Corso di formazione sull'utilizzo del datacenter ReCaS-Bari: I sistemi batch

Alessandro Italiano

Agenda

- Introduzione/Definizione di Batch System
- L'implementazione del batch system al ReCaS DataCenter
- Utilizzare il batch system, esempi e best practice

Definizione di batch[system]

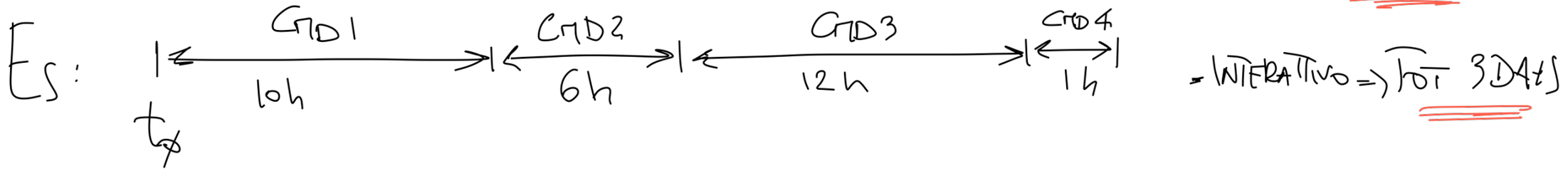
- La traduzione dall'Inglese di "batch" e' "infornata" ed il parallelo con il mondo della cucina e' molto calzante

    - Preparo qualche cosa da mangiare che richiede la cottura in forno
    - Imposto il tipo di cottura e la temperatura del forno ottimale per cuocere quello che ho preparato
    - Inforno per un certo periodo di tempo
    - Non interagisco con la pietanza mentre è in forno
    - al termine della cottura tiro fuori ciò che avevo informato sperando che la cottura abbia fatto la sua parte[molto spesso non e' cosi !] .

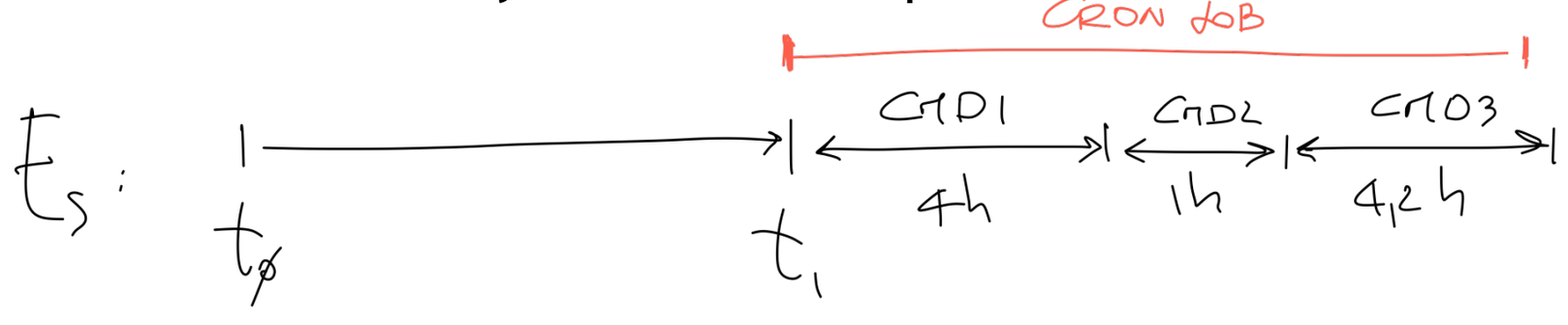
Definizione di batch[system]

- In ambito sistemistico il concetto di batch riporta alla capacità di eseguire comandi in modalità non interattiva.

  - Tipicamente per eseguire in serie una lista di comandi che alternativamente richiederebbero l'interazione con la shell.



- Il cron job è un altro tipico esempio di esecuzione in modalità batch di comandi Unix. L'esecuzione del cronjob è fissata in un preciso momento del futuro.



Definizione di batch system

- Negli anni 60/70 la possibilità di spostare l'esecuzione dei programmi in modalità non interattiva in avanti nel futuro è stata una vera rivoluzione perché ha permesso di

    - diminuire di fatto i tempi di attesa per recuperare l'output del programma eseguito, poichè veniva eseguito subito dopo il precedente in maniera automatica.

        - allocare tutti i cicli di CPU disponibili delle macchine che all'epoca erano considerate merce rarissima.

- Il batch system nasce e si sviluppa come sistema in grado allo stesso tempo di

    - garantire l'esecuzione di programmi in modalità non interattiva, comunemente detti batch job

    - sfruttare al massimo le risorse computazionali disponibili[al netto dell'efficienza dei batch job]

Il Batch System

- Un batch system deve essere in grado di gestire
  - Una coda in cui conservare tutti i batch job fino alla loro esecuzione. [Queue]
  - Lo stato delle risorse in maniera consistente. [LRMS]
  - L'allocazione del batch job sulla risorsa appropriata tenendo presente i requisiti del programma da eseguire. [Scheduler]

