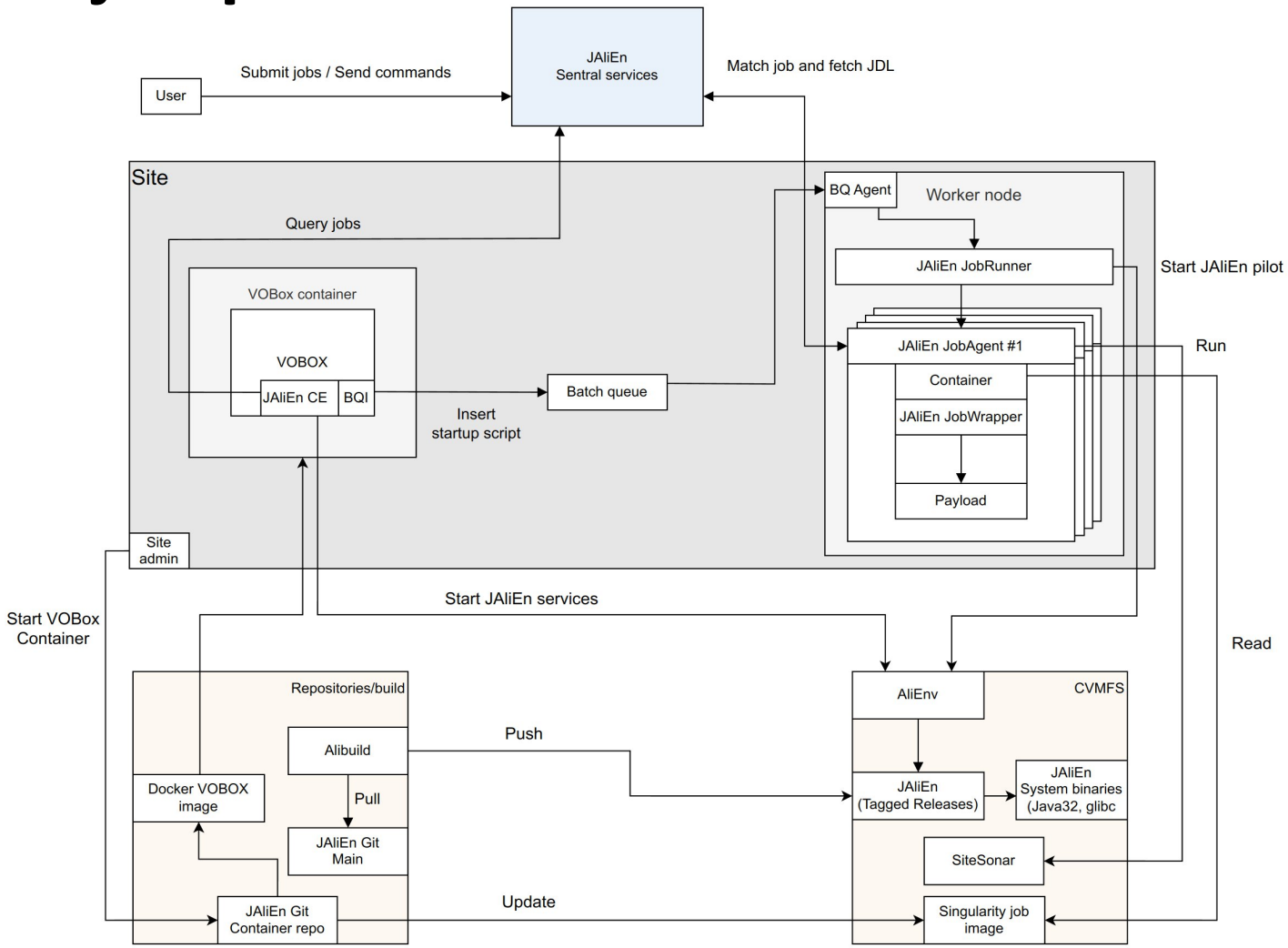
