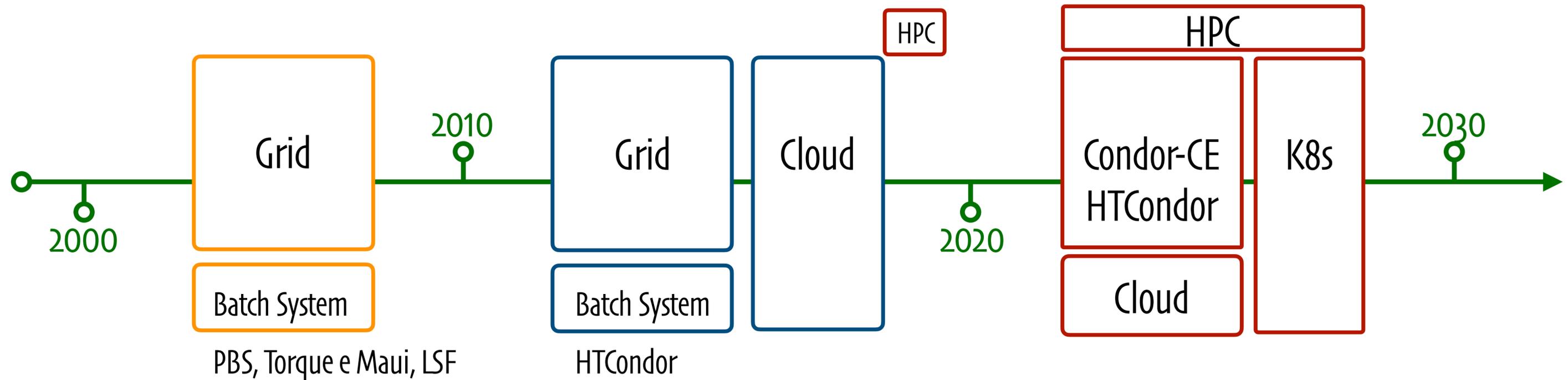


# Batch System[HTCondor] per HPC

Tutorial Days . Pisa . 25 - 27 November 2024

Alessandro Italiano . INFN Bari

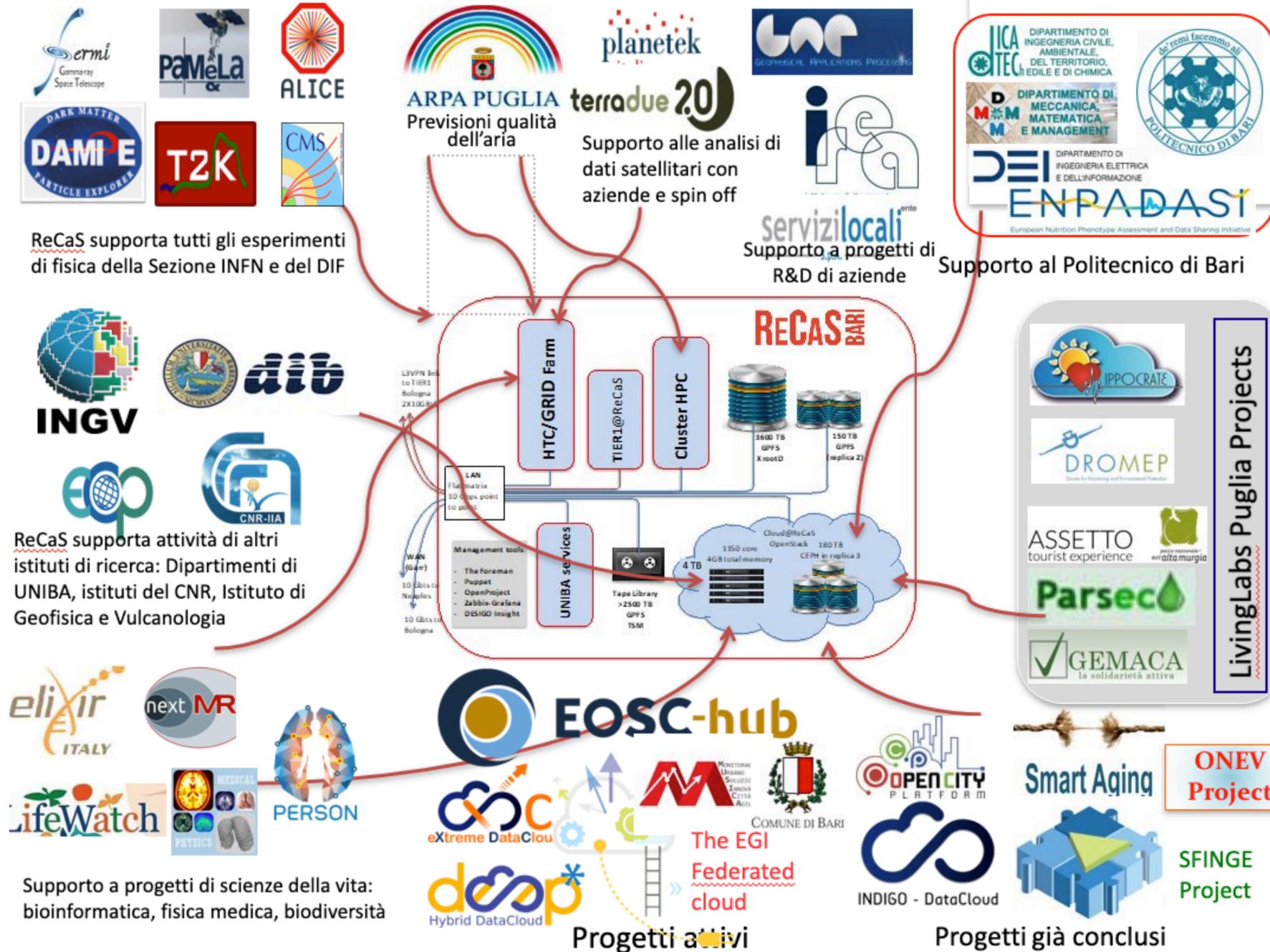
# HTThroughputComputing nell'INFN, il Batch System resiste