USERS

PGR1

PGR2

PGR3

PGR4

BATCH SYSTEM

SCHEDULER

QUEUE

LRMS

RESOURCE

TIPO A

TIPO C

TIPO B

TIPO D

Il Batch System

- Facilita l'uso delle risorse
    - Sottomissione di job in maniera flessibile e attraverso cmdline o api integrabili con workflow applicativi più complessi.
    - In termini di quantità. Un utente potrebbe avere a disposizione migliaia di core che darebbero un decisivo boost alle proprie attività di calcolo
    - In termini di eterogeneità. Memoria, Core, MultiCore, Disco Veloce, GPU, Rete a bassa latenza
- Garantisce una accesso giusto [Fair] alle risorse
    - Ogni Gruppo di Ricerca/SottoGruppo/utente usa una quota dinamica delle risorse disponibili che é funzione delle risorse usate nel passato e del contribuito anno versato al datacenter
    - I pochi limiti imposti garantiscono l'usabilità delle risorse in un ambiente multidisciplinare

Il Batch System del DataCenter ReCaS

- Esperienze quasi ventennale con i Batch System, in particolare di tipo OpenSource
- Al principio era PBS[Torque e Maui]
- In questo momento ci sono due Batch System in uso
  - HTCondor è subentrato col cambio generazionale mantenendo la natura OpenSource
  - PBS resiste[ancora per poco] solo per la gestione del cluster HPC

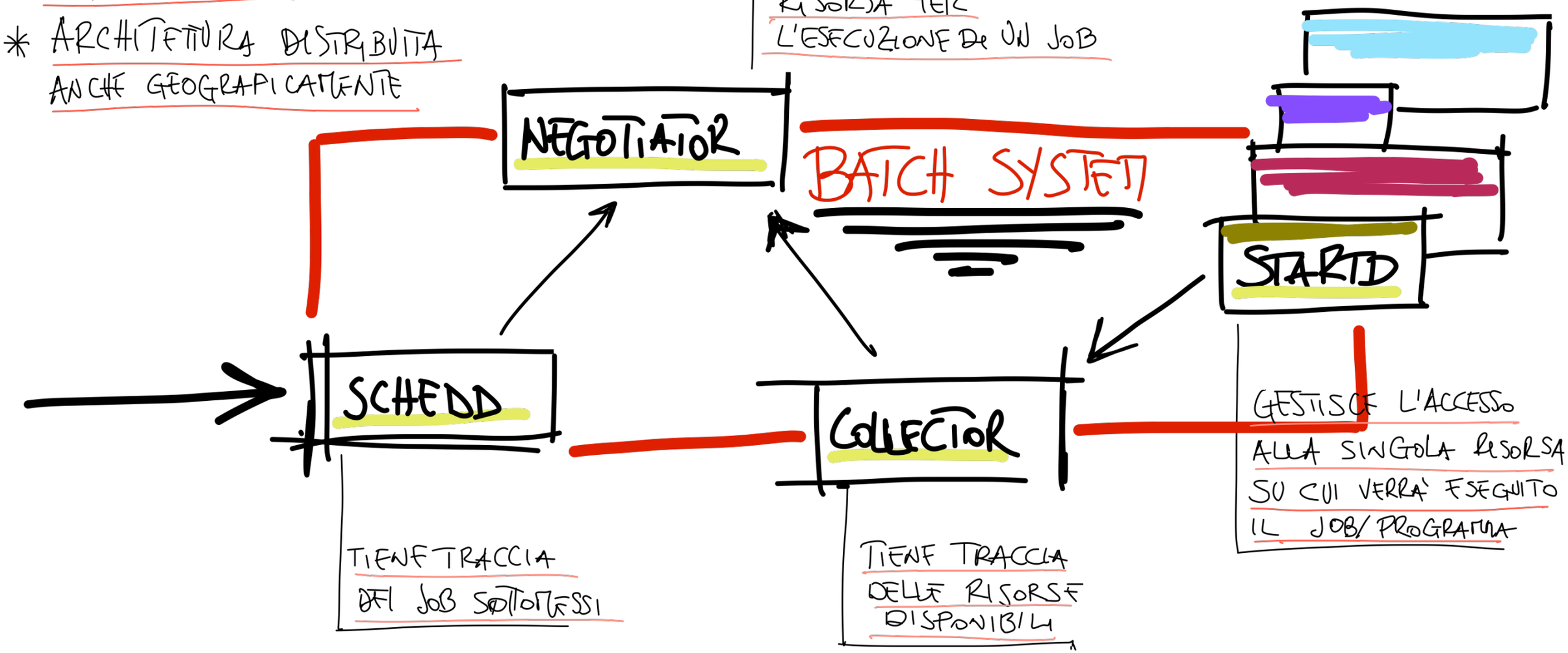
HTCondor

- Nasce 34 anni fa insieme al concetto di High Throughput Computing
- Sviluppato e mantenuto dall'Università del Wisconsin-Madison
- Si é sviluppato sul concetto dell' opportunistic computing sfruttando i cicli di CPU Idle da cui prende il nome originario "Condor"
- Negli ultimi anni ha avuto un grande diffusione nelle varie comunità scientifiche sicuramente per via della sua robustezza oltre al fatto di essere license-free

HTCondor: Architettura

* OGNI ELEMENTO DELL'ARCHITETTURA CORRISPONDE AD UN PROCESSO UNIX
* PROCESSI IN ESECUZIONE PREFERIBILMENTE SU HOST DIVERSI
* ARCHITETTURA DISTRIBUITA ANCHE GEOGRAFICAMENTE

