU
  - Vengono create alcune slot, che vengono sottratte dalla slot primaria prima che siano "idle"
  - La slot primaria non ha più abbastanza risorse per soddisfare il job, e il matchmaking viene arrestato

Conclusioni

- HTCondor estremamente stabile per il T2
  - A mia memoria, mai stato necessario un riavvio di alcun servizio
- Necessario un po' di lavoro sull'integrazione con CREAM
  - Rimasto in sospeso a causa della ridotta base di utenti
  - Il grosso è fatto, si tratta di generalizzare gli script per scenari diversi di deployment (es. "job-attribute" vs "multi-schedd")
  - Packaging degli info provider per la distribuzione con EMI
- Quindi…


Si… può… fare!

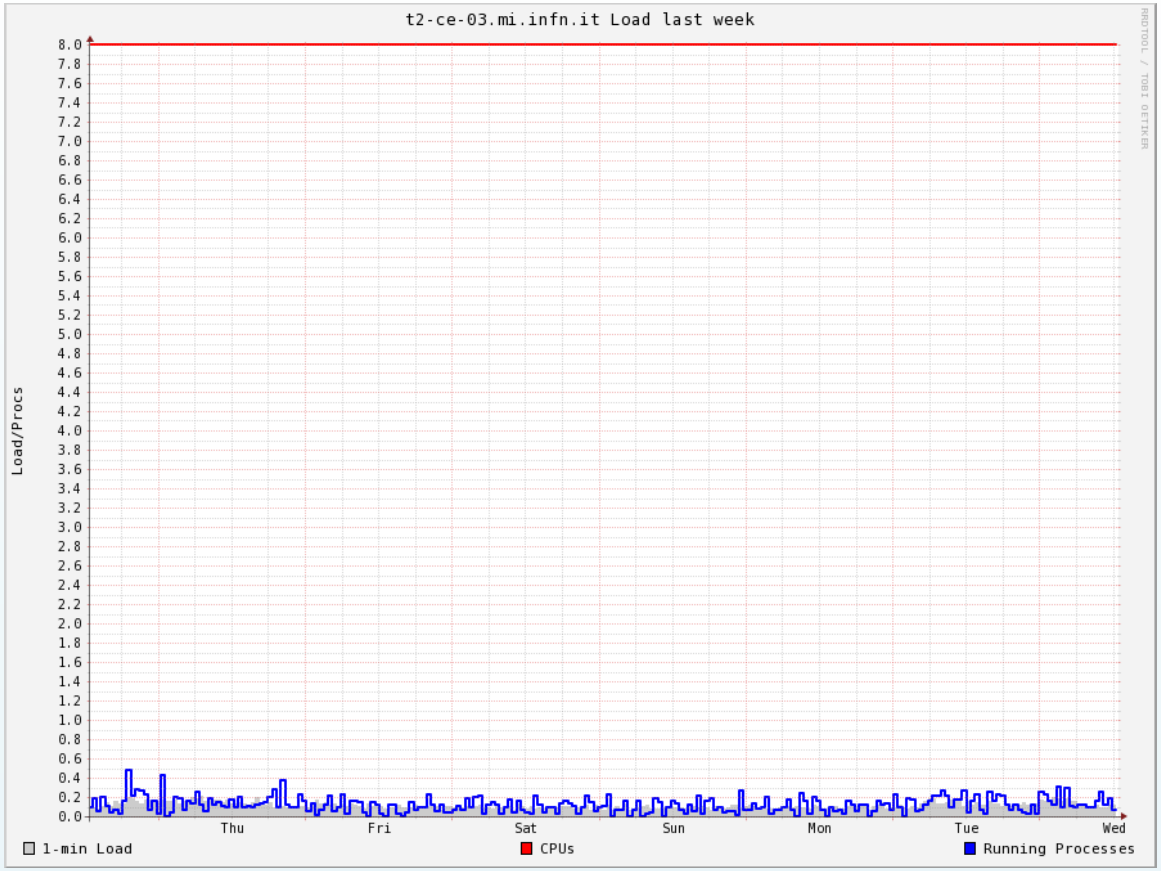
BACKUP SLIDES

- Configurazione dei WN (slot partizionabili)
  - tentativo in corso

```
# Make a single partitionable slot
SLOT_TYPE_1 = cpus=100%, ram=100%
SLOT_TYPE_1_PARTITIONABLE = True
NUM_SLOTS_TYPE_1 = 1

## Specify which jobs the startd is willing to start
#   mc  = only multicore job
#   sc  = only single core job
#   all = any job
MCmode = "mc"
STARTD_ATTRS = $(STARTD_ATTRS) MCmode

## Make MCmode modifiable at runtime
ENABLE_RUNTIME_CONFIG = True
STARTD.SETTABLE_ATTRS_OWNER = MCmode
SETTABLE_ATTRS_OWNER = MCmode

## Start only the proper kind of jobs
START = $(START) && ((MCmode == "all”) || (MCmode == "sc" && TARGET.RequestCpus == 1 ) || ( MCmode
== "mc" && TARGET.RequestCpus == 8 ) )

STARTD_DEBUG = D_FULLDEBUG D_COMMAND
```

