SLING in HPC Maister

Uporaba gruč: SLURM in ARC (v1.2)

Barbara Krašovec, Jan Jona Javoršek

Institut »Jožef Stefan« & SLING

Delavnica SLING, Februar 2020 FRI, Ljubljana

Vsebina predstavitve I

Vsebina predstavitve

SLING

Figure: SLING: Slovensko nacionalno superračunalniško omrežje

SLING splet, delavnice, primeri, navodila

```
http://www.sling.si/
http://doc.sling.si/ (v delu)
```

Članice SLING in oprema

- Arctur: gruči Arctur1, Arctur2 (tržna storitev)
- Arnes: gruča Jost *(EGI)*
- ARSO: 2 namenski gruči SGI
- CLARIN.si: namenska gruča NVidia DGX-1
- FMF Uni-Lj: več namenskih gruč
- FS Uni-Lj: gruča HPCFS *(PRACE)*
- IJS: gruče Atos, SiGNET, gruče R4/F8, NSC, CIPKEBIP, K3
- KI: gruča ARC, gruča Vrana
- FIŠ: gruča Rudolf, *gruča Trdina*
- UM: **gruča Maister**
- UNG: gruča Zorro
- ComTrade, Elixir, INZ, MJU (oblak), Xenya

HPC RIVR

Za člane in uporabnike SLING tudi:

- konzorcij za vzpostavitev nacionalne infrastrukture
- vzpostavitev **jedra znanja** in vrhunskih kompetenc
- vključuje vse ključne ustanove vzhodne regije
- vodstvo: Univerza v Mariboru
- infrastrukturni član: IZUM
- ne le član, **obveznost SLING**
- del nacionalne strategije in mednarodnih povezav
- **mednarodni** in bilaterlani projekti in sodelovanja, kot so EuroHPC Vega petascale, EuroHPC Leonardo ipd.

Vsebina predstavitve

Vrste gruč

- Večina gruč v SLING namenskih / optimiziranih
- Različne in podobne: soopravilnost, skupni mehanizmi, tehnična priporočila in dobre prakse (strokovni forum SLING)
- Splošne in večnamenske gruče:
  - horizontalno skaliranje (interconnect)
  - vertikalno skaliranje (many cores, big ram)
  - velepodatki
  - visoko-pretočno izvajanje in podatkovna jezera
  - fleksibilnost programske in strojne opreme
    - podpora za različice programov (standardno)
    - nadgradnje operacijskega sistema
    - nadgradnje in širitve strojne opreme
    - programska oprema uporabnikov
  - integracije v mednarodna okolja (PRACE, EuroHPC, EGI, ESFRIs, mednarodni projekti)
  - podpora za interaktivno delo, znanstvene portale, izobraževanje

Ustroj obstoječih gruč v SLING

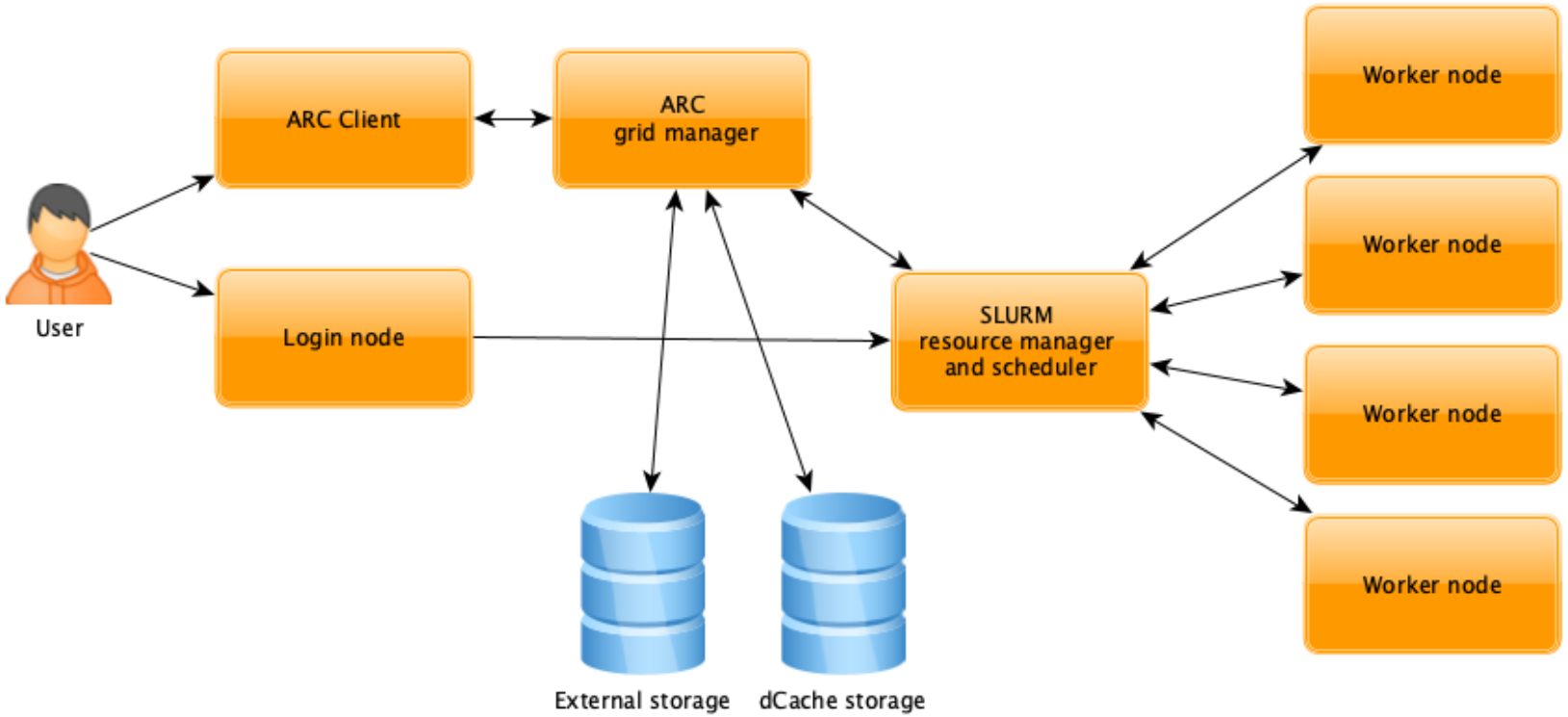


Figure: Ustroj tipične gruče SLING

Podporna infrastruktura:
podatki, navodila, programje, vsebniki, moduli, znanstveni portali

Vsebina predstavitve

SLURM - Simple Linux Utility for Resource Management

- SLURM: sistem vrst in upravitelj virov
- razvit za moderno tehnologijo okolja GNU/Linux in sodobne gruče
- omogoča upravljanje poslov na osnovi vrste parametrov: fairshare, QOS, starost, velikost in zahteve posla ipd.
- scalability: gruče 3+ milijonov jeder
- cilji arhitekture: odpornost na napake, prenosnost in skalabilnost
- razširljivost: več kot 100 vtičnikov (overjanje, MPI, popis porabe, mrežna topologija, oddaja poslov)
- projekt odprte kode