TROVA LA MIGLIOR RISORSA PER L'ESECUZIONE DI UN JOB



HTCondor: Classads

- Classified Advertisements
- Sono usati per descrive i singoli componenti del Batch system
    - Job, server, processi, risorse
- Ogni classad é una lista di attributi ognuno dei quali mappato ad una espressione più o meno complessa
    - attributo = espressione
    - RequestMemory = 4096
    - Owner == "italiano"
    - leave_in_queue = JobStatus == 4 && ((CurrentTime - CompletionDate) < 7200)

HTCondor: Classads

- Il funzionamento di HTCondor è basato profondamente sullo scambio, la modifica ed il parsing dei classads tra processi
  - lo startd invia il suo classad al collector al momento della registrazione
  - il negotiator fa il parsing di tutti i classad dello schedd e del collector per trovare la migliore risorsa disponibile per l'esecuzione di un job
  - quando un job parte alcuni attributi del classad vengono modificati o aggiunti

HTCondor: classads

Job Classad

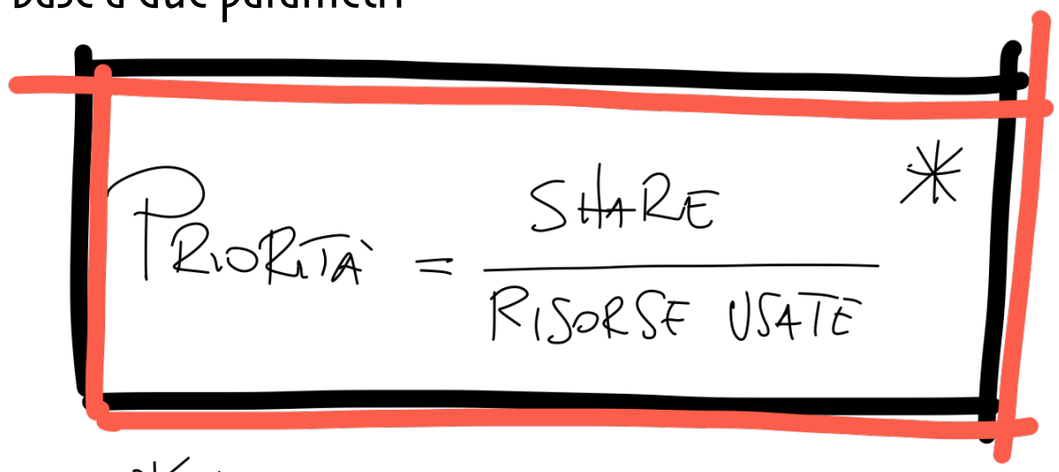
```
MaxHosts = 1
MemoryUsage = ( ( ResidentSetSize + 1023 ) / 1024 )
Env = "CONDOR_ID=2024598"
AccountingGroup = "group_astroparticle.dampeuser.dampeuser"
User = "dampeuser@ReCaSCluster"
EncryptExecuteDirectory = false
OnExitHold = false
CoreSize = 0
WallClockCheckpoint = 90864
MachineAttrCpus0 = 1
WantRemoteSyscalls = false
MyType = "Job"
Rank = Memory
leave_in_queue = JobStatus == 4 && ( ( CurrentTime - CompletionDate ) < 7200 )
CumulativeSuspensionTime = 0
MinHosts = 1
PeriodicHold = false
PeriodicRemove = false
Err = "/lustre/dampe/XROOTD/UserSpace/dampe_prod/mc/workdir/BARI/allProton-v6r0p4_10TeV_100TeV_FTFP_CSUP05/Simulation/2xxxx/4xxx/1xx/10/output.err"
ProcId = 0
EnteredCurrentStatus = 1550937170
BlockWriteKbytes = 0
UserLog = "/lustre/dampe/XROOTD/UserSpace/dampe_prod/mc/workdir/BARI/allProton-v6r0p4_10TeV_100TeV_FTFP_CSUP05/Simulation/2xxxx/4xxx/1xx/10/output.clog"
NumJobStarts = 1
AutoClusterAttrs = "_cp_orig_RequestCpus,_cp_orig_RequestDisk,_cp_orig_RequestMemory,JobUniverse,LastCheckpointPlatform,NumCkpts,RequestCpus,RequestDisk,RequestMemory,MachineLastMatchTime,Group,sgmalice,CheckpointPlatform,PartitionableSlot,WithinResourceLimits,ConcurrencyLimits,NiceUser,Rank,Requirements,ops,_cp_orig_RequestGPUs,RequestGPUs,diacono,italiano,Owner,localMountsReady,asi,DiskUsage,FileSystemDomain,MaxHosts"
JobUniverse = 5
AutoClusterId = 6215
In = "/dev/null"
Requirements = ( MaxHosts == 1 ) && ( TARGET.OpSysAndVer == "CentOS7" ) && ( TARGET.Arch == "X86_64" ) && ( TARGET.Disk >= RequestDisk ) && ( TARGET.Memory >= RequestMemory ) && ( ( TARGET.FileSystemDomain == MY.FileSystemDomain ) || ( TARGET.HasFileTransfer ) )
PublicClaimId = "<90.147.169.90:9618>#1541482239#18728#..."
ClusterId = 2024598
CumulativeRemoteUserCpu = 0.0
NumJobCompletions = 0
WhenToTransferOutput = "ON_EXIT"
LastMatchTime = 1550937170
CompletionDate = 0
request_memory = 2500
RecentBlockWrites = 0
BufferSize = 524288
StartdPrincipal = "execute-side@matchsession/90.147.169.90"
TargetType = "Machine"
LeaveJobInQueue = false
RecentStatsLifetimeStarter = 1200
job_lease_duration = 2400
JobNotification = 0
Owner = "dampeuser"
CondorPlatform = "$CondorPlatform: x86_64_RedHat7 $"
CommittedTime = 0
QDate = 1550837344
JobLeaseDuration = 2400
RecentBlockWriteKbytes = 0
TransferIn = false
ExitStatus = 0
StartdIpAddr = "<90.147.169.90:9618?addrs=90.147.169.90-9618+[--1]-9618&noUDP&sock=10633_ee9b_3>"
NumJobMatches = 1
RootDir = "/"
JobCurrentStartDate = 1550937170
GlobalJobId = "ettore.recas.ba.infn.it#2024598.0#1550837344"
CurrentHosts = 1
RemoteSysCpu = 130.0
TotalSuspensions = 0
WantCheckpoint = false
LastJobLeaseRenewal = 1551029534
PeriodicRelease = false
NumCkpts_RAW = 0
CondorVersion = "$CondorVersion: 8.6.13 Oct 30 2018 BuildID: 453497 $"
Out = "/lustre/dampe/XROOTD/UserSpace/dampe_prod/mc/workdir/BARI/allProton-v6r0p4_10TeV_100TeV_FTFP_CSUP05/Simulation/2xxxx/4xxx/1xx/10/output.log"
ShouldTransferFiles = "IF_NEEDED"
BlockReads = 0
DiskUsage = 1
JobRunCount = 1
CumulativeSlotTime = 0
CommittedSlotTime = 0
LocalUserCpu = 0.0
RecentBlockReads = 0
JobStartDate = 1550937170
Group = "dampe"
ExitBySignal = false
StreamErr = false
```