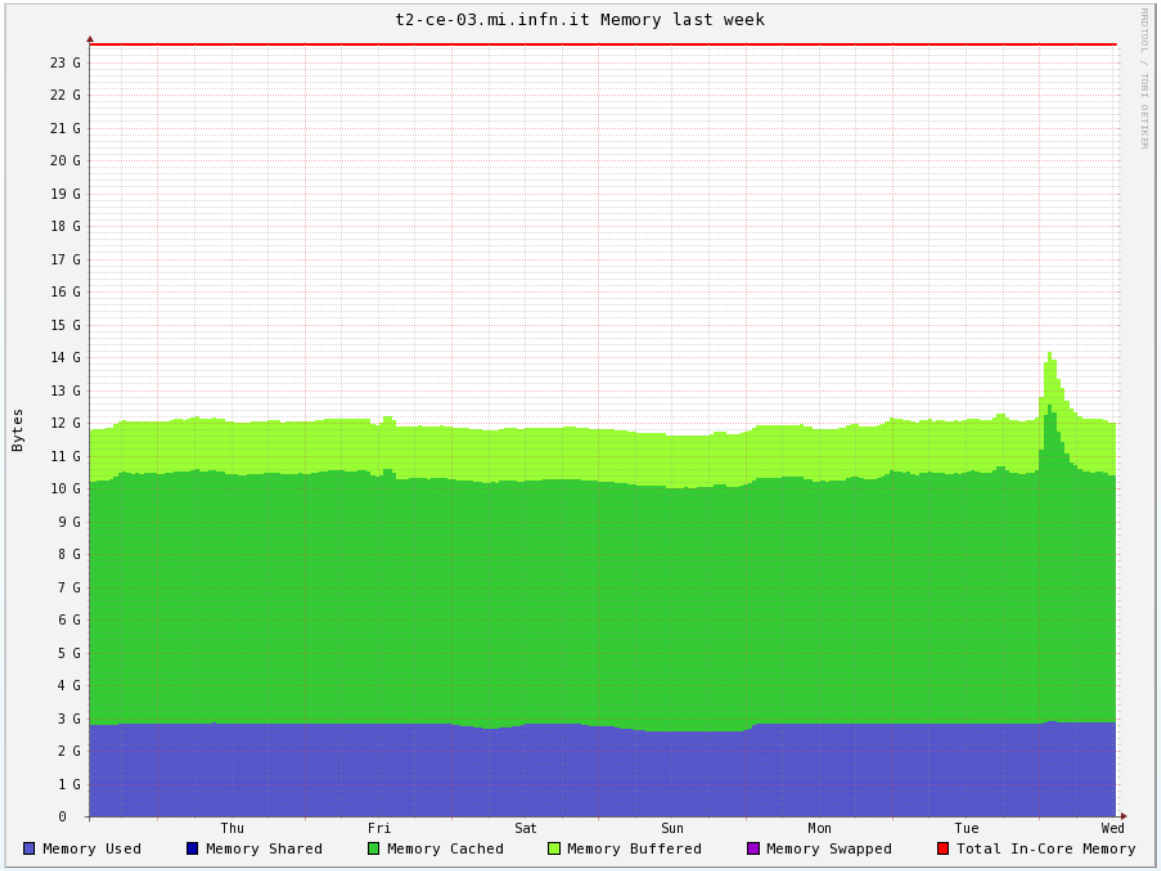
HA e flocking

- Section 3.2.11.1 of HTCondor manual:
  - "The HTCondor high availability mechanisms discussed in this section currently do not work well in configurations involving flocking. The individual problems listed listed below interact to make the situation worse. Because of these problems, we advise against the