

Leonardo Opportunistico stato e prossimi passi

Tommaso/Daniele

Recap stato attuale

Sono stati ripristinati, migliorati dove possibile e testati tutti e quattro gli assets come nelle precedenti macchine (A2 e M100)

- Runtime environment
- Rete - outbound connectivity
- Accesso dati esperimento da CNAF e da federazioni
- Meccanismo pressione per l'acquisizione dei nodi

Nodo edge dedicato

Nel contesto delle attività ICSC (Spoke0) abbiamo ottenuto un nodo di edge al CINECA (icsc01-ext.leonardo.cineca.it + IP interno CINECA)

- È configurato come un nodo di login accessibile ssh solo da CNAF
- Lo usiamo per gestire scripts per la pressione
- Ci serve per integrare la parte cloud infn con leonardo via offloading
 - Ci facciamo girare interLink

Non siamo amministratori è solo un nodo dedicato che può offrirci qualche maniglia per far funzionare i nostri workflow

- Vedi dopo

Runtime environment

CVMFS con afuse

- Il setup di Leonardo è fatto per isolare tutto con "job_container" slurm
- Per montare il client serve una soluzione tutta in userspace, perchè interna al mount namespace della shell istanziata da slurm
 - Hanno proposto di usare questo <https://github.com/pcarrier/afuse>
 - Tutto automatizzato usando una convenzione per il job name
 - `srun -t 1400 -A inf24_lhc_2 --pty -p dcgp_usr_prod --ntasks=1 -J AA_CVMFS_BB --cpus-per-task=1 bash`

Abbiamo montato i repositories CERN e fatto configurare i repo pubblici INFN (datacloud.infn.it e unpacked.infn.it)

- Ci serve per le dipendenze WN
 - `source /cvmfs/datacloud.infn.it/repo/wn-leonardo/v1/etc/profile.d/wn-leonardo-setup.sh`
 - In questo modo, il pilot gira sul sistema standard e NON dentro un container singularity (M100)
- Abbiamo un CVMFS publisher (VM su cloud CNAF) disponibile per pubblicare quello che serve sui repo INFN

Outbound connectivity

Sui nodi slurm (sia GP che Booster) sono aperti outbound

- Tutto il CERN “computing center”
- Tutto il CNAF, sia il Tier1 che la parte Cloud
- Nodi selezionati FNAL (per accedere a servizi CMS – non indispensabile ma utile)

È un pochino di più di quanto avevamo su M100 e A2. Se serve qualcosa ancora credo si possa negoziare

- Dovrebbe sicuramente bastare per gli LHC con la limitazione nota dell'accesso ai dati fuori dal Tier1 CNAF (vedi dopo)

Accesso ai dati

Lettura e scrittura su Tier1 CNAF testato e funziona

- Per CMS stiamo usando:
 - `davs://xfer-cms.cr.cnaf.infn.it:8443/cmsdisk`
- Per leggere dalla federazione abbiamo ripristinato il proxy xrootd. Per ora una VM su cloud CNAF (non carrozzata)

La logica per l'accesso ai dati in cms è: Priorità su CNAF e fallback sulla federazione passando dal proxy

```
8 <data-access>
9   <catalog volume="CNAF_GPFS" protocol="xrootd_local"/>
10  <catalog volume="CNAF_GPFS" protocol="XRootD-cache"/>
11 </data-access>
12 </source-config>
```

```
{
  "protocol": "xrootd_local",
  "access": "global-ro",
  "prefix": "root://xrootd-cms-redirector.cr.cnaf.infn.it:1094/"
},
{
  "protocol": "XRootD-cache",
  "access": "global-ro",
  "prefix": "root://131.154.98.28:1094/"
}
```

Questo è CMS specific ma il “setup del proxy” può essere replicato per tutti

Meccanismo di pressione / creazione pilot

- NON usiamo al momento jobs pilot sottomessi tramite il CE (prossimo passo), ma usiamo il meccanismo dei **manual_glidein**; in questo modo serve sottomissione di jobs SLURM che facciano partire pilots
 - GitHub: https://github.com/tommasoboccali/leonardo_opportunistic
- Meccanismo di pressione che funziona su GP e Booster, basato su files di configurazione
 - `{"jobexecutor": "./slurm_glidein_dcgp_lowprio.job", "log_prefix": "logs/gp_job_log_", "jobname": "GP_CMS_GLIDEIN_CVMFS_JOB", "max_running": 100, "max_idle": 5, "max_hours": 20000}`
 - Meccanismo dinamico: modificando il file, si possono cambiare runtime i parametri (per esempio aumentare / diminuire il numero di jobs running / in attesa)
 - **In realizzazione** meccanismo che lo faccia automaticamente guardando quanti pilot sono agganciati dal payload (quindi auto-tune del numero di jobs)
- **Importante:** “_lowprio”: abbiamo avuto accesso a una coda low priority, NON accountata (“gratis”) → possiamo implementare un meccanismo davvero opportunistico, alla backfill