Host Classad

```
ConsumptionDisk = quantize(target.RequestDisk,{ 1024 })
TotalCondorLoadAvg = 6.91
IsValidCheckpointPlatform = ( TARGET.JobUniverse =!= 1 || ( ( MY.CheckpointPlatform =!= undefined ) && ( ( TARGET.LastCheckpointPlatform =?= MY.CheckpointPlatform ) || ( TARGET.NumCkpts == 0 ) ) ) )
JavaSpecificationVersion = "1.7"
JavaVendor = "Oracle Corporation"
RemoteGroup = "group_pk"
RemoteAutoregroup = false
MonitorSelfTime = 1551027466
OpSys = "LINUX"
AccountingGroup = "group_pk.rsg01@ReCaSCluster"
Mips = 16749
DetectedCpus = 64
LastFetchWorkSpawned = 0
JobRankPreemptions = 0
AuthenticatedIdentity = "condor_pool@recascluster"
HibernationState = "NONE"
HasMPI = true
TotalLoadAvg = 8.73
LastDrainStartTime = 1542641367
OpSysAndVer = "SL6"
JobId = "10699.7"
Rank = Memory
JobUserPrioPreemptions = 0
MyType = "Machine"
ConsumptionMemory = quantize(target.RequestMemory,{ 128 })
LoadAvg = 1.0
VirtualMemory = 280496080
MachineResources = "Cpus Memory Disk Swap"
SlotWeight = Cpus
CheckpointPlatform = "LINUX X86_64 2.6.x normal 0x2aaaaaaab000 ssse3 sse4_1 sse4_2"
HardwareAddress = "0c:c4:7a:1e:0d:ce"
HasJobDeferral = true
TotalSlots = 8
TotalDisk = 940272180
HasJICLocalStdin = true
TotalSlotDisk = 9402860.0
JobUniverse = 5
HasFileTransferPluginMethods = "file,ftp,http,data"
Requirements = ( START ) && ( IsValidCheckpointPlatform ) && ( WithinResourceLimits )
NumPids = 100
PublicClaimId = "<90.147.168.184:41526>#1529366873#192985#..."
ClockMin = 1077
UpdatesTotal = 15
UtsnameVersion = "#1 SMP Tue May 22 03:28:18 UTC 2018"
CpuIsBusy = false
HasJICLocalConfig = true
CpuBusy = ( ( LoadAvg - CondorLoadAvg ) >= 0.5 )
UidDomain = "ReCaSCluster"
Arch = "X86_64"
HasCheckpointing = true
RecentJobPreemptions = 0
ExpectedMachineQuickDrainingCompletion = 1551028064
WakeOnLanEnabledFlags = "NONE"
TargetType = "Job"
SubnetMask = "255.255.254.0"
HasFileTransfer = true
has_ssse3 = true
KeyboardIdle = 21660580
CondorPlatform = "$CondorPlatform: x86_64_RedHat6 $"
SlotType = "Dynamic"
OpSysVer = 607
CpuBusyTime = 0
ParentSlotId = 1
StartdIpAddr = "<90.147.168.184:41526>"
RecentDaemonCoreDutyCycle = 0.005225291496099072
StarterAbilityList = "HasTDP,HasFileTransferPluginMethods,HasJobDeferral,HasJICLocalConfig,HasJICLocalStdin,HasPerFileEncryption,HasFileTransfer,HasVM,HasReconnect,HasMPI,HasJava,HasRemoteSyscalls,HasCheckpointing"
ConsumptionCpus = TARGET.RequestCpus
HasReconnect = true
DaemonCoreDutyCycle = 0.02649954356212347
CanHibernate = true
GlobalJobId = "pk.recas.ba.infn.it#10699.7#1551023512"
WithinResourceLimits = ( ifThenElse(TARGET._cp_orig_RequestCpus =!= undefined,TARGET.RequestCpus <= MY.Cpus,MY.ConsumptionCpus <= MY.Cpus) && ifThenElse(TARGET._cp_orig_RequestDisk =!= undefined,TARGET.RequestDisk <= MY.Disk,MY.ConsumptionDisk <= MY.Disk) && ifThenElse(TARGET._cp_orig_RequestMemory =!= undefined,TARGET.RequestMemory <= MY.Memory,MY.ConsumptionMemory <= MY.Memory) )
TotalMemory = 258306
Machine = "wn-recas-uniba-59.recas.ba.infn.it"
UpdatesLost = 0
JobStart = 1551023521
RetirementTimeRemaining = 179655
MaxJobRetirementTime = 3600 * 51
MonitorSelfAge = 21660593
Disk = 9402721
```