Izvorna koda:

```
git://github.com/SchedMD/slurm.git
```

Glavne komponente sistema SLURM

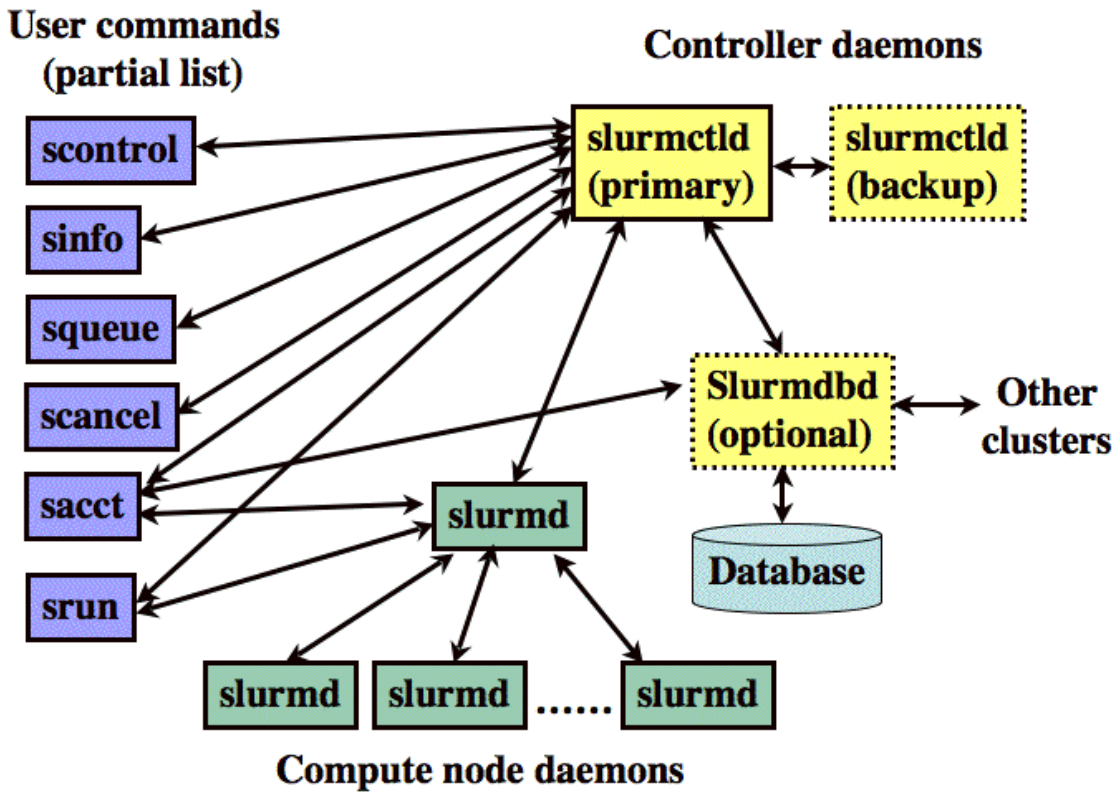


Figure: Komponente sistema SLURM

Vir: `https://slurm.schedmd.com/overview.html`

Dostop preko sistema SLURM

- Za dostop se uporablja prijavno vozlišče
- Dostop do vozlišča preko SSH omogoča uporabniško ime SLING SSO (Strežnik FreeIPA `https://fido.sling.si`)
- Dostop je na voljo uporabnikom v univerzitetnih, izobraževalnih, raziskovalnih in industrijsko-razvojnih institucijah ter uporabnikom Arnes
- Dostopi so različni; nacionalni odprti dostop, pravila v okviru SLING, osebni, testni, projektni dostop, delavnice ...

Pošiljanje nalog preko SLURM-a

Uporaba treh ukazov:

- **sbatch** se uporablja za zagon izvršljivih datotek, ki se bodo izvedle, ko pridejo na vrsto. Izvršljiva datoteka lahko vsebuje več `srun` ukazov za zagon poslov. Pri oddaji posla preko `sbatch`, dobimo ID/številko naloge v vrsti.
- **salloc** se uporablja za dodeljevanje virov za izvajanje posla v realnem času. Praviloma se uporabi, da se ustvari lupina, ki se nato uporabi za izvedbo zagona posla s `srun`.
- **srun** se uporablja za izvedbo nalog v realnem času.
  Posel lahko vsebuje več korakov (ang.: steps), ki se izvedejo zaporedno ali vzporedno, na neodvisnih ali skupnih virih v dodeljevanju posla vozlišču. Navadno se `srun` uporablja v kombinaciji s `sbatch` ali `salloc`.

Primerjava SLURM-a z drugimi upravljavci vrst je na voljo na:
`https://slurm.schedmd.com/rosetta.pdf`

srun - Zagon paralelnih poslov s SLURM-om.

SLURM `srun` pogosto enačimo z `mpirun` za posle tipa MPI. Če ne gre za paralelne naloge tega tipa, je bolje uporabiti `sbatch`. Primer uporabe:

- Primer 1: `srun hostname`
- Primer 2: `srun -N 2 hostname`
- Primer 3: `srun -n 2 hostname`
- Primer 4: `srun -N 2 -n 2 hostname`
- Primer 5: `srun -N 2 -n 4 hostname`
  - `-N` ali `-nodes`: število vozlišč
  - `-n` ali `-ntasks`: število nalog v poslu
- Dobra praksa: uporabite `-n` ali `-N` s stikalom `-ntasks-per-node`

sbatch - Pošiljanje izvršljive datoteke preko SLURM-a

`sbatch` raba:

- Zahtevane vire in druge parametre za izvedbo posla (izbira vrste/particije, čas trajanja naloge, določitev izhodne datoteke ipd), lahko določite s parametri SBATCH, ki jih navedete v začetku vaše izvršljive datoteke,takoj za `#!/bin/bash`.

```
#!/bin/bash
#SBATCH -job-name=test
#SBATCH -output=result.txt
#SBATCH -ntasks=1
#SBATCH -time=10:00
#SBATCH -mem-per-cpu=100
srun hostname
srun sleep 60
```

Pošiljanje naloge: `sbatch test-script.sh`

salloc - Dodeljevanje virov v SLURM-u.

Raba `salloc`:

- `salloc` zahteva dodelitev virov za naš posel in zažene lupino na prvem vozlišču, ki ustreza zahtevam za naš posel
- vire določimo enako kot s `srun` ali `sbatch`
- ko posel zaključimo, se dodeljeni viri sprostijo
- Primer:

```
salloc -partition=grid -nodes=2 -ntasks-per-node=8
-time=01:00:00
module load mpi
srun ./hellompi
exit
```

Primeri za srun/sbatch/salloc

- `-ntasks-per-node` : število nalog na posameznem vozlišču
- `-c` or `-cpus-per-task` : število jeder za posamezen posel
- `-time`: čas za izvedbo posla (ang. walltime)
- `-output`: stdout v določeno datoteko
- `-job-name`: ime posla
- `-mem`: skupno predpomnilnika za izvedbo posla
- `-mem-per-cpu`: predpomnilnik/jedro
- `-cpu-bind=[quiet,verbose]`:
  priključi naloge glede na CPU

Za več možnosti: `man srun/sbatch` ali `srun/sbatch -help`

Razlika med srun in sbatch

- oba ukaza izvajamo z enakimi stikali
- `sbatch` edini pozna nize poslov z istimi vhodnimi podatki (ang. *array jobs*)
- `srun` edini pozna možnost izvedbe posla `-exclusive`, ki omogoča alokacijo celega vozlišča in s tem izvedbo več paralelnih nalog znotraj ene alokacije virov (od SLURM v20.02 vključno z dodatnimi viri `gres`, npr. GPU).

Primer posla s sbatch

```
sbatch -partition=grid -job-name=test -mem=4G -time=5-0:0
-output=test.log myscript.sh
```

je enako kot:

```
sbatch -p grid -J test -mem=4G -t 5-0:0 -o test.log
myscript.sh
```

in enako kot:

```
#!/bin/bash
#SBATCH -partition=gridlong
#SBATCH -job-name=test
#SBATCH -mem=4G
#SBATCH -time=5-0:0
#SBATCH -output=test.log
sh myscript.sh
```

Osnovni pregled vrste

Kako preveriti status vozlišč in vrste?