# HTC e HPC a "challenging relationship"

- Scenario: risorse di calcolo a disposizione, Utenti che fanno calcolo HTC e HPC
  - Il processo di allocazione, nel caso di contesa delle risorse, non e' banale perche' un job HTC che richiede 1 core "troverà spazio" piu' facilmente rispetto ad un Job HPC che richiede N Core. I Job HTC tenderanno a monopolizzare le risorse disponibili
  - Sara' necessario implementare dei meccanismi che per soddisfare una delle due tipologie di job andranno necessariamente a penalizzarle l'altra tipologia
  - Inoltre i job HPC oltre a richiedere hardware specifico, vedi InfiniBand, hanno la necessita' di essere eseguiti su hardware identico altrimenti il task che va finire su una cpu lenta impiegherà più tempo di un task dello stesso job HPC allocato su una cpu piu' veloce

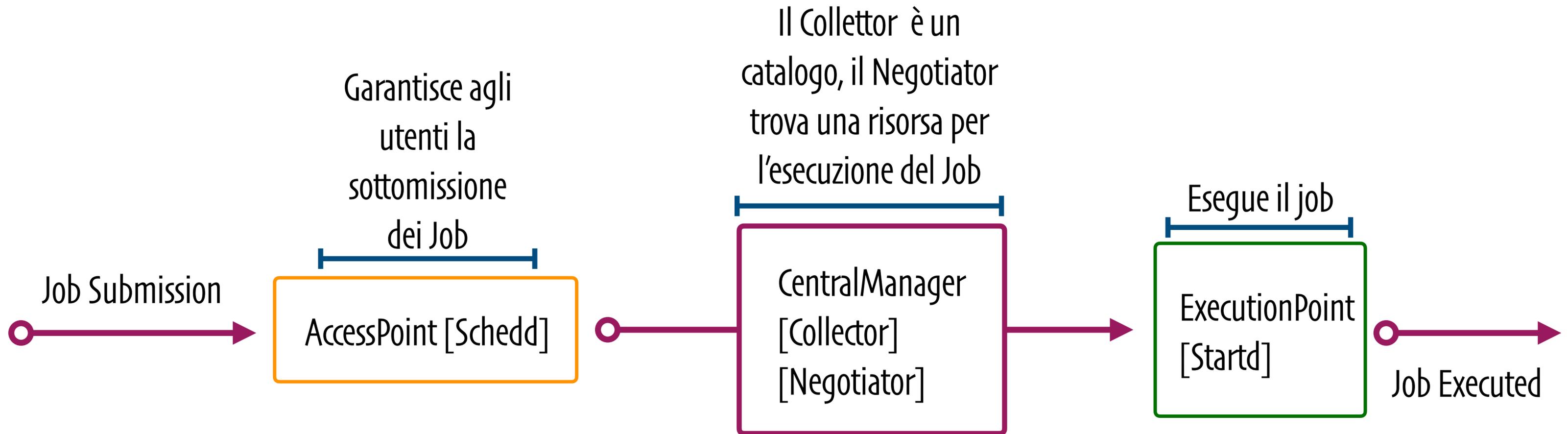
# ReCas-[IBisCO]-Bari Datacenter



# ReCas-[IBisCO]-Bari Datacenter

- la Natura eterogenea degli utenti del DataCenter determina anche le modalità di reperimento delle risorse stesse.
- Tipicamente ogni progetto nel quale il datacenter è coinvolto come ResourceProvider porta con se delle risorse che vengono acquistate in base ai requisiti stessi del progetto che li finanzia.
  - Tale approccio incide decisamente sul livello di eterogeneità delle risorse presenti nel DataCenter
  - Per contro non possiamo permetterci una gestione eterogenea delle risorse che tenderebbe all'uso esclusivo delle risorse.
- A ReCaS tendiamo quindi ad unificare la gestione e l'accesso attraverso il Batch System HTCondor anche per gli "use cases" HPC

# HTCondor Architettura distribuita



Attenzione: l'uso di una qualsiasi funzionalità di HTCondor potrebbe richiedere la sua configurazione su uno o più elementi dell'Architettura

Referenza: <https://htcondor.readthedocs.io/en/latest/admin-manual/introduction-admin-manual.html#the-different-roles-a-machine-can-play>

# Slot Partizionabili

- La differenza banale tra HTC e HPC sta appunto nel numero di core che tipicamente vengono richiesti per questa tipologia di job
  - 1 core/cpu per un job standard HTC
  - N core/cpu per un job standard HPC che potenzialmente possono essere allocati su piu' di un host
- Il BatchSystem, HTCondor nel nostro caso, deve essere in grado quindi di allocare "dinamicamente" le risorse necessarie per soddisfare le richieste/requirements di esecuzione dei job.
- HTCondor alloca dinamicamente le risorse attraverso la funzionalità degli "slot partizionabili"

# Slot Partizionabili

- Ogni ExecutionPoint[startd] ha in esecuzione un processo denominato "condor\_startd" che contiene N "slots"
- Dal punto di vista di HTCondor lo slot è l'entità logica in cui un job può essere eseguito.
- Ad ogni slot HTCondor associa delle risorse in termini di CPU, Disco e Memoria[e anche GPU]
- Affinché un job possa essere eseguito all'interno di uno slot quest'ultimo deve poter soddisfare almeno[>=] le risorse richieste dal job in termini di CPU, Disco e Memoria[e anche GPU]
- come ogni entità all'interno di HTCondor lo slot ha la sua rappresentazione in ClassAd
- Lo slot ha un nome del tipo slot<N>@hostname oppure slot<M\_N>@hostname
- Gli slot possono essere di tre tipi, Statici, Partizionabili e Dinamici