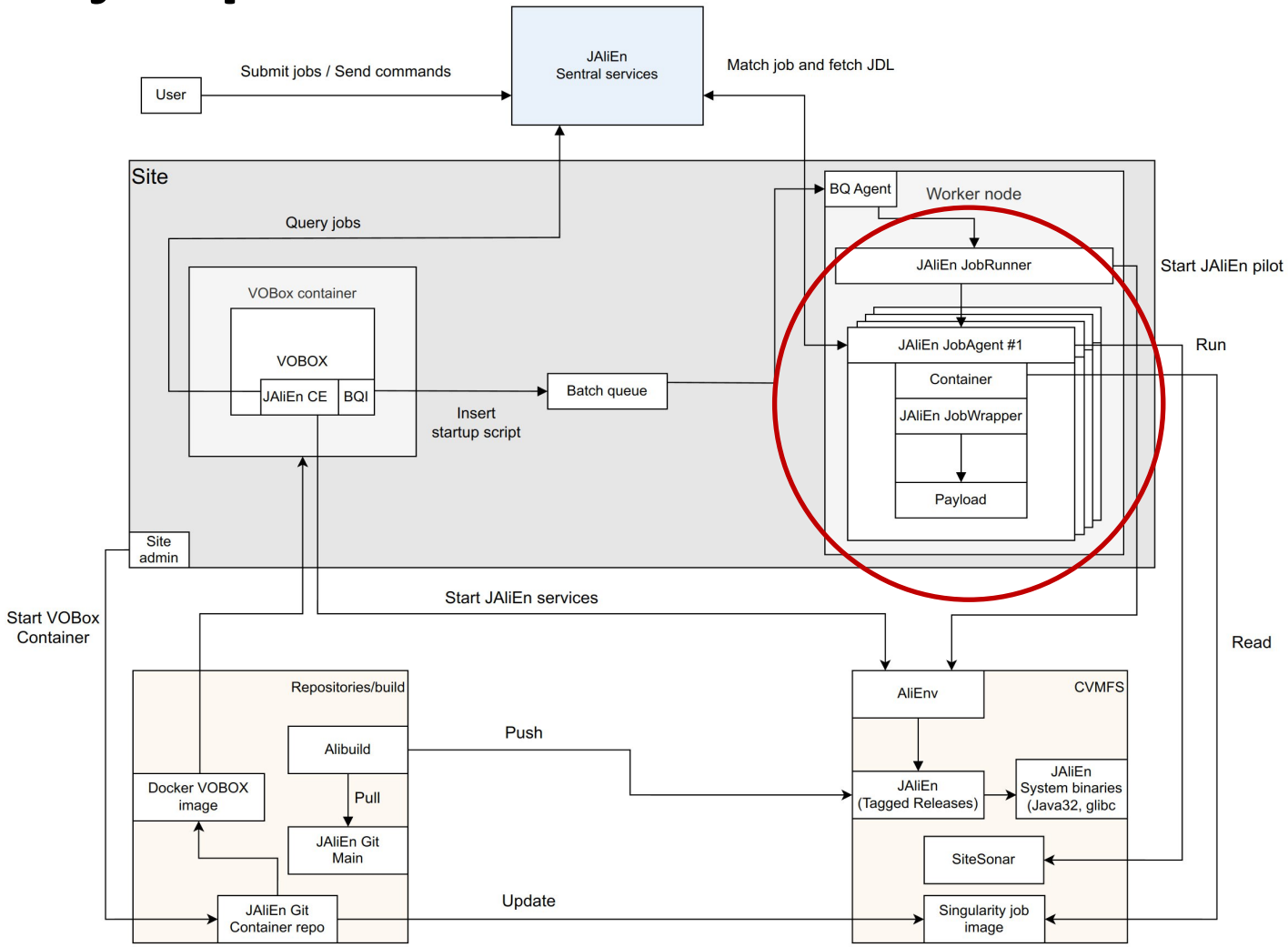
Job pilot features and job isolation