- `squeue -l` podrobnosti poslov v vrsti (-l = long)
- `squeue -u $USER` posli uporabnika $USER
- `squeue -p gridlong` naloge v vrsti gridlong
- `squeue -t PD` posli, ki čakajo v vrsti
- `squeue -j <jobid> -start` predviden čas začetka izvajanja posla
- `sinfo` status vrste/particije,`sinfo -l -N` podrobne informacije
- `sinfo -T` prikaz rezervacij
- `scontrol show nodes` podatki o vozliščih

Maister sinfo

Pregled podatkov o vozliščih: `sinfo`

```
grid* up 2-00:00:00 3 resv gpu[01-02,04]
grid* up 2-00:00:00 30 mix cn[01-17,20-27,34,41-45]
grid* up 2-00:00:00 4 alloc cn[46-48],gpu[03,05-06]
grid* up 2-00:00:00 45 idle cn[18-19,28-33,35-40],dpcn[01-28]
long up 14-00:00:0 4 mix cn[42-45]
long up 14-00:00:0 4 alloc cn[46-48]
```

Upravljanje SLURM poslov

- `sacct`: podatki o popisu porabe za končanega posla in posla, ki je še v teku(`sacct -j <jobid>`)
- `sstat`: statistika posla, ki se izvaja (`sstat -j <jobid> -format=AveCPU,AveRSS,AveVMSize, MaxRSS,MaxVMSize`)
- `scontrol show`: npr. `scontrol show job|partition`
- `scontrol update`: spremeni posel
- `scontrol hold`: začasno zaustavi posel
- `scontrol release`: sprosti posel
- `scancel`: prekliči posel
- `sprio`: prikaži prioriteto posla

SLURM: izbira okolja za izvedbo

- gruča Maister podpira uporabo vsebnikov Singularity - vsebniki, ki so namenjeni splošni uporabi, so na voljo pod `/ceph/sys/singularity`
- programska oprema je na voljo tudi preko modulov (delo v teku): `module avail`
- nekaj programske opreme je prevedene v `/ceph/grid/software`
- uporabnik ima več možnosti prilagajanja okolja za izvajanje, in sicer z uporabo Conde, vsebnikov Singularity, vsebnikov udocker

SLURM: Izbira ustreznih vozlišč

- `-w` or `-nodelist` : določi vozlišča za izvedbo posla
- `-x` or `-exclude` : izloči vozlišča pri izvajanju posla (`srun -exclude=cn[01,04,08]`)
- `-C, -constraint=<list>` : izberi tip vozlišča z določeno oznako (`srun -constraint=infiniband`)
- `srun -p grid ...`: izbira particije
- `srun -exclusive`: zagon posla v ekskluzivnem načinu

Pregled oznak vozlišč `sinfo -o %n,%f`

SLURM: analiza napak

- za pregled porabe virov uporabite `sstat` in `sacct`
- `-o output` datoteka, kamor boste shranili stdout
- `-v verbose` za vklop podrobnejšega besedilnega opisa
- `scontrol show job <jobid>` za pregled statusa naloge
- `squeue -u $USER`: statusi nalog

Dodatni viri:

- `https://wiki.deac.wfu.edu/user/SLURM:Troubleshooting`
- `https://slurm.schedmd.com/job_exit_code.html`

SLURM: zaporedne naloge

```
#!/bin/bash
#SBATCH --time=4:00:00
#SBATCH --job-name=steps
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --output=job_output.%j.out
# %j bo zamenjan z ID-jem naloge
# Po privzetih nastavitvah bo srun uporabil vse vire
#(8 nalog za vsak posel)
srun ./posel.1
srun ./posel.2
srun ./posel.3
```

SLURM in oblike paralelizma

- večnitni posli (ang. multithreaded jobs
- večjedrni posli (ang. multicore jobs)
- nizi poslov (ang. array jobs
- posli čez več vozlišč

SLURM: paralelne naloge

```
#!/bin/bash
#SBATCH --time=4:00:00
#SBATCH --job-name=parallel
#SBATCH --nodes=8
#SBATCH --output=job_output.%j.out
srun -N 2 -n 6 --cpu-bind=cores ./naloga.1 &
srun -N 4 -n 6 -c 2 --cpu-bind=cores ./naloga.2 &
srun -N 2 -n 6 --cpu-bind=cores ./naloga.3 &
wait
```

SLURM: večnitenje

- HT je vklopljen na vozliščih z Intel procesorji (gpu01-06), na ostalih vozliščih niti niso omogočene
- za uporabo niti uporabite `-threads-per-core=2`

SLURM: GPU

- Gruča Maister ima 6 vozlišč s 4 Nvidia Tesla v100 karticami
- Dve možnosti zagona poslov GPU:
  - `sbatch -gres=gpu:4`
  - `sbatch -gpus=4 -gpus-per-node=4`
- CUDA je nameščena na sistemu in uporablja prevajalnik /usr/bin/cuda-gcc

SLURM in X11

- določeni programi zahtevajo interaktivnost
- v SLURM-u podprto od različice 17.11 naprej
- na Maistru je možnost posredovanja X11 vključena (ang. X11 forwarding)
- obstaja tudi možnost `-x11=[batch|first|last|all]`, s čimer določimo, na katero vozlišče bo omogočeno posredovanje X11 (privzeto je first)

```
ssh -X rmaister.hpc-rivr.um.si
salloc -n1
srun -x11 ime_programa

Primer za Relion:
srun --gres=gpu:4 --pty --x11=first \
singularity exec --nv \
/ceph/sys/singularity/relion-cuda92.sif relion
```

SLURM in MPI

SLURM omogoča tri načine izvedbe poslov-MPI:

- Direktno z uporabo `srun` in API PMI2 ali PMIx
- SLURM dodeli vire, nato MPI posel zaženemo z `mpirun` znotraj sbatch skripte
- SLURM dodeli vire, `mpirun` zažene naloge prek SSH/RSH (te naloge so potem izven kontrole SLURM-a)

- `#SBATCH -ntasks=16` zagnanih bo 16 MPI ranks
- `#SBATCH -cpus-per-task=1` za vsak MPI rank eno jedro
- `#SBATCH -ntasks-per-socket=8` vsako jedro 8 nalog
- `#SBATCH -nodes=1` vse bodo zagnane na istem vozlišču

Dobra praksa

Za posle MPI uporabite direkten način, z uporabo `srun` in PMIx (npr. `srun -mpi=pmix`)

SLURM PMI/MPI

Različice OpenMPI	
Open MPI <= 1.X	Podpira PMI, ne pa PMIx
2.X <= Open MPI < 3.X	Podpora za PMIx 1.X
3.X <= Open MPI < 4.X	Podpora za PMIx 1.X in 2.X
Open MPI >= 4.X	Podpora za PMIx 3.X

Table: Kompatibilnost OpenMPI in PMIx

Kateri API je podprt v SLURM-u?

Preverite z `srun -mpi=list`

Izbira ustrezne implementacije MPI

- Na gruči Maister trenutno 2 modula za MPI:
  - `mpi/openmpi-x86_64`: za naloge prek TCP
  - `mpi/openmpi-4.0.2` (privzeti): za naloge prek IB
- Pri uporabi MPI prek infinibanda je treba izbrati vozlišča, ki imajo strojno podporo za InfiniBand:
  `-constraint=infiniband`

Primer posla z MPI preko SLURM (1)

- Najprej potrebujemo MPI-knjižnice
  - Pregled modulov: `module avail`
  - Naložimo modul MPI: `module load mpi`
- Primer MPI `hellompi` lahko prenesete iz Wikipedije: `http://en.wikipedia.org/wiki/Message_Passing_Interface#Example_program`
- In ga prevedete z:
  `mpicc wiki-mpi-example.c -o hello.mpi`

Lahko uporabite tudi primer iz naslednje prosojnice.