Priorità di esecuzione

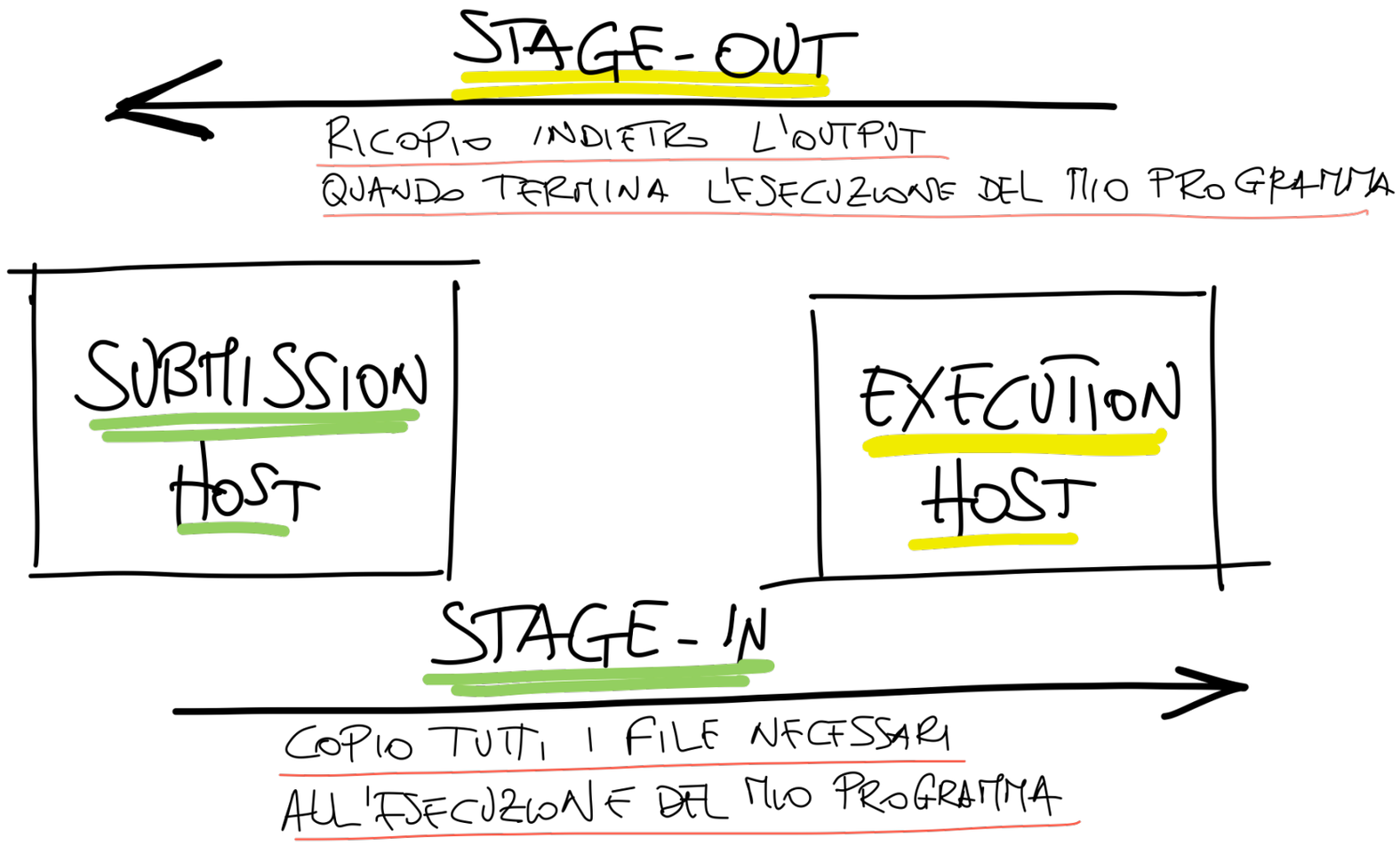
- l'ordine con cui i job vengono eseguiti e' un aspetto importate che il batch system deve saper gestire in modo da garantire un accesso equo alle risorse da parte degli utenti.
  - Utente A sottomette 100k job, subito dopo Utente B sottomette 1 job.
  - Utente B non può aspettare che tutti i 100k job dell'Utente A siano eseguiti prima di vedere il suo unico job in esecuzione
- nell'arco degli anni il fair share si è affermato come meccanismo per il calcolo della priorità di esecuzione dei job.
- Il fair share calcolo dinamicamente la priorita' di un job in base a due parametri
  - lo share di risorse che l'utente/Gruppo può usare
  - la quantità di risorse in termini di WallClockTime