# Slot Partizionabili

- Lo slot statico per sua natura non ha nessuna utilità in questo contesto
  - richiede che ogni slot sia configurato a mano
  - frammentazione alle stelle e quindi risorse inutilizzate
- Lo slot partizionabile invece e' alla base del discorso "multi-tutto"
  - va configurato tipicamente un p-slots per nodo in cui vanno messe tutte le risorse presenti.
  - non viene usato per eseguire job ed il parent slot di eventuali "slot dinamici"
  - viene indicato con il nome "slot<N>@hostname"
- Lo slot dinamico e' creato dinamicamente al bisogno per l'esecuzione di un job prendendo le risorse dallo p-slot parent.
  - Le risorse prese sono quelle necessarie ad eseguire un job in coda
  - questo slot ha il nome "slot<M\_N>@hostname". Dove M e' l'id del p-slots
  - viene usato per l'esecuzione del job e in generale puo' essere rimosso al termine dell'esecuzione del job stesso

# Slot Partizionabili

l'host wn-7-9-2 ha 7 slots di cui un p-slots. Ogni slot ha le sue risorse assegnate sottratte dal parent p-slots

```
[root@htc-ctl-3 ~]# condor_status -startd wn-7-9-2.recas.ba.infn.it
Name                               OpSys      Arch      State      Activity  LoadAv  Mem      ActvtyTime
slot1@wn-7-9-2.recas.ba.infn.it    LINUX      X86_64    Unclaimed  Idle       0.000   937130   2+23:15:08
slot1_1@wn-7-9-2.recas.ba.infn.it  LINUX      X86_64    Claimed    Busy       1.000   16000    0+03:12:47
slot1_2@wn-7-9-2.recas.ba.infn.it  LINUX      X86_64    Claimed    Busy       0.000   4096     1+03:23:09
slot1_7@wn-7-9-2.recas.ba.infn.it  LINUX      X86_64    Claimed    Busy       1.000   16000    0+03:12:48
slot1_8@wn-7-9-2.recas.ba.infn.it  LINUX      X86_64    Claimed    Busy       1.000   16000    0+03:12:48
slot1_9@wn-7-9-2.recas.ba.infn.it  LINUX      X86_64    Claimed    Busy       1.000   16000    0+03:12:47
slot1_14@wn-7-9-2.recas.ba.infn.it LINUX      X86_64    Claimed    Busy       1.990   14080    0+00:48:19

      Total Owner Claimed Unclaimed Matched Preempting Backfill  Drain
X86_64/LINUX      7      0      6      1      0      0      0      0
      Total      7      0      6      1      0      0      0      0
[root@htc-ctl-3 ~]# condor_status -startd wn-7-9-2.recas.ba.infn.it -af Name Cpus Memory Disk
slot1@wn-7-9-2.recas.ba.infn.it 119 937130 417176894
slot1_1@wn-7-9-2.recas.ba.infn.it 1 16000 419696
slot1_2@wn-7-9-2.recas.ba.infn.it 1 4096 419696
slot1_7@wn-7-9-2.recas.ba.infn.it 1 16000 419696
slot1_8@wn-7-9-2.recas.ba.infn.it 1 16000 419696
slot1_9@wn-7-9-2.recas.ba.infn.it 1 16000 419696
slot1_14@wn-7-9-2.recas.ba.infn.it 4 14080 419696
[root@htc-ctl-3 ~]#
```

# Slot Partizionabili

l'host wn-gpu-7-7-28.recas.ba.infn.it ha 3 slots cui un p-slots. Ogni slot ha le sue risorse assegnate tra cui anche una GPU sottratte dal parent p-slots

```
[[root@htc-ctl-3 ~]# condor_status -startd wn-gpu-7-7-28.recas.ba.infn.it
Name                               OpSys      Arch      State      Activity  LoadAv  Mem      ActvtyTime
slot1@wn-gpu-7-7-28.recas.ba.infn.it  LINUX      X86_64    Unclaimed  Idle      0.000  1994121  2+23:20:11
slot1_1@wn-gpu-7-7-28.recas.ba.infn.it LINUX      X86_64    Claimed    Busy      1.000   8192    2+23:09:26
slot1_2@wn-gpu-7-7-28.recas.ba.infn.it LINUX      X86_64    Claimed    Busy      1.000   8192    2+23:10:08

      Total  Owner  Claimed  Unclaimed  Matched  Preempting  Backfill  Drain
X86_64/LINUX    3     0       2         1         0         0         0         0
      Total    3     0       2         1         0         0         0
[[root@htc-ctl-3 ~]# condor_status -startd wn-gpu-7-7-28.recas.ba.infn.it -af Name Cpus Gpus Memory Disk
slot1@wn-gpu-7-7-28.recas.ba.infn.it 254 0 1994121 2001943869
slot1_1@wn-gpu-7-7-28.recas.ba.infn.it 1 1 8192 2005956
slot1_2@wn-gpu-7-7-28.recas.ba.infn.it 1 1 8192 2005956
[[root@htc-ctl-3 ~]#
```

# Slot Partizionabili: config

NodeType: EP

definisco il numero di slot presenti sul nodo

```
NUM_SLOTS = 1
```

definisco il numero di slot di un certo tipo

```
NUM_SLOTS_TYPE_1 = 1
```

assegno le risorse ad ogni tipo di slot

```
SLOT_TYPE_1 = cpus=100%,mem=100%,auto
```

definisco il tipo di job in questo caso p-slots

```
SLOT_TYPE_1_PARTITIONABLE = true
```

Uso una policy per il "consumo" delle risorse del p-slots

```
CONSUMPTION_POLICY = true
```

modifico la policy specifica della Cpu

```
CONSUMPTION_CPUS = TARGET.RequestCpus
```

Passo sempre attraverso un "negotiation cycle"

```
CLAIM_PARTITIONABLE_LEFTOVERS = false
```