MPI hello world

```c
/* C Example */
#include <stdio.h>
#include <mpi.h>

int main (argc, argv)
     int argc;
     char *argv[];
{
  int rank, size;

  MPI_Init (&argc, &argv);                  /* starts MPI */
  MPI_Comm_rank (MPI_COMM_WORLD, &rank); /* get current process id  */
  MPI_Comm_size (MPI_COMM_WORLD, &size); /* get number of processes */
  printf( "Hello world from process %d of %d\n", rank, size );
  MPI_Finalize();
  return 0;
}
```

Primer posla z MPI preko SLURM (2)

Primer:

```
#!/bin/bash
#SBATCH --job-name=test-mpi
#SBATCH --output=result-mpi.txt
#SBATCH --ntasks=4
#SBATCH --time=10:00
#SBATCH --mem-per-cpu=1000

module load mpi
git clone https://github.com/LLNL/mpiBench.git
cd mpiBench
make
srun --mpi=pmix --constraint=infiniband -n2 -N2 ./mpiBench
```

SLURM MPI in Singularity

Singularity omogoča uporabo prenosljivih vsebnikov, ki lahko tečejo na namiznem ali prenosmen računalniku, a hkrati podpirajo napredne tehnologije, kot so MPI in GPGPU, ter so primerni za uporabo s sistemom SLURM.
Kombinacijo Singularity in MPI lahko uporabljamo na dva načina:

- če uporabljamo paralelizacijo v okviru enega vozlišča, lahko uporabimo MPI znotraj vsebnika
- če posel paraleliziramo na več vozlišč, moramo uporabiti MPI na gostitelju

```
#MPI znotraj vsebinika
srun singularity exec centos7.sif mpirun ./hello

#MPI na več vozliščih
mpirun singularity exec centos7.sif ./hello
```

(Prenosljivost in soopravilnost s knjižnicami MPI in GPGPU je

SLURM: nizi poslov

- podpora za pošiljanje podobnih nalog z istimi vhodnimi podatki
- uporaba možna le s `sbatch`

```
sbatch --array=1-3 -N1 slurm_skripta.sh

#npr, da bo ustvarjena naloga z ID 101
```

SLURM: nizi poslov (2)

Zagnani bodo naslednji nizi:

```
# array index 1
SLURM_<jobid>=101
SLURM_ARRAY_JOB_ID=42
SLURM_ARRAY_TASK_ID=1

# array index 2
SLURM_<jobid>=102
SLURM_ARRAY_JOB_ID=101
SLURM_ARRAY_TASK_ID=2

# array index 3
SLURM_<jobid>=103
SLURM_ARRAY_JOB_ID=102
SLURM_ARRAY_TASK_ID=3
```

SLURM: nizi poslov (3)

Zagon naloge:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --output=program-%A_%a.out
#SBATCH --error=program-%A_%a.err
#SBATCH --time=00:30:00
#SBATCH --array=101-103