submit file

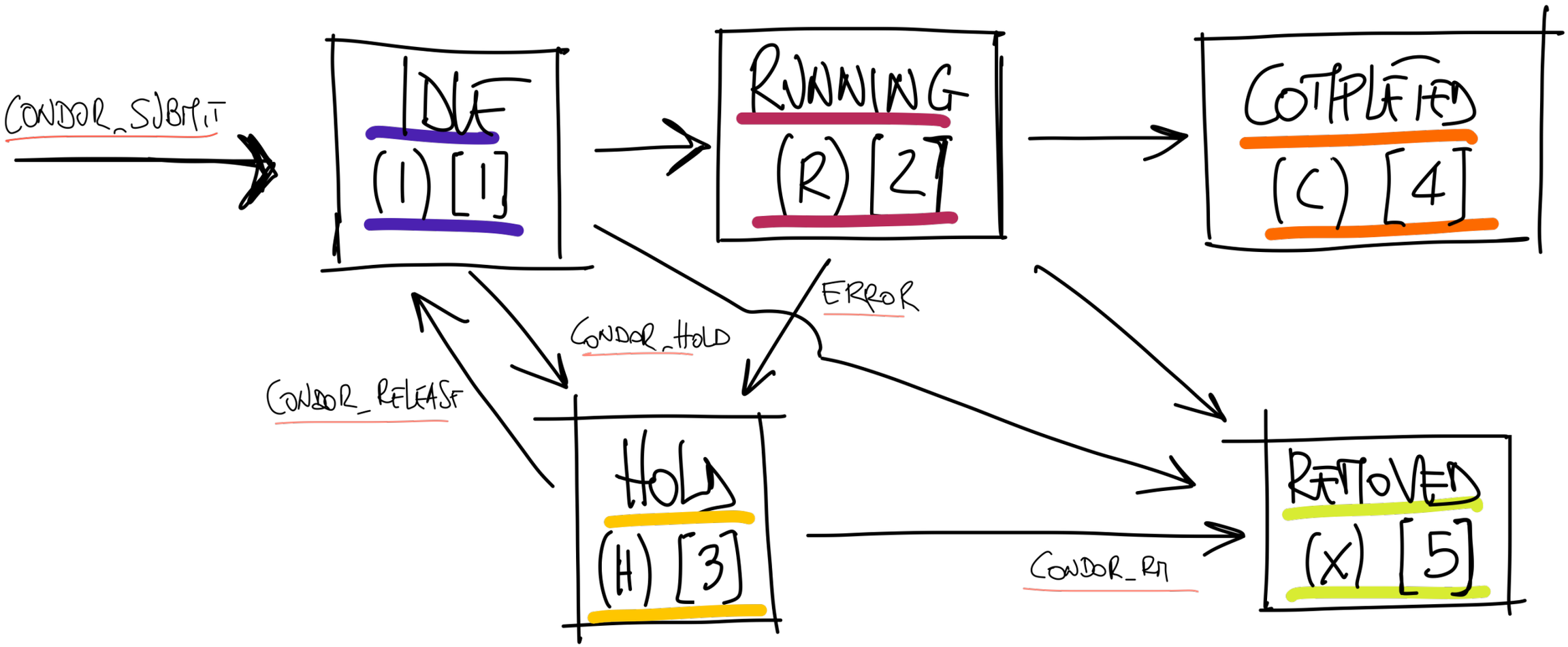
- e' un file che ha lo scopo di istruire HTCondor sulle operazioni da compiere per l'esecuzione del mio programma
- viene usato necessariamente per la sottomissione via command line dei batch job
- un po' come per il classad anche per il subimt file e' prevista una sintassi di tipo
  - attributo = espressione
  - vedi slide successive per l'esempio completo