# Multicore job, un primo approccio all'HPC

- La definizione di Job Multicore è semplice, un Job che richiede N Core, con  $N > 1$  è considerato tale
- Il supporto ai job multicore è un primo approccio verso le applicazioni HPC, in senso lato naturalmente.
  - Gli N core/cpu che HTCondor mi alloca li posso usare come meglio credo.
  - Posso far partire un'applicazione multithread ma nulla mi vieta di far girare un'applicazione MultiTask di tipo HPC
- A ReCas ad un certo punto ci siamo resi conto che invece di andare dietro a cluster HPC super fighi aveva più senso sfruttare meglio due aspetti a noi facilmente accessibili:
  - il supporto ai Job Multicore che HTCondor fornisce
  - la disponibilità di nuove risorse con specifiche risorse
    - Ogni server è dotato di un numero elevato di core presenti, nell'ordine delle centinaia di core ed N GPU installate
- Abbiamo deciso quindi di supportare applicazioni HPC, utilizzando HTCondor, con le seguenti caratteristiche
  - a basso parallelismo, con  $\#Task \leq \#CorePerServer$
  - richiesta di un  $\#GPU \geq 1$

# Multicore job, un primo approccio all'HPC

- I p-slots da soli non garantiscono che un job multicore vada in esecuzione in tempi brevi anzi potrebbe rimanere inchiodato in coda per diverso tempo
  - lo slot dinamico viene creato al momento dell "negotiation cycle".
  - Se in quel momento non ci sono N core liberi sullo stesso host per soddisfare la richiesta del job MultiCore, lo slot non verrà creato ed il job rimarrà in coda.
  - Anzi, le risorse che non possono essere allocate al job multicore perché non sono sufficienti verranno allocate ad altri job in attesa di essere eseguiti, questo per ogni "negotiator cycle"
- Per risolvere questa questione bloccante a tutti gli effetti, HTCondor tra la sua complessità offre due soluzioni efficaci
  - Il Defrag
  - Una seconda soluzione che non ha un nome ufficiale e che abbiamo chiamato Multi HostPartition[concetto ereditato dall'esperienza con LSF al Tier1]

# HTCondor: Defrag

- il defrag e' un processo che ha un solo compito. Liberare risorse in modo che un job multicores possa trovare "spazio"
- Agisce mettendo in "drain"[svuotando] una serie di p-slots.
  - i job running nei d-slots continuano ad essere eseguiti fino al loro termine liberando le risorse che stavano usando
- Per attivare il defrag occorre:
  - che il processo defrag sia running su uno dei nodi del cluster
  - definire una "defragmentation policy" che definsca cosa e come mettere in drain

NodeType: CM

# Defrag: config

master

prebatch / data / htc9 / nodes / htc-ctl-4.recas.ba.infn.it.yaml



Update htc-ctl-4.recas.ba.infn.it.yaml

Alessandro Italiano authored 18 seconds ago

htc-ctl-4.recas.ba.infn.it.yaml 1.12 KB

```
1 ---
2 recas::profiles::condor::ctl:
3   ha:
4     config:
5       DAEMON_LIST: 'MASTER, COLLECTOR, NEGOTIATOR, HAD, REPLICATION, SHARED_PORT, DEFRAG'
6   defrag:
7     config:
8       DEFRAG_INTERVAL: '60'
9       DEFRAG_DRAINING_MACHINES_PER_HOUR: '200.0'
10      DEFRAG_MAX_CONCURRENT_DRAINING: '150'
11      DEFRAG_SCHEDULE: 'graceful'
12      MAX_DEFRAG_LOG: '104857600'
13      MAX_NUM_DEFRAG_LOG: '10'
14      DEFRAG_RANK: TotalSlotCpus
15      DEFRAG_CANCEL_REQUIREMENTS: '(PartitionableSlot && (Cpus >= 8))'
16      DEFRAG_REQUIREMENTS: '( PartitionableSlot && Offline != True && FileSystemDomain == "GPFS" && \
17                            OpSysAndVer == "CentOS7" && Cpus < 8 ) && \
18                            ( TotalSlotCpus == 40 || TotalSlotCpus == 20 || TotalSlotCpus == 24 || TotalSlotCpus == 16 || \
19                            ( RegExp("wn-2-20", Machine) && TotalSlotCpus == 64 ) || \
20                            ( RegExp("wn-infn-3-9", Machine) && TotalSlotCpus == 64 ) || \
21                            ( RegExp("wn-infn-3-8", Machine) && TotalSlotCpus == 64 ) || \
22                            ( RegExp("wn-infn-3-5", Machine) && TotalSlotCpus == 64 ))'
23
24      GROUP_SORT_EXPR: 'ifThenElse(AccountingGroup=?="<none>", 3.4e+38, ifThenElse(AccountingGroup=?="group_cms.locmcore", -33,
```

- in DAEMON\_LIST va aggiunta la keyword DEFRAG
- DEFRAG\_CANCEL\_REQUIREMENTS parametro delicato perche' definisce quando terminare il processo di drain perche' ho liberato le risorse che mi servono
- DEFRAG\_REQUIREMENTS attraverso un'espressione piu' o meno complessa definisce la lista dei p-slots che si possono mettere in "drain"
- DEFRAG\_SCHEDULE definisce come mettere in "drain" i p-slot, "gracefull" indica che lascia terminare i job che sono in esecuzione

# Defrag: Config

NodeType: EP

```
343 defrag:
344   config:
345     MAXJOBRETIREMETIME: '3600*51'
346     OnlyMulticoreInterval: '3*$(NEGOTIATOR_INTERVAL:60)'
347     IsMulticore: 'RequestCpus >= IfThenElse(Cpus<8,1,8)'
348     IsntUnmatchedPSlot: 'PartitionableSlot!=true || State=="Matched"'
349     OnlyMulticoreJobsAfterDrain: '$(IsntUnmatchedPSlot) || $(IsMulticore) || $(StateTimer) > $(OnlyMulticoreInterval)'
350     #START: '$(START) && ( $(OnlyMulticoreJobsAfterDrain) )'

301 recas::profiles::condor::startd:
302   node:
303     config:
304       STARTD_DEBUG: 'D_FULLDEBUG'
305       NUM_CPUS: '%{recas::profiles::condor::c_slots}'
306       DAEMON_LIST: 'MASTER, STARTD'
307       #START: 'IfThenElse(TotalSlotCpus == 64, localMountsReady == True && $(OnlyMulticoreJobsAfterDrain), localMountsReady == True)'
308       #START: '$(OnlyMulticoreJobsAfterDrain))'
309       #START: 'TotalSlots < 54'
310       #START: 'true'
311       START: '(localMountsReady == True && $(OnlyMulticoreJobsAfterDrain))'
312       #START: '(localMountsReady == True && Group != "usi" && $(OnlyMulticoreJobsAfterDrain))'
313       #RANK: 'TotalCpus - Cpus'
314       RANK: 'Memory'
315       SUSPEND: 'false'
316       PREEMPT: 'false'
317       KILL: 'false'
318       NUM_SLOTS: '1'
319       NUM_SLOTS_TYPE_1: '1'
320       SLOT_TYPE_1: 'cpus=100%,mem=100%,auto'
321       SLOT_TYPE_1_PARTITIONABLE: 'true'
322       CONSUMPTION_POLICY: 'true'
323       CONSUMPTION_CPUS: 'TARGET.RequestCpus'
324       SLOT_WEIGHT: 'Cpus'
```