srun --ntasks-per-node=24 \
  ./program input_${SLURM_ARRAY_TASK_ID}.txt
```

SLURM: odvisnost poslov (dependencies)

- zagon z uporabo `#SBATCH -dependency=<type>`
  - `after:jobID`: po tem, ko se začne naloga jobID
  - `afterany:jobID`: po tem, ko se konča jobID
  - `afterok:jobID`: po tem, ko se jobID konča uspešno
  - `afternotok:jobID`: po tem, ko se jobID konča neuspešno
  - `singleton`: po katerikoli nalogi z istim imenom

SLURM — dobre prakse

- Večji začasni prostor na zahtevo na /data1
- Večji I/O: na zahtevo /ceph/grid/data (ali ARC)
- Večji začasni prostor na zahtevo na /data1
- Več krajših nalog je bolje od ene dolge (trenutno particija long)

Vsebina predstavitve

Advanced Resource Connector (ARC)

Vmesno programsko opremo ARC razvija kolaboracija NorduGrid:

- dostopnost: open-source
- overjanje/avtorizacija, upravljanje s posli in podatki
- enotna vmesna programska oprema za različne gruče
  agnostično za sistem vrst
  (Local Resource Management Systems, npr. SLURM, PBS, Condor, LoadLeveler, LSF, SunGridEngine . . .)
- okolja za izvajanje nalog

Ustroj strežnika ARC-CE

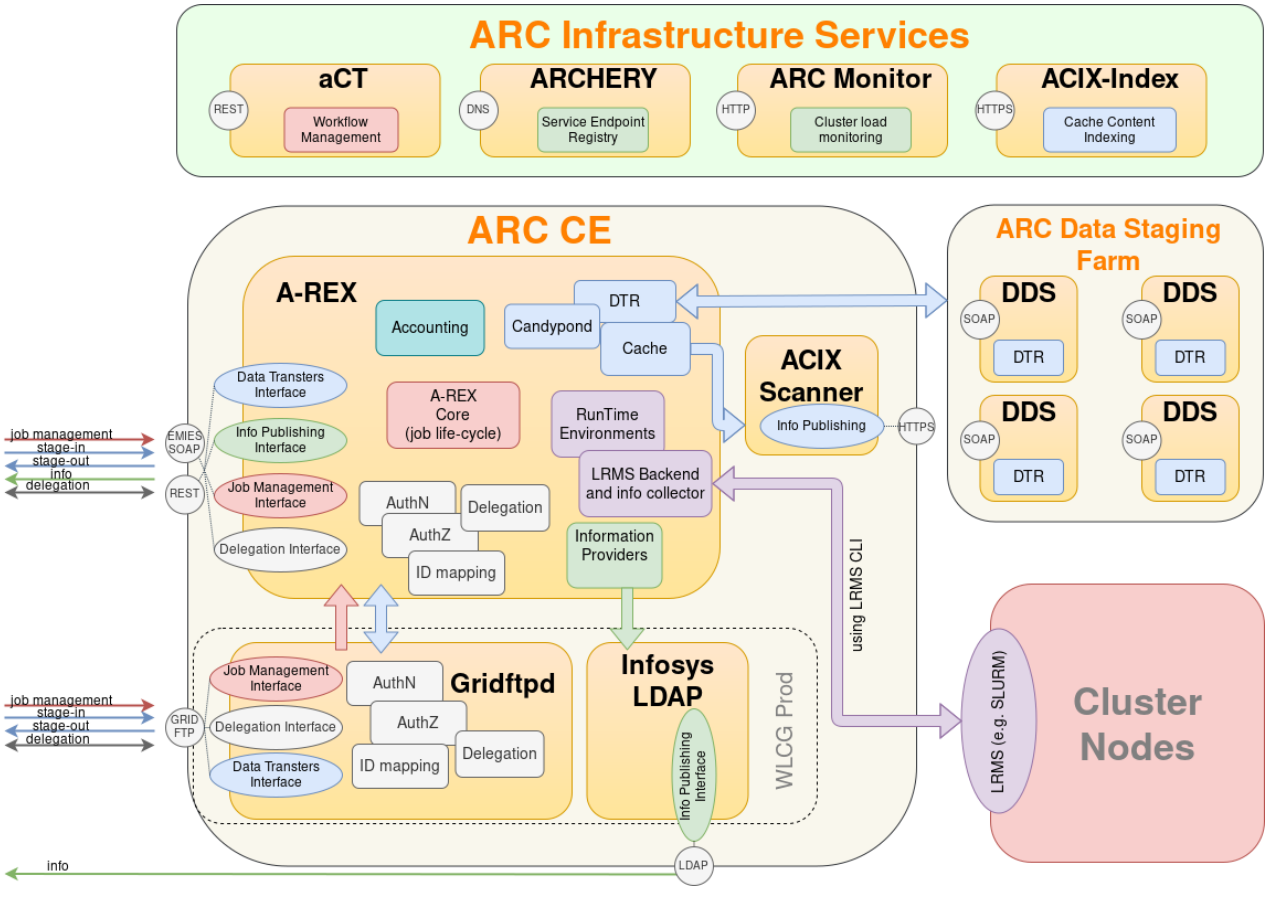


Figure: Ustroj strežnika ARC-CE

Source: `www.nordugrid.org`

ARC: overjanje in avtorizacija

- **identifikacija:** mednarodno digitalno potrdilo x509 (npr. SiGNET CA)
- posli uporabljajo kratkoživo posredno potrdilo (X.509 proxy)
- **avtorizacija:** članstvo v skupini— virtualna organizacija
- posredno potrdilo vsebuje akreditacijo virtualne organizacije (članstvo, vloga)
- Gruče SLING podpirajo nacionalo VO: **gen.vo.sling.si**.
- Ukaz: `arcproxy -S gen.vo.sling.si`

Overjanje in avtorizacija z ARC (authN & authZ)

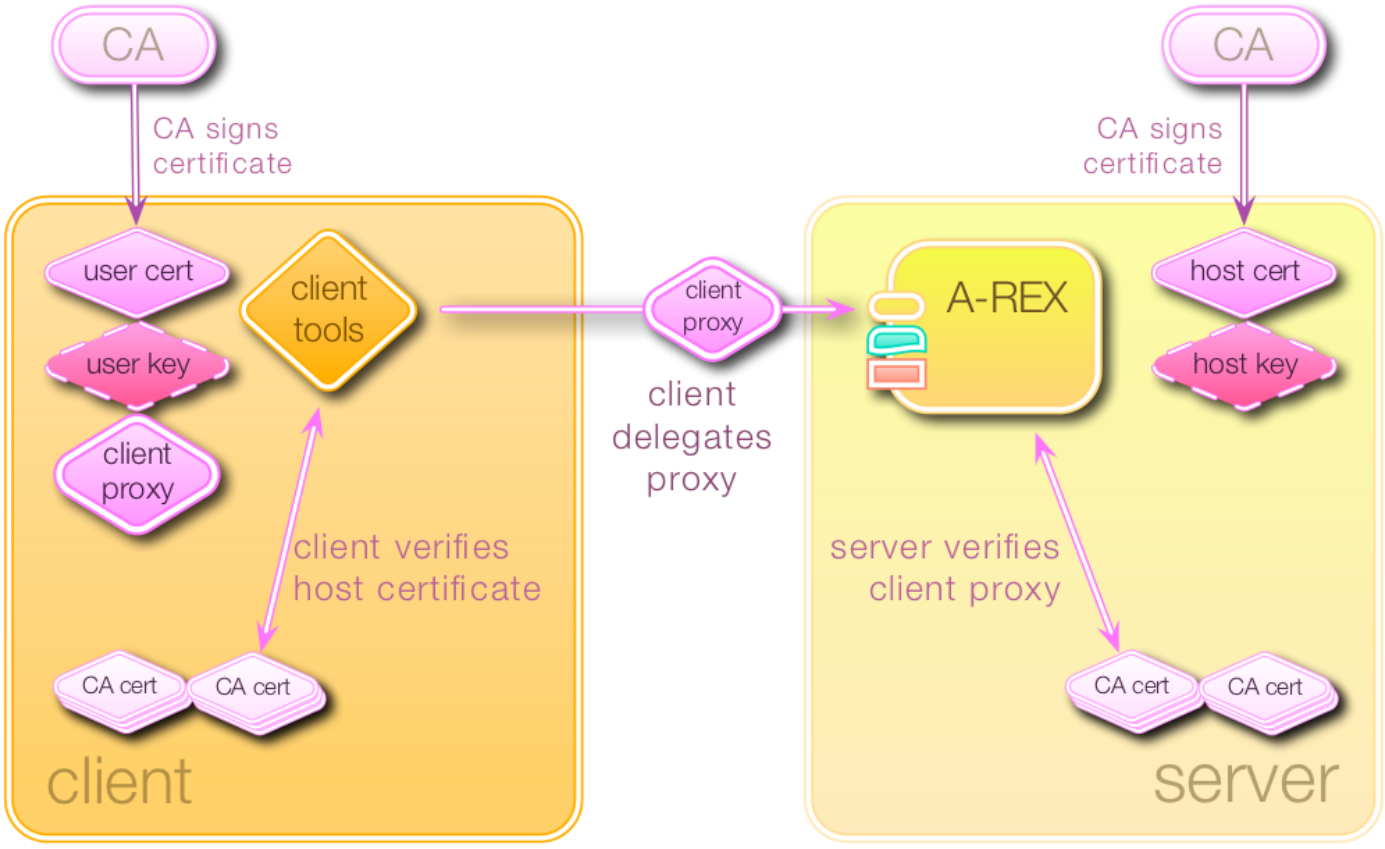


Figure: Overjanje in avtorizacija z ARC CE

Vir: `www.nordugrid.org`

ARC: življenjski krog posla

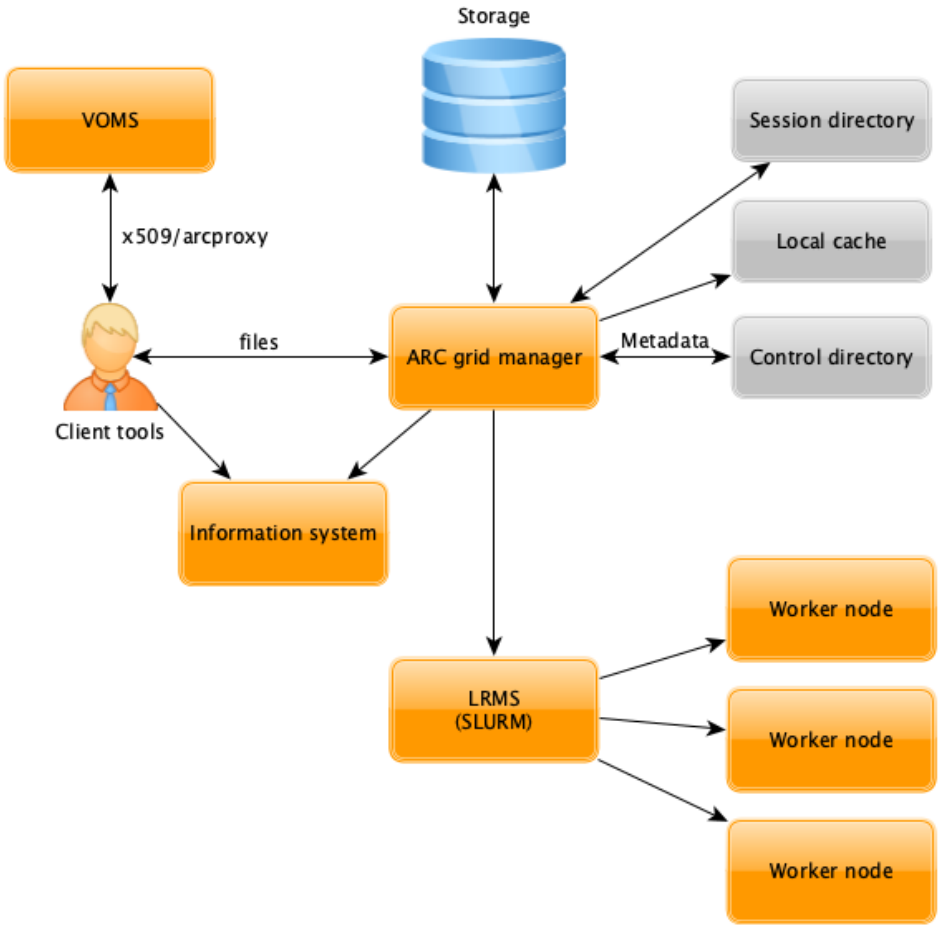


Figure: Življenski krog posla v sistemu ARC

ARC: življenjski krog posla: stanja

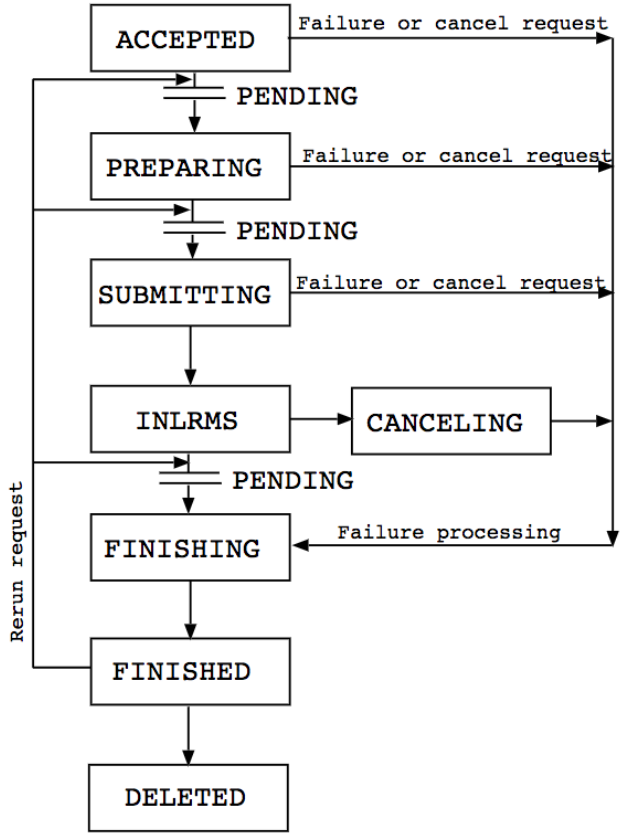


Figure: Stanja poslov v sistemu ARC

Vir: www.nordugrid.org

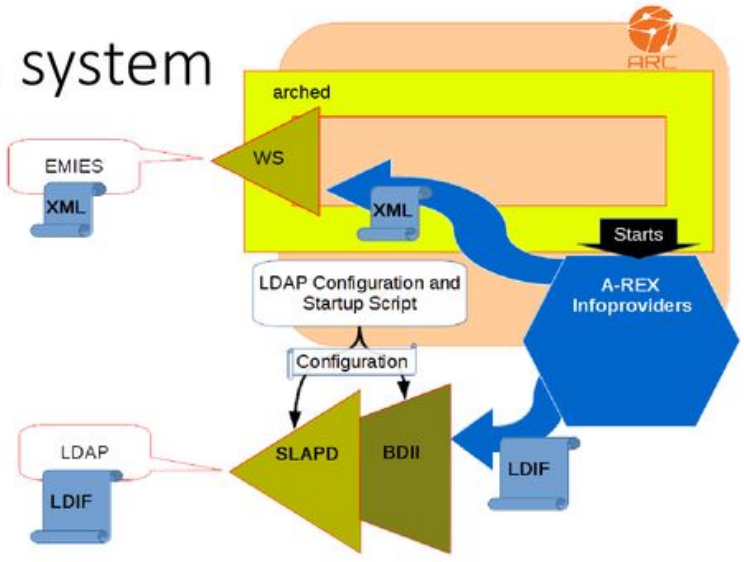
Informacijski sistem ARC - infosys (1)

- standardizirani podatki o gruči v LDAP
  - posli
  - uporabniki
  - lastnosti gruče
  - nameščena porgramska oprema
- Sistemski viri (infoproviders): moduli za različne sisteme vrst, ki zajemajo podatke
- Podatki so zajeti v začasni bazi podatkov
- Format podatkov NorduGRID
  ter standardizirana shema GLUE2
- Na voljo kot LDIF/LDAP in XML/http

Informacijski sistem ARC (2)

Overview of ARC Information system

- ARIS: ARC Resource Information Service
  - Installed on the CE or storage resource
  - Publishes via
    - LDAP
    - OASIS-WSRF (OASIS)
  - Info on: OS, architecture, memory, running and finished jobs, what users are allowed to run, trusted certificate authorities, …
  - GLUE2.0 schema (GLUE 1.2, Nordugrid schema)
    - GLUE2: An attempt to unify information and make it independent from the specific LRMS
- Infoproviders collect dynamic state info
  - Grid layer (A-REX or GridFTP server)
  - Local operating system (/proc area)



- Interfaces to
  - UNIX fork
  - PBS-family
  - Condor
  - Sun Grid Engine
  - IBM LoadLeveler
  - SLURM

Output:

LDIF format populates local LDAP tree

XML format for OASIS-WSRF

20.09.2017 Maiken Pedersen, University of Oslo 3

Figure: Informacijski sistem ARC (Vir: Maiken Pederson)

Nastavitev uporabniškega vmesnikda ARC (ARC client)

- na prijavnem vozlišču na gruči Maister je **uporabniški vmesnik ARC že nameščen in nastavljen**
- namestite paket `nordugrid-arc-client` iz repozitorijev Nordugrid - na voljo za CentOS, Fedora, Ubuntu, Debian
- pripravite digitalno potrdilo (skripta vsebuje navodila)
- nastavite nacionalni VOMS (vomses & vomsdir) za potrebne VO, npr. gen.vo.sling.si
  (navodila na spletnih straneh SLING)
- pripravite opis naloge v XRSL
  `http://www.nordugrid.org/documents/xrsl.pdf`
- pripravite posredno potrdilo:
  `arcproxy -S gen.vo.sling.si`

Pridobitev v digitalnega potrdila

- Spletna stran SiGNET CA vsebuje navodila in korensko potrdilo
- Prvič se je treba overiti (lahko pri vseh večjih članicah SLING)
- Enkrat na leto je treba zahteviti novo potrdilo
- Vlogo in prevzem se od 2019 opravi s skripto (ali ročno, npr. OpenSSL

Skripta SiGNET CA za vlogo zahtevka in prevzem potrdil:

`http://signet-ca.ijs.si/req/make-request.sh`

Primer uporabe ARC: test.xrsl