trasferimento dei file

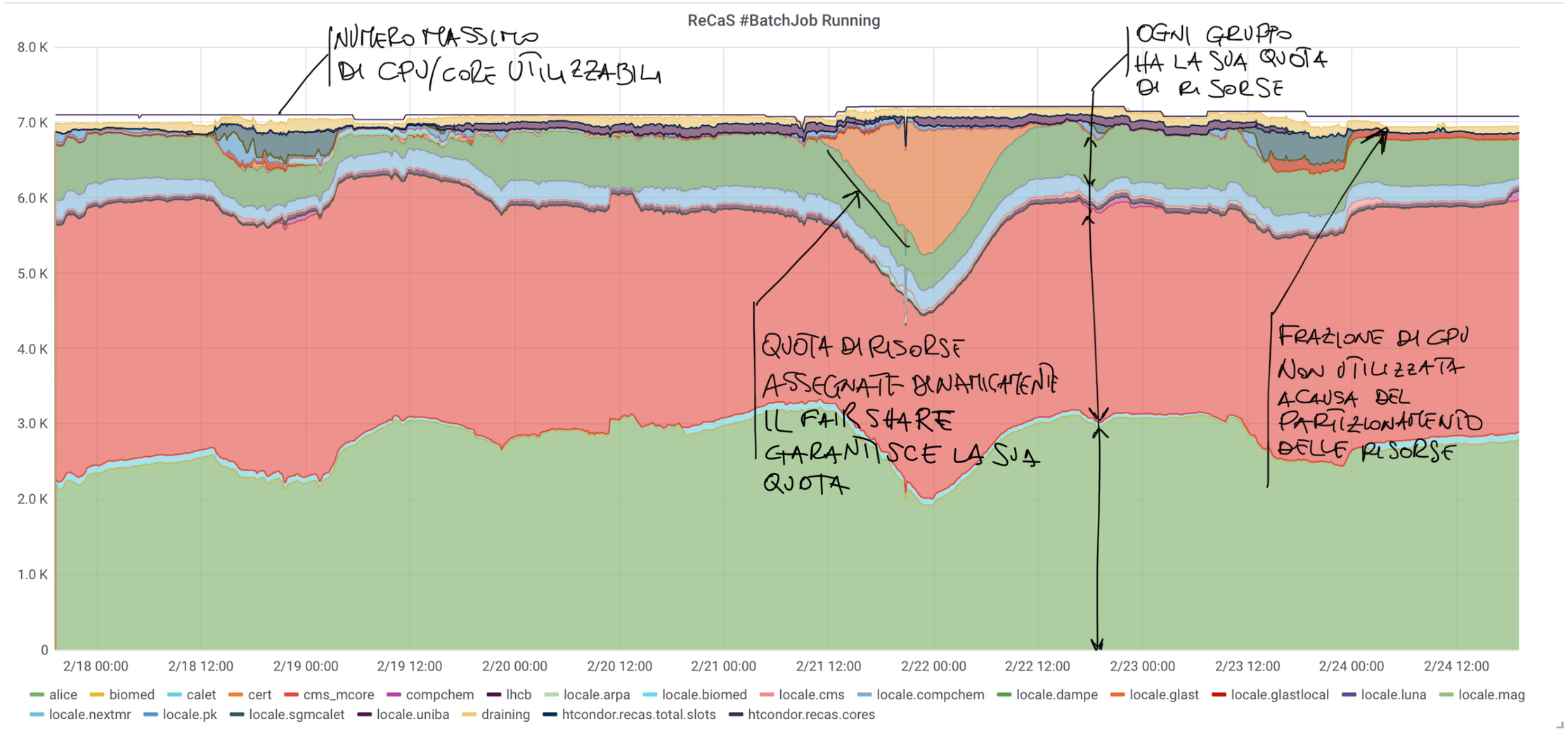


- Necessario perché submission ed execution host sono due macchine distinte, connesse tra di loro attraverso una rete IP.
- Ogni Batch System ha il suo meccanismo di trasferimento di file che è necessario per il funzionamento stesso
- Si può fare a meno di questa funzionalità se c'è a disposizione un filesystem posix condiviso tra tutti i nodi del Batch System

Lo stato di un job



I job running del batch system del DataCenter ReCaS



utilizzare HTCondor

- E' necessario conoscere e saper usare almeno tre comandi per poter iniziare ad utilizzare HTCondor
  - condor_status
    - si usa per chiedere informazioni al batch system relativamente alle risorse disponibili
  - condor_q
    - si usa per chiedere informazioni al batch system relativamente ai job presi in carico
  - condor_submit
    - usato per la sottomissione dei batch job cioè dei nostri programmi che vogliamo mandare in esecuzione su una risorsa accessibile e gestita dal batch system

utilizzare HTCondor

- elencare i collector
  - condor_status -collector
- elencare i negotiator
  - condor_status -negotiator
- elencare gli schedd
  - condor_status -schedd

utilizzare HTCondor

- elencare le risorse disponibili all'interno del batch system
  - condor_status -compact
- visualizzare il classad di una risorsa
  - condor_status -l slot1@wn-infn-3-9-27.recas.ba.infn.it
- visualizzare solo alcuni attributi del classad
  - condor_status slot1@wn-infn-3-9-27.recas.ba.infn.it -af OpSysAndVer MyType MachineResources TotalCpus TotalMemory
- visualizare solo alcune risorse disponibili
  - condor_status -constraint "TotalMemory < 122880" -compact
  - condor_status -constraint "TotalCpus == 64" -compact
  - condor_status -constraint "TotalGpus >= 1" -compact

Definiamo il submit file

```
[italiano@ui02 htcondor]$ cat NumeriPrimi.submit

executable     = NumeriPrimi.py
arguments      = 0 1000

when_to_transfer_output = on_exit
transfer_executable = False


output    = out.$(Cluster).$(Process)
error     = err.$(Cluster).$(Process)
log    = log.$(Cluster).$(Process)


request_cpus = 1
request_memory = 20MB
requirements    = ( TARGET.OpSysAndVer == "CentOS7")


queue 1
```

- specificare l'eseguibile
- specificare gli eventuali argomenti che l'eseguibile richiede
- il commando che alla fine htcondor eseguirà sulla risorsa remota sarà:
  - NumeriPrimi.py 0 1000

Definiamo il submit file

```
[italiano@ui02 htcondor]$ cat NumeriPrimi.submit

executable      = NumeriPrimi.py
arguments       = 0 1000


transfer_executable = False
#transfer_input_files = us.dat, wi.dat
when_to_transfer_output = on_exit


output    = out.$(Cluster).$(Process)
error     = err.$(Cluster).$(Process)
log    = log.$(Cluster).$(Process)


request_cpus = 1
request_memory = 20MB
requirements    = ( TARGET.OpSysAndVer == "CentOS7")


queue 1
```

- HTCondor eventualmente puo' trasferire tutti i file necessari per l'esecuzione del mio programma

- in questo caso specifico istruisco condor per non trasferire l'eseguibile

- Poiche' il mio programma non richiede file d'input non ho necessita' di trasferire alcun input file