Tests con meccanismo di pressione

```
[tboccali@login07 leonardo_opportunistic]$ squeue --me
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
5985766	dcgp_usr_	GP_CMS_G	tboccali	PD	0:00	1	(Priority)
5985763	dcgp_usr_	GP_CMS_G	tboccali	PD	0:00	1	(Priority)
5985448	dcgp_usr_	GP_CMS_G	tboccali	R	9:41	1	lrdn4394
5985450	dcgp_usr_	GP_CMS_G	tboccali	R	9:41	1	lrdn4432
5985429	dcgp_usr_	GP_CMS_G	tboccali	R	10:39	1	lrdn4313
5985430	dcgp_usr_	GP_CMS_G	tboccali	R	10:39	1	lrdn4357
5985417	dcgp_usr_	GP_CMS_G	tboccali	R	11:41	1	lrdn3997
5985418	dcgp_usr_	GP_CMS_G	tboccali	R	11:41	1	lrdn4158
5985389	dcgp_usr_	GP_CMS_G	tboccali	R	12:42	1	lrdn4947
5985390	dcgp_usr_	GP_CMS_G	tboccali	R	12:42	1	lrdn3883
5985378	dcgp_usr_	GP_CMS_G	tboccali	R	13:43	1	lrdn3745
5985379	dcgp_usr_	GP_CMS_G	tboccali	R	13:43	1	lrdn3845
5985352	dcgp_usr_	GP_CMS_G	tboccali	R	14:46	1	lrdn4687
5985353	dcgp_usr_	GP_CMS_G	tboccali	R	14:46	1	lrdn4847
5985342	dcgp_usr_	GP_CMS_G	tboccali	R	15:41	1	lrdn4356
5985344	dcgp_usr_	GP_CMS_G	tboccali	R	15:41	1	lrdn4570
5985341	dcgp_usr_	GP_CMS_G	tboccali	R	16:42	1	lrdn4681
5985317	dcgp_usr_	GP_CMS_G	tboccali	R	16:49	1	lrdn3850
5985318	dcgp_usr_	GP_CMS_G	tboccali	R	16:49	1	lrdn4567
5985309	dcgp_usr_	GP_CMS_G	tboccali	R	17:40	1	lrdn4551
5985310	dcgp_usr_	GP_CMS_G	tboccali	R	17:40	1	lrdn4915
5985125	dcgp_usr_	GP_CMS_G	tboccali	R	27:08	1	lrdn4935
5985130	dcgp_usr_	GP_CMS_G	tboccali	R	27:08	1	lrdn4309
5985627	dcgp_usr_	GP_CMS_G	tboccali	R	4:27	1	lrdn4469
5985630	dcgp_usr_	GP_CMS_G	tboccali	R	4:27	1	lrdn4573
5985642	dcgp_usr_	GP_CMS_G	tboccali	R	3:39	1	lrdn4603
5985643	dcgp_usr_	GP_CMS_G	tboccali	R	3:39	1	lrdn4609
5985694	dcgp_usr_	GP_CMS_G	tboccali	R	0:40	1	lrdn4154
5985697	dcgp_usr_	GP_CMS_G	tboccali	R	0:40	1	lrdn4310
5985658	dcgp_usr_	GP_CMS_G	tboccali	R	1:40	1	lrdn4245
5985660	dcgp_usr_	GP_CMS_G	tboccali	R	1:40	1	lrdn4261
5985649	dcgp_usr_	GP_CMS_G	tboccali	R	2:41	1	lrdn4618
5985650	dcgp_usr_	GP_CMS_G	tboccali	R	2:41	1	lrdn4685

Test con 40 nodi = 4480 cores

Meccanismo di pressione che gira su un tmux sul nostro nodo edge (accessibile direttamente da CNAF)