```
&
(executable = /usr/bin/env)
(jobname = "test")
(stdout=test.log)
(join=yes)
(gmlog=log)
(memory=1000)
```

- Prijava s posrednim potrdilom:
  `arcproxy -S gen.vo.sling.si`
- Oddajte posel z odjemalcem ARC:
  `arcsub -c maister.hpc-rivr.um.si test.xrsl`

Kaj se zgodi?

- prenos podatkov + gradnja skripte (SBATCH za SLURM)
- zagon posla v SBATCH z lokalnega uporabnika na gruči
- spremljanje statusa + shranjevanje podatkov

ARC: upravljanje in spremljanje poslov

- **arcsub** za oddajo poslov:
  `arcsub -c maister.hpc-rivr.um.si test.xrsl`
- **arcget** za prevzem rezultatov:
  `arcget -a` ali `arcget <jobid>`
- **arcinfo** za podatke o gruči:
  `arcinfo maister.hpc-rivr.um.si`
- **arcstat** za podatke o poslih:
  `arcget -a` ali `arcget <jobid>`
- **arccat** za ogled izpisa poslov:
  `arccat <jobid>`
- **arckill** za ukinitev posla:
  `arckill <jobid>`

Programska okolja: ARC Runtime Environments - RTE

- RTE je od implementacije neodvisen mehanizem za nastavitev okolja za izvajanje.
- Podpira skripte, environment in vsebnike.
- Ponuja standardna, lokalna in uporabniška okolja.
- Seznam nastavljenih RTE:
  `http://www.sling.si/gridmonitor/loadmon.php`
- ali z LDAP query:
  ```
  ldapsearch -x -h maister.hpc-rivr.um.si -p 2135 -b
  ' Mds-Vo-name=local,o=grid'  | grep
  nordugrid-cluster-runtimeenvironment
  ```
- Nastavite (lahko več) v opisni datoteki `xrsl`:
  `(runtimeenvironment="APPS/BASE/GPU")`

ARC in naloge MPI (1)

Priprava opisne datoteke `xrsl` z opisom zahtev
(namesto spremenjivk SBATCH v primeru za SLURM):