use of flocking to pools with high availability mechanisms enabled."
- http://research.cs.wisc.edu/htcondor/manual/v8.4/3_11High_Availability.html#SECTION00411210000000000000



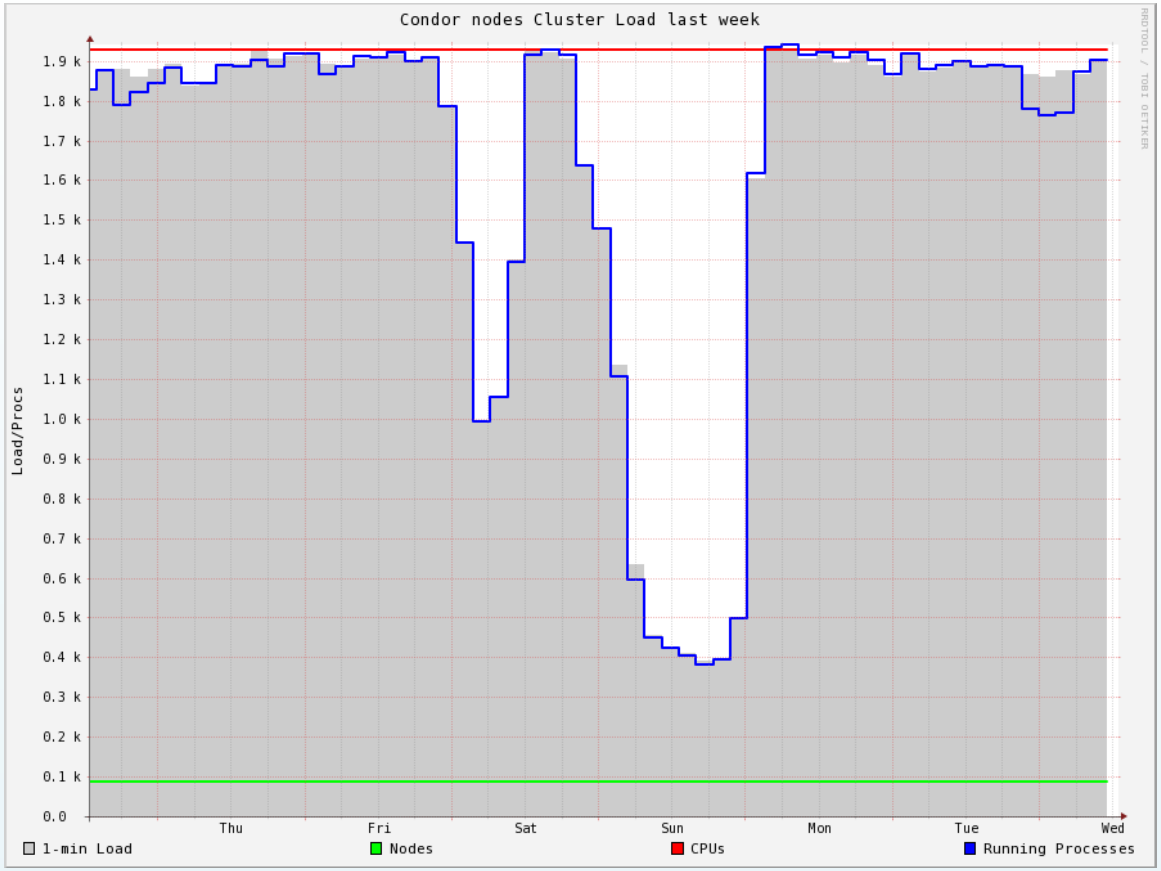
Carico del Central Manager

Ganglia 1 week graph



Central manager memory usage

Ganglia 1 week graph



Worker Nodes load

Ganglia 1 week graph