```
=== Date: 2024-07-18 16:34:48.924652
(running, pending) = (10,0)
Starting 10 jobs.
Submitted job logs/gp_job_log_1721313288
Submitted job logs/gp_job_log_1721313290
Submitted job logs/gp_job_log_1721313291
Submitted job logs/gp_job_log_1721313292
Submitted job logs/gp_job_log_1721313293
Submitted job logs/gp_job_log_1721313294
Submitted job logs/gp_job_log_1721313296
Submitted job logs/gp_job_log_1721313297
Submitted job logs/gp_job_log_1721313298
Submitted job logs/gp_job_log_1721313299
Current Max_Running 100
Current Max_Pending 10
Cycles missing: 3599
```

Primi test con Virtual Kubelet

Istanziato JupyterLab su Leonardo Booster usando un k8s al CNAF e interLink su edge node come test per le attività del deliverable MS8 di Spoke0

Sarà il playground per integrare altri workflow nel contesto ICSC (spoke>0)

- Piano di lavoro in progress.

Aperto a qualunque test anche se non presente nelle attività specifiche di ICSC

Server Options

Select your desired image:

ghcr.io/dodas-ts/htc-dask-wm:v1.0.6-ml-infn-ssh-v5

Source docker image from DODAS

biancoj/jlab-ai

Source docker image (from ai-infn platform)

fcvmfs/datacloud.infn.it/test/jlab-ssh

Source docker image from DODAS

Select your desired number of cores:

Select your desired memory size:

GPU Offloading Options

GPU Model	Total GPUs	Used GPUs	Available GPUs
pcc-A200	4	0	4
T4	1	0	1

Total GPUs available: 5
Used GPUs: 0
Unused GPUs: 5

Enable Offloading to:

The screenshot shows a terminal window with the following content:

```
$ nvidia-smi
/usr/bin/sh: 1: nvidia-smi: not found
$ bash
```

Below the terminal output is a large, stylized graphic that reads "RESEARCH FLOW".

Below the graphic is a system message: "You are running this container as user with ID 31675 and group 25200, which should map to the ID and group for your user on the Docker host. Great!"

Below the message is a table of GPU information:

NVIDIA-SMI 530.30.02		Driver Version: 530.30.02		CUDA Version: 12.1	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile Uncorr. ECC
Fab. Temp.	Perf	Parti./Usage/Cap	Memory-Usage	GPU-MIG	Compute M.
0	NVIDIA A100-SXM-64GB	On	00000000:1D:00:0 OFF	0	0
N/A	42C P0	63W / 475W	001B / 65536MB	0%	Default Disabled
1	NVIDIA A100-SXM-64GB	On	00000000:5C:00:0 OFF	0	0
N/A	42C P0	61W / 475W	001B / 65536MB	0%	Default Disabled
2	NVIDIA A100-SXM-64GB	On	00000000:8F:00:0 OFF	0	0
N/A	42C P0	63W / 456W	001B / 65536MB	0%	Default Disabled
3	NVIDIA A100-SXM-64GB	On	00000000:C8:00:0 OFF	0	0
N/A	41C P0	59W / 458W	001B / 65536MB	0%	Default Disabled

Below the table is another table showing process information:

Processes:						GPU Memory Usage
GPU ID	GI ID	CI	PID	Type	Process name	GPU Memory Usage
No running processes found						

Below the table is a "Singularity" prompt.

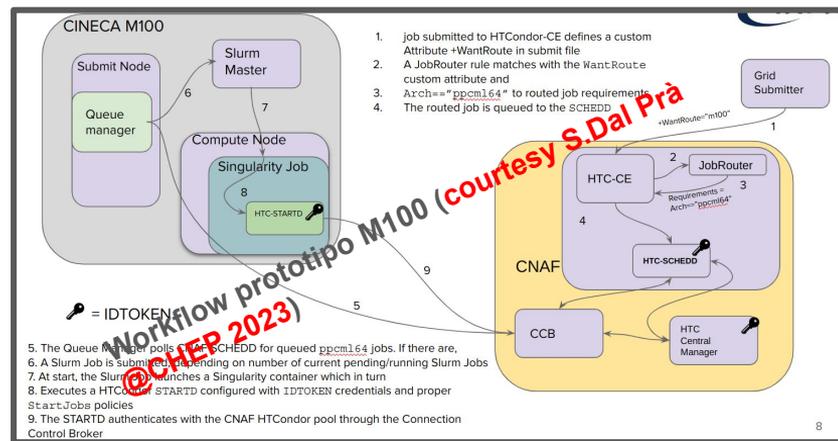
Prossimi passi: Integrazione con batch Tier1

Sul fine vita di M100 avevamo fatto una prima prototipizzazione del modello di integrazione con il batch del Tier1 sostanzialmente basato su

- Attivazione di startd (opportunamente configurati) via slurm
- Implementazione del routing a livello di CE
 - Se serve può implementare “filtro” per alcune Tipologie di job (- *want opportunistic*-)
 - Per CMS non è necessario

Mancava un meccanismo di pressione

- Ora possiamo usare quello appena mostrato
 - Eventualmente estendibile con un piccolo Check sullo stato dello schedd



Probabilmente è molto sinergico al Modello ASI di N.Mori et al