# Considerazioni sul Defrag

- Abbiamo visto come il defrag
  - mette in drain dei p-slot scegliendo da una lista di nodi definita da "DEFRAG\_REQUIREMENTS"
  - il processo di drain di un p-slots viene terminato quando il parametro "DEFRAG\_CANCEL\_REQUIREMENTS" ritorna True.
    - A ReCaS questo parametro e' True quando si liberano 8 core e quindi il defrag viene usato esclusivamente per i job multicores degli esperimenti LHC
  - tale aspetto ha un "side effect", questa configurazione non garantisce di liberare risorse per job che richiedono più di N core

# HTCondor HostPartition

- il concetto alla base e' la divisione in gruppi degli host che fanno parte del cluster HTCondor
- Nel caso specifico di ReCaS abbiamo definito due HostPartion
  - quella HTC per eseguire job che richiedo al massimo 8 core
  - quella HPC per eseguire job che richiedono un "basso parallelismo" o delle GPU
- per l'utente la scelta tra le due partizioni e' trasparente, HTCondor attraverso una configurazione specifica decide dove mandare in esecuzione i job
- In particolare sfruttiamo la funzionalità del "Job Transform" per andare a modificare/aggiungere qualche classAd utile alla causa

# HostPartition: config

NodeType: AP

```
61 transform:
62   config:
63     SCHEDD_CLASSAD_USER_MAP_NAMES: '$(SCHEDD_CLASSAD_USER_MAP_NAMES) AcctGroupMap AcctGroupMapHPC'
64     CLASSAD_USER_MAPFILE_AcctGroupMap: '/etc/condor/ReCaS_Accounting_Map'
65     CLASSAD_USER_MAPFILE_AcctGroupMapHPC: '/etc/condor/ReCaS_Accounting_Map_HPC'
66     JOB_TRANSFORM_NAMES: 'LeaveJobInQueue AccountingGroup HPC LeaveJobInQueuecbrugnoni'
67     JOB_TRANSFORM_LeaveJobInQueuecbrugnoni: '[ Requirements = Owner == "cbrugnoni"; set_LeaveJobInQue
68     JOB_TRANSFORM_LeaveJobInQueue: '[ set_LeaveJobInQueue = ( JobStatus == 4 && ((CurrentTime - Comp
69     JOB_TRANSFORM_AccountingGroup: '[ copy_AccountingGroup = "RequestedAccGroup"; eval_set_Accounting
70     JOB_TRANSFORM_HPC: '[ Requirements = RequestCpus > 4 || RequestGPUs > 0; \
71                       copy_AccountingGroup = "HTCAccGroup"; set_HPC = True; \
72                       eval_set_AccountingGroup = userMap("AcctGroupMapHPC",Owner);]
73     PROTECTED_JOB_ATTRS: '$(PROTECTED_JOB_ATTRS) AccountingGroup'
74     SUBMIT_REQUIREMENT_NAMES: '$(SUBMIT_REQUIREMENT_NAMES) AccountingGroup'
75     SUBMIT_REQUIREMENT_AccountingGroup: 'AccountingGroup isnt undefined'
76     SUBMIT_REQUIREMENT_AccountingGroup_REASON: 'Could not map to a valid AccountingGroup'
77
```

JOB\_TRANSFORM\_HPC  
viene applica ai soli  
job che soddisfano alcuni  
requisiti, aggiungo un  
ClassAd "HPC = True" oltre  
all'AccountingGroup

# HostPartition: config

## NodeType: EP

```
9
10 recas::profiles::condor::startd:
11   gpu:
12     config:
13       GPU_DISCOVERY_EXTRA: '-extra'
14       use feature: 'GPUs'
15   node:
16     config:
17       SLOT_TYPE_1: 'cpus=100%,GPUs=100%,mem=98%,auto'
18       #START: '(localMountsReady == True && Group != "asi" && Owner != "sgmalice010" && Group != "pk" && $(OnlyMulticoreJobsAfterDrain))'
19       START: 'localMountsReady == True && HPC == true'
20
21
```

- Configurazione specifica per uno dei nodi che appartengono alla partizione HPC
- Tra le risorse dello p-slots ho aggiunto tutte le GPU presenti sul nodo
- All'espressione START ho aggiunto "HPC = True" in modo che su questo nodo/p-slots possano essere eseguiti solo i jobHPC