Maxim Storetvedt |ALICE Tier-1/Tier-2 Workshop 2024 | Seoul, Korea | 16/04/2024

Background: job pilots and WNs

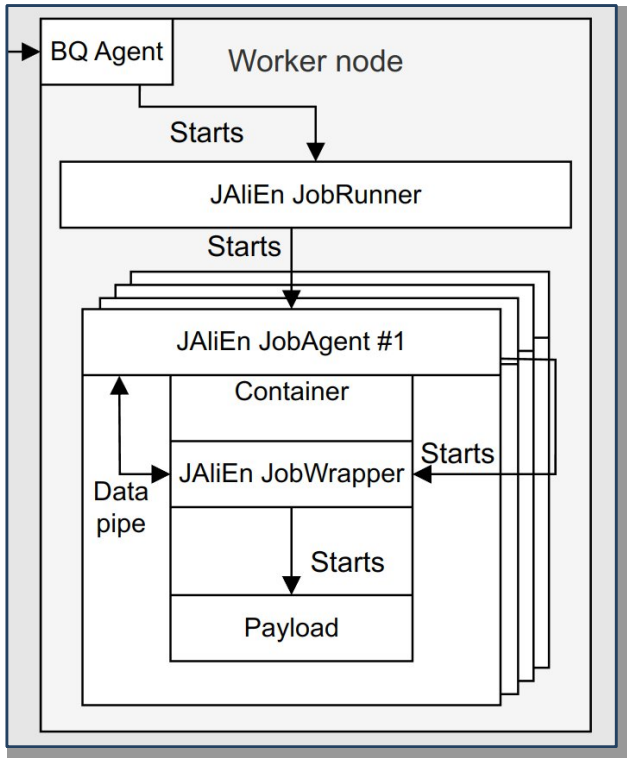


Background: job pilots and WNs



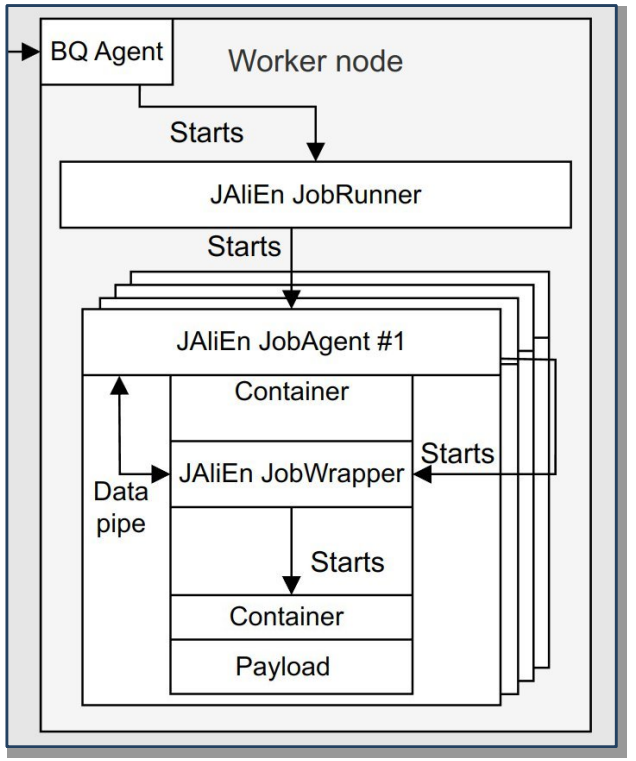
Job pilots and WNs

- Each JAliEn pilot consists of three components:
  - JAliEn **JobRunner**[1]: Resource/multicore handler
  - JAliEn **JobAgent**[2]: Job matcher/monitoring handler
  - JAliEn **JobWrapper**[2]: Payload executor
- Started by script generated by JAliEn CE, which
  - Prepares environment
  - Loads pilot using libraries and Java from CVMFS
    - System agnostic
- The JobWrapper runs on a separate JVM
  - Handles payload that can be several cores per job slot
  - Automatically **put in an isolated container** by JobAgent



Job pilots and WNs

- Each JAliEn pilot consists of three components:
  - JAliEn **JobRunner**[1]: Resource/multicore handler
  - JAliEn **JobAgent**[2]: Job matcher/monitoring handler
  - JAliEn **JobWrapper**[2]: Payload executor
- Started by script generated by JAliEn CE, which
  - Prepares environment
  - Loads pilot using libraries and Java from CVMFS
    - System agnostic
- The JobWrapper runs on a separate JVM
  - Handles payload that can be several cores per job slot
  - Automatically **put in an isolated container** by JobAgent
  - When possible, **also wraps the payload in a container** for process control



Job containers & environment

- **All** Grid jobs **must** now be wrapped by a top-level container
  - Provides a **tried-and-tested environment** across sites/nodes
  - Additional **isolation** from WN host
- Images located located in CVMFS
  - `/cvmfs/alice.cern.ch/containers/fs/singularity`
  - Build recipe available on Gitlab
    - PRs possible for package requests
- Specific image **selected by JAliEn** based on required packages for job
  - Can be CentOS 7, Alma 8 or Alma 9
  - Override possible by site in "*container.properties*"
- When **multiple** containers used, the **same image** is propagated down
- GPUs are **supported** across the multiple layers
  - Compatibility check done by JAliEn, with flags/mounts added as needed

Notable job-pilot changes

- Re-enabled **idle timeout** for jobs
  - Jobs that do nothing for longer durations will be **killed**
    - I.e. **never** once surpassed **10% cputime within 15 min**
- All WNs without AVX support are now **blocked** from production jobs
- Failed jobs that never reach a RUNNING state are automatically **resubmitted**
  - But only up to **three times**
- New Job states
  - **ERROR_A:** Job matched, but unable to continue
  - **ERROR_VN:** Validation failed to start
- Option provided to **skip** check for **corefiles**
- Option provided to manually set WN **architecture**
- ***Alternative architectures***
- ***Cgroups v2*** *for enforcing resource constraints*