- trasferire l'output una volta terminato il job [on_exit] e' il default

Definiamo il submit file

```
[italiano@ui02 htcondor]$ cat NumeriPrimi.submit

executable     = NumeriPrimi.py
arguments      = 0 1000


transfer_executable = False
#transfer_input_files = us.dat, wi.dat
when_to_transfer_output = on_exit


output    = out.$(Cluster).$(Process)
error     = err.$(Cluster).$(Process)
log    = log.$(Cluster).$(Process)


request_cpus = 1
request_memory = 20MB
requirements   = ( TARGET.OpSysAndVer == "CentOS7")


queue 1
```

- output/error per fare il redirect di stdout e stderr sui relativi file
- log invece tenere traccia dei progressi del job
- il nome del file puo' essere parametrizzato

Definiamo il submit file

```
[italiano@ui02 htcondor]$ cat NumeriPrimi.submit

executable     = NumeriPrimi.py
arguments      = 0 1000


transfer_executable = False
#transfer_input_files = us.dat, wi.dat
when_to_transfer_output = on_exit


output    = out.$(Cluster).$(Process)
error     = err.$(Cluster).$(Process)
log    = log.$(Cluster).$(Process)


request_cpus = 1
request_memory = 20MB
requirements   = ( TARGET.OpSysAndVer == "CentOS7")


queue 1
```

- di default il batch system assegna un set di risorse per ogni job.
  - 1 Core, 3GB
- eventualmente si possono richiede in maniera esplicita le risorse necessarie alla esecuzione del programma
- si può richiedere una tipologia specifica di nodo su cui eseguire il proprio job attraverso un'espressione più o meno complessa da utilizzare con l'attributo requirements

Definiamo il submit file

```
[italiano@ui02 htcondor]$ cat NumeriPrimi.submit

executable     = NumeriPrimi.py
arguments      = 0 1000


transfer_executable = False
#transfer_input_files = us.dat, wi.dat
when_to_transfer_output = on_exit


output    = out.$(Cluster).$(Process)
error     = err.$(Cluster).$(Process)
log    = log.$(Cluster).$(Process)


request_cpus = 1
request_memory = 20MB
requirements   = ( TARGET.OpSysAndVer == "CentOS7")


queue 1
```

- definisce il numero di job da sottomettere.
  - Default 1
- molto utile quando si vuole sottomettere un bunch di job. In questo caso basta specificare il numero esatta di job che si vuole sottomettere

Lavoriamo con i job

- sottomettere il submitfile
  - condor_submit -name <schedd name> submit_file_name
- sottomettere un job interattivo
  - condor_submit -name <schedd name> -interactive submit_file_name
  - tipicamente per i job interattivi il submit file contiene solo l'istruzione "queue 1"
- condor_submit ritorna un numerino che e' l'identificativo del job, unico per schedd.
  - Nella forma $(ClusterId).$(ProcId)

Lavoriamo con i job

- controllo lo stato dei job
  - condor_q -name <schedd name> -nobatch
- controllo lo stato di un singolo job
  - condor_q -name <schedd name> <jobid> -nobatch
- visualizzo l'intero classad di un job
  - condor_q -name <schedd name> <jobid> -l

Lavoriamo con i job

- visualizzo solo alcuni attributi del classad.
  - Nell'esempio seguente visualizzo lo stato, l'host su cui il job sta eventualmente runnando e l'ora in cui ha raggiunto lo stato corrente
  - condor_q -name <schedd name> <jobid> -af LastRemoteHost JobStatus 'formatTime(EnteredCurrentStatus, "%H:%M:%S/%d/%m")'
- visualizzo solo alcuni job
  - condor_q -name <schedd_name> -constraint "Group == \"dampe\" && JobStatus == 1"
- visualizzare maggiori dettagli sul perche' il mio job e' ancora in coda in stato "1/Pending/Idle", cioe' la sua esecuzione non e' ancora iniziata.
  - condor_q -name <schedd name> -better-analyze <jobid>

Lavoriamo con i job

- in caso di necessità posso anche modificare alcuni attributi dopo che il job e' stato sottomesso allo schedd
  - condor_qedit -name <schedd> <jobid> RequestMemory 4096
    - cambio la memoria necessaria al mio job a 4GB
- posso anche editare più job alla volta usando il giusto filtro
  - condor_qedit -name <schedd> -constraint "Owner == \"epicardi\"" requirements "(TARGET.OpSysAndVer == \"CentOS7\")"
    - aggiungo un requirement a tutti i job di epicardi. I job devono essere eseguito su macchine con sistema operativo CentOS7

Lavoriamo con i job

- in caso di necessità posso anche collegarmi direttamente sull'host che sta' eseguendo il mio job
  - condor_ssh_to_job -name <schedd> <jobid>
  - avremmo a disposizione una shell attraverso cui fare debug o controllare che tutto sta procedendo come previsto
- I job una volta terminati vengono rimossi dalla coda.
  - A meno che non si istruisca HTCondor per lasciarli in coda per un certo periodo anche dopo la loro terminazione.
- In generale si puo' sempre accedere ai classad una volta che il job viene rimosso dalla coda
  - condor_history -name <schedd> -constraint "Owner == \"epicardi\"" -limit 1
  - visualizzo l'ultimo job terminato dell'utente epicardi

help

- La guida completa la trovate qui
  - https://www.recas-bari.it/index.php/it/recas-bari-partecipazioni/guide-e-manuali
- Siamo sempre raggiungibili via mail, meglio se sentite sempre prima
  - giacinto.donvito@ba.infn.it