# HostPartition: usage

```
[[root@htc-ctl-3 ~]# condor_status -pr status_pslots.cpf | grep -e "Machine\\"gpu
Machine Platform Condor Cpus Free Gpus Free Rank Mem(Gb) MaxMem FreeMem% FreeMem DSlots CpuUtil Jobs/10Min
wn-gpu-7-7-28.recas.ba.infn.it CentOS7_ 9.0.17 256 6 2 0 191753 2003.45 204800 9.3 187.3 13 0.07 0.00
wn-gpu-7-7-30.recas.ba.infn.it CentOS7_ 9.0.17 256 2 2 0 64905 2003.45 204800 3.2 63.4 13 0.18 0.50
wn-gpu-7-7-32.recas.ba.infn.it CentOS7_ 9.0.17 256 14 2 0 130057 2003.45 204800 6.3 127.0 12 0.12 0.00
wn-gpu-7-8-30.recas.ba.infn.it CentOS7_ 9.0.17 256 0 2 0 1060233 2003.45 512000 51.7 1035.4 8 0.08 1.00
wn-gpu-7-8-32.recas.ba.infn.it CentOS7_ 9.0.17 256 6 2 0 142217 2003.45 204800 6.9 138.9 14 0.15 0.00
wn-gpu-7-9-28.recas.ba.infn.it CentOS7_ 9.0.17 256 4 2 0 1863049 2003.45 131072 90.8 1819.4 3 0.98 0.00
wn-gpu-7-9-30.recas.ba.infn.it CentOS7_ 9.0.17 256 14 2 0 351113 2003.45 204800 17.1 342.9 12 0.05 3.50
wn-gpu-7-9-32.recas.ba.infn.it CentOS7_ 9.0.17 256 14 2 0 191881 2003.45 204800 9.4 187.4 12 0.10 0.00
wn-gpu-8-3-2.recas.ba.infn.it CentOS7_ 9.0.17 255 255 1 1 2010698 2003.65 undefi 98.0 1963.6 0 0.00 0.00
wn-gpu-8-3-6.recas.ba.infn.it CentOS7_ 9.0.17 255 2 1 0 167498 2003.65 278528 8.2 163.6 12 0.04 0.50
wn-gpu-8-3-10.recas.ba.infn.it CentOS7_ 9.0.17 255 6 1 0 1792202 2003.65 204800 87.4 1750.2 8 0.74 23.00
wn-gpu-8-3-14.recas.ba.infn.it CentOS7_ 9.0.17 255 10 1 0 1387338 2003.65 204800 67.6 1354.8 7 0.61 22.00
wn-gpu-8-3-18.recas.ba.infn.it CentOS7_ 9.0.17 255 14 1 0 81354 2003.65 278528 4.0 79.4 12 0.25 0.00
wn-gpu-8-3-22.recas.ba.infn.it CentOS7_ 9.0.17 255 14 1 1 1998410 2003.65 12288 97.4 1951.6 1 0.73 2.50
wn-gpu-8-8-28.recas.ba.infn.it CentOS7_ 9.0.17 256 5 3 0 1953033 2003.45 32768 95.2 1907.3 6 0.83 3.00
wn-gpu-8-8-30.recas.ba.infn.it CentOS7_ 9.0.17 256 5 3 0 1953033 2003.45 32768 95.2 1907.3 6 0.84 3.00
wn-gpu-8-8-32.recas.ba.infn.it CentOS7_ 9.0.17 256 13 3 0 1916297 2003.45 32768 93.4 1871.4 14 0.22 28.00
[[root@htc-ctl-3 ~]#
```

```
wn-7-6-10.recas.ba.infn.it CentOS7_ 9.0.17 128 0 6826 995.42 65536 0.7 6.7 27 0.71 0.00
wn-7-6-11.recas.ba.infn.it CentOS7_ 9.0.17 128 1 96170 995.42 65536 9.4 93.9 29 0.57 2.00
wn-7-6-21.recas.ba.infn.it CentOS7_ 9.0.17 128 0 34602 995.42 65536 3.4 33.8 30 0.73 0.00
wn-7-6-22.recas.ba.infn.it CentOS7_ 9.0.17 128 0 407722 995.42 65536 40.0 398.2 20 1.00 0.00
wn-7-6-23.recas.ba.infn.it CentOS7_ 9.0.17 128 0 259114 995.42 65536 25.4 253.0 23 0.77 0.00
wn-7-6-24.recas.ba.infn.it CentOS7_ 9.0.17 128 0 14762 995.42 65536 1.4 14.4 30 0.67 4.50
wn-7-6-25.recas.ba.infn.it CentOS7_ 9.0.17 128 0 255146 995.42 65536 25.0 249.2 23 0.62 0.00
wn-7-6-26.recas.ba.infn.it CentOS7_ 9.0.17 128 0 255146 995.42 65536 25.0 249.2 23 0.70 0.00
wn-7-6-27.recas.ba.infn.it CentOS7_ 9.0.17 128 0 30634 995.42 65536 3.0 29.9 30 0.61 0.00
wn-7-6-28.recas.ba.infn.it CentOS7_ 9.0.17 128 1 117162 995.42 65536 11.5 114.4 26 1.24 13.50
wn-7-6-29.recas.ba.infn.it CentOS7_ 9.0.17 128 0 34602 995.42 65536 3.4 33.8 30 0.66 4.50
wn-7-7-2.recas.ba.infn.it CentOS7_ 9.0.17 128 0 38570 995.42 65536 3.8 37.7 30 1.04 1.00
wn-7-7-3.recas.ba.infn.it CentOS7_ 9.0.17 128 0 34602 995.42 65536 3.4 33.8 30 0.80 5.50
wn-7-7-4.recas.ba.infn.it CentOS7_ 9.0.17 128 0 22698 995.42 65536 2.2 22.2 30 0.64 0.00
wn-7-7-5.recas.ba.infn.it CentOS7_ 9.0.17 128 0 34602 995.42 65536 3.4 33.8 30 0.55 1.00
wn-7-7-6.recas.ba.infn.it CentOS7_ 9.0.17 128 0 145578 995.42 65536 14.3 142.2 30 0.50 0.00
wn-7-7-7.recas.ba.infn.it CentOS7_ 9.0.17 128 1 92202 995.42 65536 9.0 90.0 29 0.90 8.50
```