Alternative architectures

Alternative architectures

- **Aarch64 supported** as of JAliEn **1.7.9**
  - Hardcoded x86 paths removed
  - **Automatic** matching of **binaries**
    - JAliEn CE and job-pilots
  - **Automatic** matching of **containers**
    - Corresponding aarch64 versions of platforms requested by job
  - Monitoring adjusted to work across architectures
  - Updated ML script for aarch64 **voboxes**
- Changes kept as generic as possible
  - Allows us to easily slot-in support for more architectures in future (**e.g. RISC-V**)

Alternative architectures

- **Aarch64 supported** as of JAliEn **1.7.9**
  - Hardcoded x86 paths removed
  - **Automatic** matching of **binaries**
    - JAliEn CE and job-pilots
  - **Automatic** matching of **containers**
    - Corresponding aarch64 versions of platforms requested by job
  - Monitoring adjusted to work across architectures
  - Updated ML script for aarch64 **voboxes**
- Changes kept as generic as possible
  - Allows us to easily slot-in support for more architectures in future (**e.g. RISC-V**)
- But, for now, jobs with aarch64 builds must **manually** be submitted to aarch64 sites
  - To be changed with new brokering service by Kalana Wijethunga (link)
    - Allows aarch64 sites to transparently run and match jobs just as any other x86 site
  - Enabled by new Optimizer by Haakon André Reme-Ness (link)

Improving job isolation with Cgroups v2

Improving job isolation with Cgroups v2

- **Cgroups v2** allows for **fine-grained resource control** per job
  - Controllers for CPU, memory, IO...
  - In theory, can be used **unprivileged** – perfect for integrating with job pilot
  - Can be automatically handled by Apptainer, but...
- ...**unprivileged Cgroups v2** forced to run on **user.slice***
  - Apptainer insists on moving its processes there
    - **Outside** of original slot cgroup!
  - And only possible for an interactive user with a session
    - Or, if **lingering** for a user is enabled
  - Must be done in advance by site admin

Improving job isolation with Cgroups v2

- **Cgroups v2** allows for **fine-grained resource control** per job
  - Controllers for CPU, memory, IO...
  - In theory, can be used **unprivileged** – perfect for integrating with job pilot
  - Can be automatically handled by Apptainer, but...
- ...**unprivileged Cgroups v2** forced to run on **user.slice***
  - Apptainer insists on moving its processes there
    - **Outside** of original slot cgroup!
  - And only possible for an interactive user with a session
    - Or, if **lingering** for a user is enabled
  - Must be done in advance by site admin

However,
moving outside of the slot cgroup
also means **no more slot limits**

Working around unprivileged limitations

- Fully unprivileged cgroups generally use **user.slice**
  - Set up automatically upon user login
- **But**, other cgroups *can* be handed to unprivileged users
  - If prepared in advance by a privileged user
- Specifically, a user must be given ownership of
  - The **cgroup** — i.e. its top-level directory in /sys/fs/cgroup
  - The **cgroup.procs** file — to allow moving processes in/out of it
  - The **cgroup.subtree_control** file — to allow delegation of controllers to subgroups

...but should this not already done by CE/LRMS when slot cgroup is constructed?