```
&
(count = 4)
(jobname = "test-mpi")
(inputfiles =
 ("hellompi.sh" "")
 ("hellompi.c" "")
)
(outputfiles = ("result-mpi.txt" "") )
(executable = "hellompi.sh")
(stdout = "hellompi.log")
(join = yes)
(walltime = "10 minutes")
(gmlog = log)
(memory = 100)
(runtimeenvironment = "APPS/BASE/OPENMPI-DEFAULT")
```

ARC in naloge MPI (2)

Skripta prevede program v C in ga zažene z `mpirun`:

```
#!/bin/bash
echo "Compiling example"
mpicc -o hello hellompi.c
echo "Done."
echo "Running example:"
mpirun -mca btl openib,self -np 4 $PWD/hello >
result-mpi.txt
echo "Done."
```

Vsebina predstavitve

Upravljanje s podatki

Gruče uporabljajo različne lokacije za podatke:

- Uporabniški prostor na prijavenem vozlišču (user home, omejen s kvoto)
- Podatkovni prostor na lokalnem diskovju (po dogovoru)
- Začasni prostor za vhodno/izhodne datoteke poslov ARC (lokalno diskovje)
- Vmesna shramba (cache) ARC za vhodne podatke
- Podatki dostopni preko ARC RTE
- Kratko/srednjeročna shramba na strežniku SMR/dCACHE (dcache.arnes.si)

ARC podpira še številne protokole za dostop do podatkov in shranjevanje: `ftp`, `gsiftp`, `http`, `https`, `httpg`, `dav`, `davs`, `ldap`, `srm`, `root`, `rucio`, `s3`.
ARC uporablja vmesni pomnilnik in optimizira prenose.

Strežnik dCache

- Strežnik dCache na Arnesu je na voljo uprabnikom SLING preko gen.vo.sing.si in drugih VO.
- Dostop in kapacitete se določa za posamezne skupine/VO.
- 100TB je na voljo za vhodno-izhodne podatke nalog.

dCache howto

`http://www.sling.si/sling/users/streznik-za-podatke/`

ARC in upravljanje s podatki

Podatkovne shrambe SRM (Storage Resource Manager), npr. dCache, znajo uporabiti VO skupine za dostop. Odjemalec ARC omogoča neposredno upravljanje s podatki, ki so lahko tudi vhodne ali izhodne datoteke za posle.

- **arcls** seznam vsebine oddaljenega imenika
- **arccp** kopiranje datotek
- **arcrm** brisanje datotek
- **arcmkdir** nov imenik
- **arcrename** preimenovanje datotek

Primer:

```
arcls gsiftp://dcache.arnes.si/data/arnes.si/gen.vo.sling.si/barbara/
centos7.sif
gmp\_test.c
gmp\_test.sh
gmp\_test.xrsl
...
```

Vsebina predstavitve

Prilagajanje programske opreme

Osnovni mehanizmi

- nameščanje paketov na vozlišča (std, local, opt)
- environment modules / runtime environments
- namestitev na ravni uporabnika (conda, locallibs ipd.)
- prevajanje na prijavnem vozlišču ali v okviru naloge
- uporaba lastnega vsebnika (containers)
- uporaba sistemskega vsebnika

Vidik raziskovalca in administratorja

prof. Rok Žitko, IJS & FMF Uni-Lj

Prednosti vsebnikov (containers)

- prenosnost programske opreme in vsebnikov
- izolacija programov v ločenih imenskih prostorih in omejevanje uporabe virov (linux namespaces in resource usage s cgroups)
- ne zahteva administrativnega dostopa na gruči (zahteva za ustvarjanje vsebnikov, ne namestitev zagon)
- zelo majhen vpliv na učinkovotost (performance overhead) glede na VM
- veliko možnosti prilagajanja (OS, aplikacije, orodja)
- konsistenca izvajanja poslov

Vsebniki Singularity

Navodila za Singularity

`https://sylabs.io/docs/`

- predpostavlja okolje superračunalniške gruče
- uporabnik, omrežje in podatki uporabnika so enaki
- soopravilnost z Docker (in drugimi standardi in sistemi)
- knjižnice predpripravljenih vsebnikov:
  - Docker hub: `https://hub.docker.com/`
  - Singularity hub: `https://singularity-hub.org/`
  - Nvidia cloud: `https://ngc.nvidia.com/`
  - Quay: `https://quay.io/search`

Primer prenosa obstoječega vsebnika:

```
singularity pull imgname.sif <hub>://<image>[:<tag>]
```

Delo z vsebniki Singularity

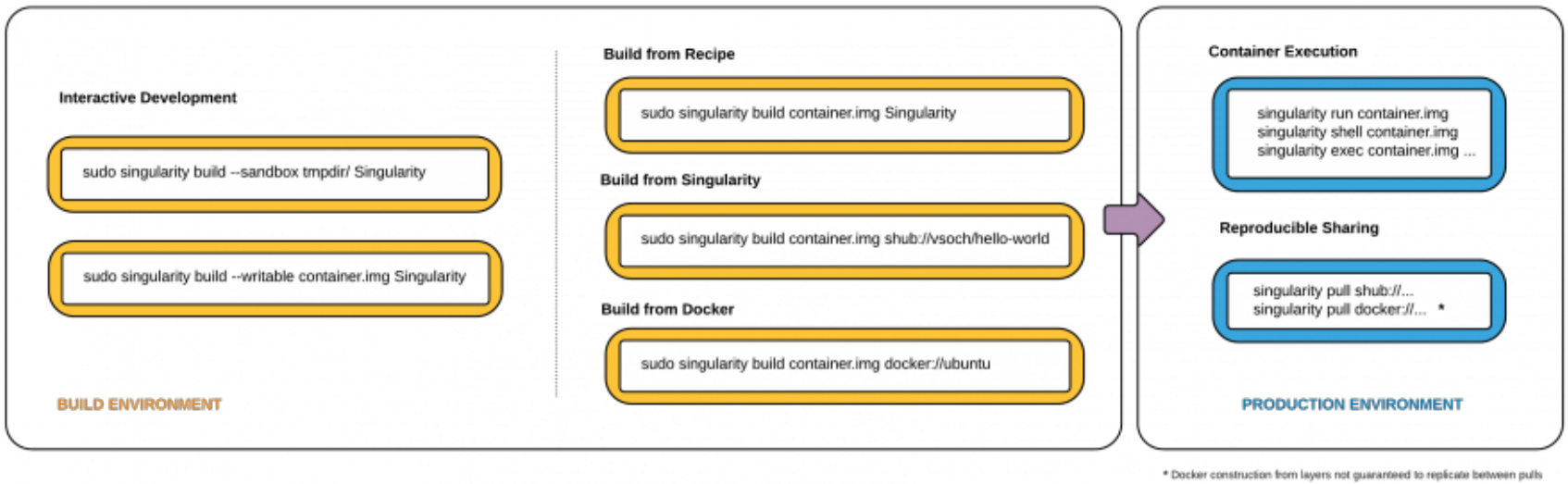


Figure: Delo z vsebniki Singularity

Vir: `https://singularity.lbl.gov/`

Singularity: ukazi

- Ukazna lupina v vsebniku: **singularity shell**

  ```
  singularity shell docker://ubuntu:eoan
  ```

- Zagon ukaza v vsebniku: **singularity exec**