# HPC MultiHost

- Naturalmente i nostri utenti eterogenei aumentato e con loro richieste di poter eseguire job MPI standard, quindi anche con la possibilità che i core allocati siano distribuiti su più host.
- HTCondor supporta anche questo use-case con uno "Universe[Execution Environment]" dedicato denominato appunto "Parallel"
- Il supporto di questo Universe all'interno di un cluster HTCondor richiede delle risorse dedicate
  - un AP[AccessPoint][schedd] dedicato da cui sottomettere i job MPI
  - degli EP[ExecutionPoint][startd] dedicati al solo uso esclusivo di eseguire i job sottomessi dal AP dedicato
- Lo "universe supporta" due modi diversi per far partire l'applicazione MPI
  - uno standard e quindi HTCondor si preoccupa di generare l'hostfile che poi mpirun usa per sapere dove e quanti task far partire
  - in alternativa HTCondor può essere istruito a far partire N istanze dello stesso eseguibile su M nodi diversi

# MPI MultiHost: config

- I job che richiedono lo "universe" Parallel si ritrovano in automatico il seguente ClassAd
  - Scheduler = "DedicatedScheduler@full.host.name"
- Sull'EP[Startd] la configurazione si riduce alle seguenti righe
  - definisco una variabile[knob] con il nome dello scheduler dedicato ai job MPI
    - DedicatedScheduler = "DedicatedScheduler@full.host.name"
  - Aggiungo questa variabile tra quelle annunciate dal startd
    - STARTD\_ATTRS = \$(STARTD\_ATTRS), DedicatedScheduler
  - La Variabile START impone che, solo i job che arrivano dallo scheduler dedicato possono essere eseguiti
    - START = Scheduler =?= \$(DedicatedScheduler)

# MPI MultiHost: setup

- i Nodi che devono eseguire job MPI richiedono un setup aggiuntivo
- e' necessario attivare autenticazione HostBased per ssh in modo che non sia necessario inserire la password
  - mpirun quando parte si collega via ssh sui nodi designati per partire i vari task
- storicamente il job MPI hanno usato una connessione a bassa latenza ed in particolare infiniband
  - e' quindi necessario dotarsi di uno switch infiniband, collegarci le macchine e accertarsi che i vari nodi dedicati si raggiungano anche sulla rete infiniband
  - il requirements su infiniband, a mio avviso, sta diventando sempre piu' debole perche' le reti Ethernet diventano sempre piu' veloci 10/25/40/100 Gb/s abbassando di molto la latenza

# MPI MultiHost: setup ssh

istruzioni puppet per configurare HostBased ssh access senza inserire la pwd

```
19 file { 'shosts.equiv':
20     path      => '/etc/ssh/shosts.equiv',
21     source    => 'puppet:///modules/htcondor/shosts.equiv',
22     ensure    => 'present',
23 }
24
25
26 file { 'ssh_config':
27     path      => '/etc/ssh/ssh_config',
28     source    => 'puppet:///modules/htcondor/ssh_config',
29     ensure    => 'present',
30     mode      => '0644',
31 }
32
33 file { 'sshd_config':
34     path      => '/etc/ssh/sshd_config',
35     source    => 'puppet:///modules/htcondor/sshd_config',
36     ensure    => 'present',
37     mode      => '0644',
38 }
39
40 service { 'sshd':
41     subscribe => [
42         File["ssh_config"],
43         File["sshd_config"],
44     ]
45 }
```

46

```
    $str = "#!/bin/bash
rm -f /etc/ssh/ssh_known_hosts;
for i in `cat /etc/ssh/shosts.equiv`; do /usr/bin/ssh-keyscan -t rsa,dsa `cat /etc/ssh/shosts.equiv`; done
"

file { '/root/ssh-keyscan':
    content    => $str,
    ensure     => 'present',
    mode       => '0777',
}

exec { 'ssh-keyscan':
    command    => '/root/ssh-keyscan',
    subscribe  => File['shosts.equiv'],
    refreshonly => true,
    require    => File['/root/ssh-keyscan'],
    provider   => shell
}

create_resources('ipmi::user', lookup('ipmi_users'))
$ip_ipmi = lookup('ip_ipmi')
$ip = inline_template("<% @ip_ipmi.each do |key, value| %> <% if key == @fqdn %> <%= value %> <% end -%> <% end %>")

create_resources('ipmi::network', 'lan1' => {
    type      => 'static',
    ip        => $ip,
    netmask   => '255.255.240.0',
    gateway   => '172.16.0.1',
    lan_channel => 1,
})
}
```

74  
75  
76  
77  
78

# MPI MultiHost: setup ib

- Il Riferimento per l'opensource su infiniband e' OFED[Open Fabrics Enterprise Distribution][<https://www.openfabrics.org/>]
- E' disponibile un bundle con tutta una serie di software e driver per l'hardware installato nei server
- wget [https://content.mellanox.com/ofed/MLNX\\_OFED-5.8-4.1.5.0/MLNX\\_OFED\\_LINUX-5.8-4.1.5.0-rhel7.9-x86\\_64.tgz](https://content.mellanox.com/ofed/MLNX_OFED-5.8-4.1.5.0/MLNX_OFED_LINUX-5.8-4.1.5.0-rhel7.9-x86_64.tgz)
- ./mlnxofedinstall --add-kernel-support --skip-repo
- ho anche installato i seguenti pacchetti opensm infiniband-diags
- "systemctl start opensm" per far partire un IB NetworkManager. Evidentemente lo switch non era configurato per farlo

```
[root@wn-gpu-7-7-32 ~]# ibstat
CA 'mlx5_0'
  CA type: MT4123
  Number of ports: 1
  Firmware version: 20.36.1010
  Hardware version: 0
  Node GUID: 0x08c0eb03005495fe
  System image GUID: 0x08c0eb03005495fe
  Port 1:
    State: Active
    Physical state: LinkUp
    Rate: 100
    Base lid: 9
    LMC: 0
    SM lid: 1
    Capability mask: 0xa659e848
    Port GUID: 0x08c0eb03005495fe
    Link layer: InfiniBand
[root@wn-gpu-7-7-32 ~]#
```

```
[root@wn-gpu-7-9-30 ~]# ibnodes
Ca: 0x08c0eb03005495ca ports 1 "wn-gpu-7-9-32 mlx5_0"
Ca: 0x08c0eb03005495ce ports 1 "wn-gpu-7-9-28 mlx5_0"
Ca: 0x08c0eb0300549602 ports 1 "wn-gpu-7-8-30 mlx5_0"
Ca: 0x08c0eb03005495d2 ports 1 "wn-gpu-7-8-32 mlx5_0"
Ca: 0x08c0eb03005495fe ports 1 "wn-gpu-7-7-32 mlx5_0"
Ca: 0x08c0eb03005497e2 ports 1 "wn-gpu-7-7-30 mlx5_0"
Ca: 0x08c0eb03005497de ports 1 "wn-gpu-7-7-28 mlx5_0"
Ca: 0x08c0eb030054a746 ports 1 "wn-gpu-7-9-30 mlx5_0"
Switch : 0x08c0eb0300c8f0d4 ports 41 "MF0;switch-b60a38:
MQM8700/U1" enhanced port 0 lid 6 lmc 0
[root@wn-gpu-7-9-30 ~]#
```

```
[root@wn-gpu-7-7-32 ~]# ibping -S
[root@wn-gpu-7-9-30 ~]# ibping -G 0x08c0eb03005495fe -P 1
Pong from wn-gpu-7-7-32.recas.ba.infn.it.(none) (Lid 9): time 0.292 ms
Pong from wn-gpu-7-7-32.recas.ba.infn.it.(none) (Lid 9): time 0.237 ms
Pong from wn-gpu-7-7-32.recas.ba.infn.it.(none) (Lid 9): time 0.209 ms
Pong from wn-gpu-7-7-32.recas.ba.infn.it.(none) (Lid 9): time 0.234 ms
Pong from wn-gpu-7-7-32.recas.ba.infn.it.(none) (Lid 9): time 0.204 ms
Pong from wn-gpu-7-7-32.recas.ba.infn.it.(none) (Lid 9): time 0.237 ms
Pong from wn-gpu-7-7-32.recas.ba.infn.it.(none) (Lid 9): time 0.298 ms
```

# MPI MultiHost: setup software

- UCX è un framework per protocolli di comunicazione per applicazioni HPC
- In sostanza UCX si interpone tra MPI e l'hardware, IB nel nostro caso, per ottenere una migliore comunicazione tra i task dell'applicazione MPI
- UCX si scarica e si compila che credo sia la cosa migliore perchè lo compili con i driver dell'hardware installato nel server

```
$ wget https://github.com/openucx/ucx/releases/download/v1.17.0/ucx-1.17.0.tar.gz
$ tar xzf ucx-1.17.0.tar.gz
$ cd ucx-1.17.0
$ mkdir build
$ cd build
$ ../configure --prefix=<ucx-install-path>
$ make -j4
$ make install
```

- a questo punto non rimane che compilare open-mpi facendo riferimento UCX in modo che MPI possa usare l'implementazione locale infiniband

```
$ git clone https://github.com/open-mpi/ompi.git
$ cd ompi
$ ./autogen.pl
$ mkdir build-ucx
$ cd build-ucx
$ ../configure --prefix=<ompi-install-path> --with-ucx=<ucx-install-path>
$ make
$ make install
```

# Multi Startd

- A Bari il numero di server che arrivano con Centinaia di CPU e diverse GPU installate è in trend positivo da un po' di anni oramai.
- Con questa tipologia di server allocati ad un "dedicated scheduler" per job MPI MultiHost si rischia di tornare al via
  - rischio di non usare le GPU perchè non sono presenti nella "hostpartition HPC"
  - ad ogni job che richiede una GPU viene assegnato un numero di CPU pari almeno al default togliendole quindi dalla disponibilità dei job "MPI MultiHost" oltre al fatto che questi job potrebbero essere allocati prima
- HTCondor tra la sua flessibilità offre la possibilità di definire sulla stesso server più startd
  - uno startd con le GPU e un #CPU pari alle GPU da insieme nell'hostpartition HPC
  - un altro startd con le rimanenti CPU da agganciare al "dedicated scheduler"

# Multi Startd

```
# Define the first STARTD on this machine
STARTD1 = $(STARTD)
STARTD1_ARGS = -f -local-name S1
STARTD1_LOG = $(LOCAL_DIR)/log/StartdLog.1
STARTD1_EXECUTE = $(LOCAL_DIR)/execute.1
STARTD.S1.NUM_SLOTS = 1
STARTD.S1.NUM_SLOTS_TYPE_1 = 1
STARTD.S1.SLOTS_TYPE_1 = gpus=100%,cpus=2%,mem=1/9,auto
STARTD.S1.STARTD_NAME = wn-x-y-z-gpu
STARTD.S1.STARTD_LOG = $(STARTD1_LOG)
STARTD.S1.STARTD_EXEUTE = $(STARTD1_EXECUTE)

DAEMON_LIST = $(DAEMON_LIST), STARTD1

# Define the second STARTD on this machine
STARTD2 = $(STARTD)
STARTD2_ARGS = -f -local-name S2
STARTD2_LOG = $(LOCAL_DIR)/log/StartdLog.2
STARTD2_EXECUTE = $(LOCAL_DIR)/execute.2
STARTD.S2.NUM_SLOTS = 1
STARTD.S2.NUM_SLOTS_TYPE_1 = 1
STARTD.S2.SLOTS_TYPE_1 = gpus=0,cpus=98%,mem=8/9,auto
STARTD.S2.STARTD_NAME = wn-x-y-z-cpu
STARTD.S2.STARTD_START = Scheduler =?= $(DedicatedScheduler)
STARTD.S2.STARTD_LOG = $(STARTD2_LOG)
STARTD.S2.STARTD_EXEUTE = $(STARTD2_EXECUTE)

DAEMON_LIST = $(DAEMON_LIST), STARTD2
DC_DAEMON_LIST = +STARTD1 STARTD2
```

- sullo stesso nodo faccio partire due Startd.
- Ad uno startd associo tutte le gpu all'altro solo le CPU rimanenti.
- Lo startd con cpu lo assegno al "dedicated scheduler" tramite l'espressione START
- **ATTENZIONE, configurazione non verificata**