  ```
  singularity exec docker://ubuntu:latest cat
  /etc/lsb-release
  ```

- Privzet ukaz v lokalnem vsebniku: **singularity run**

  ```
  singularity run centos8.sif
  ./centos8.sif
  ```

- Podatki o vsebniku:: **singularity inspect**

  ```
  singularity inspect -runscript centos.sif
  ```

- Nalaganje vsebnika: **singularity pull**

  ```
  singularity pull docker://ubuntu:latest
  singularity pull ubuntu.19.10.sif docker://ubuntu:eoan
  singularity pull tf.20.02-tf1-py3.sif
  docker://nvcr.io/nvidia/tensorflow:20.02-tf1
  ```

Singularity: nov vsebnik

- uporabite obstoječ vsebnik za osnovo in ga spremenite
- zgradite novega z opisno datoteko (root required)

```
sudo singularity build centos8.sif centos8.def
sudo singularity build centos8.sif docker://centos:8
```

Singularity: primer python

Preverimo različice, ki so na voljo na Docker hub:
https://hub.docker.com/_/python

```
sudo singularity build --sandbox python_3.7.6-stretch \
 docker://python:3.7.6-stretch

sudo singularity exec --writable python_3.7.6-stretch \
 pip3 install numpy nose test

sudo singularity build python_3.7.6-stretch.sif \
 python_3.7.6-stretch

singularity exec python_3.7.6-stretch.sif \
 python3 -c"import numpy; numpy.test()"
```

Singularity in SLURM

```
cat singularity_test.sh
#!/bin/bash
#SBATCH -J singularity_test
#SBATCH -o singularity_test.out
#SBATCH -e singularity_test.err
#SBATCH -p gridlong
#SBATCH -t 0-00:30
#SBATCH -N 1
#SBATCH -c 1
#SBATCH -mem=4000

# Ukazna vrstica Singularity
singularity exec centos7.sif cat /etc/os-release

sbatch singularity_test.sh
```

Singularity: nov vsebnik

```
BootStrap:docker
From:centos:latest

%setup

%runscript
    echo "Running the mpi container..."

%post
    echo "Installing the packages inside the container"
    yum -y install vim-enhanced autofs nfs-utils git perl perl-Data-Dumper automake \
    autoconf libtool gcc gcc-c++ glibc flex make
    echo "Installing mpirun from github"
    mkdir /tmp/git
        cd /tmp/git
        export PATH=/usr/local/share:$PATH
        git clone https://github.com/open-mpi/ompi.git
        cd ompi
        ./autogen.pl
        ./configure --prefix=/usr/local
        make
        make install
        /usr/local/bin/mpicc examples/ring_c.c -o /usr/bin/mpi_ring
        cd /
        rm -rf /tmp/git
        exit 0

%test
        /usr/local/bin/mpirun --allow-run-as-root /usr/bin/mpi_ring
```

Singularity in MPI

- MPI mora biti nameščen na vozlišču in v vsebniku!
- Dva načina uporabe z vsebniki Singularity:
  - Bind model: knjižnice MPI z vozlišča (bind mount MPI)
  - Hybrid model: knjižnice MPI v vsebniku (enaka različica kot na vozlišču!)
- V obeh primerih prenosnost in soopravilnost **ni** zagotovljena!

Primer:

```
mpirun -n <NUMBER_OF_RANKS> singularity exec --bind \
<PATH/TO/HOST/MPI/DIRECTORY>:<PATH/IN/CONTAINER> \
<PATH/TO/MY/IMAGE> </PATH/TO/BINARY/WITHIN/CONTAINER>
```

Singularity in InfiniBand

- OFED should be installed in the container as well and should match the host's OFED
- OFED (and also OpenMPI) libraries can be bound to the container: -B /usr -B /lib -B /etc -B /sys
  - potential problems with glib - it must be same version or newer than on the host

Singularity in GPU

Knjižnice je mogoče samodejno umestiti (bind) v vsebnik z opcijo `-nv`
Primer:

```
git clone https://github.com/tensorflow/models.git

singularity exec --nv \
docker://tensorflow/tensorflow:latest-gpu \
python models/tutorials/image/mnist/convolutional.py
```

Singularity: viri

- Singularity Docs: `https://www.sylabs.io/docs/`
- Singularity GitHub: `https://github.com/singularityware/singularity`
- Singularity on Google Groups: `https://groups.google.com/a/lbl.gov/forum/#!forum/singularity`
- Docker documentation: `https://docs.docker.com/`

Vsebniki udocker

udocker omogoča zagon vsebnikov Docker brez administratorskih pravic. Nekatere funkcije niso podprte, a spreminjanje datotek in zagon programov je omogočen.

Namestitev:

```
git clone https://github.com/indigo-dc/udocker
cd udocker
./udocker install
```

Primer z vsebnikom za OpenFoam:

```
udocker pull openfoam/openfoam5-paraview54
udocker run --name=openfoam openfoam/openfoam5-paraview54
```

Dokumentacija za udocker

`https://indigo-dc.gitbook.io/udocker/`

Conda

Navodila za orodje za upravljanje s paketi Conda

`https://docs.conda.io`

Minimalna namestitev conda: zažene se Anaconda,
v kateri je le conda (Miniconda).

Namestitev:

```
wget https://repo.anaconda.com/miniconda/
Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh -b -p
/home/USER/miniconda3
```

Namestitev paketa `conda` iz repozitorija Anaconda (`conda list`)

```
conda install numpy
```

Zdaj lahko dodate željene kanale:

```
conda config -addchannels biobuilds
```

Conda: primer Tensorflow

```
module load CUDA/10.1.105
wget \
https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_
bash Miniconda3-latest-Linux-x86_64.sh -b -p
$PWD/miniconda3
conda install -c defaults python=3.6 numpy six wheel
conda install -c defaults tensorflow-gpu
python -c \
"from tensorflow.python.client import device_lib;
print(device_lib.list_local_devices())"
```

Podpora in vsebniki

V SLING skušamo združiti napredne uporabnike v skupino za podporo, katere člani bi:

- skupaj vzdrževali programsko opremo svojega področja za gruče na `http://repo.sling.si/`
- pomagali vzdrževati navodila in dobre prakse za "svojo" opremo na `http://doc.sling.si/`
- izmenjevali recepte za `Easybuild`, `Singularity` in dobre prakse
- sodelovali s klicnimi centri Arnes in IZUM ter administratorji HPC centrov v strokovnem forumu SLING
- sodelovali v projektu EuroHPC Competence Centres v okviru slovenskega nacionalnega kompetenčnega centra

Koristne povezave

- HPC RIVR: `http://hpc-rivr.si/`
- SLING: `http://www.sling.si/sling/`
- SLURM: `https://slurm.schedmd.com/`
- Nordugrid ARC repos: `http://download.nordugrid.org/repos.html`
- Nordugrid ARC documentation: `http://www.nordugrid.org/arc/`
- XRSL: `http://www.nordugrid.org/documents/xrsl.pdf`
- SiGNET CA: `http://signet-ca.ijs.si/`
- Maister support: `hpc-podpora@um.si`
- SLING support: `support@sling.si`

Hvala sodelavcem, administratorjem in uporabnikom SLING za pomoč, potrpljenje in strokovno podporo.

Zahvale

Hvala sodelavcem, administratorjem in uporabnikom SLING za pomoč, potrpljenje in strokovno podporo.

- Andrej Filipčič
- Peter Kacin
- Simon Malek Kuzmič
- Avgust Jauk
- Dejan Lesjak
- Jakob Murko
- Matej Wedam
- Aleš Zemljak
- Matej Žerovnik
- Rok Žitko

Projekt so omogočili:
prof. dr. Zoran Ren, prof. dr. Miran Ulbin (UM),
Dejan Valh, Branko Zebec (IZUM),
Biljana Mileva Boshkoska (FIŠ)