Status of unprivileged cgroups v2 in common CEs

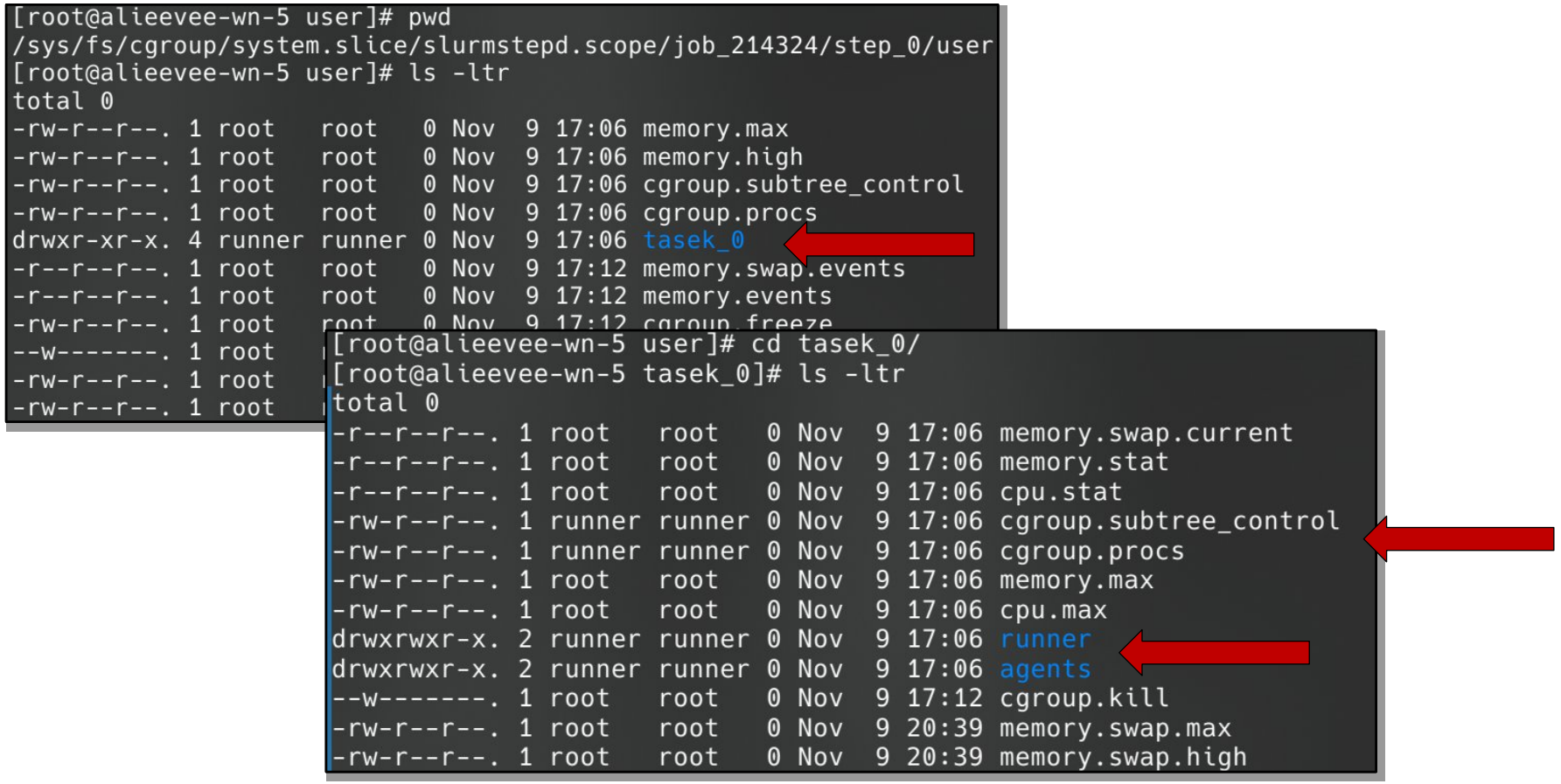
- Behaviour of both SLURM and HTCondor checked, i.e.
  - How cgroup is created for each slot
  - What ownership/permissions given
- **SLURM**
  - New cgroup dir created on each new job for slot
  - Cgroup ownership set to that of the executing user — **Great!**
  - ...but **only** cgroup ownership. All files inside still owned by root
    - Can unprivileged create new cgroups, but not move anything there
    - No unprivileged delegation of controllers
- **HTCondor**
  - New cgroup dir created on each new job for slot
  - But **all** files/directories owned by root

Workaround: a custom cgroups v2 plugin (1)

- Proof of concept Cgroups v2 plugin created for **SLURM (22.05)**
  - Existing setup already “halfway there”
  - Simple to extend functionality
  - Sets the appropriate permissions, and checks for subgroups in cleanup
- Upon building the cgroup
  - Check for given privileges
  - Attempt creating subgroups, move processes and apply limits
  - Attempt breaking limit

https://github.com/SchedMD/slurm/compare/slurm-22.05...zensanp:slurm:slurm-22.05

Workaround: a custom cgroups v2 plugin (1)



Upstreaming?

- Proof-of-concept **working** as expected
  - Tested on the *Eevee* cluster (ALICE::CERN::Eevee)
- But changes only useful if they can be **upstreamed**
  - Unlikely sites willing install external plugins/rpms
  - Unofficial changes prohibited on sites subscribed to SLURM paid support
- Request sent via the **slurm-users** mailing list...
  - No permissions to do PRs / report issues on public repository
  - No other official channels (aside from paid)
  - Archive**:** https://groups.google.com/g/slurm-users/c/v7e5DSUvPPw
- ... but **no response**

A custom cgroups v2 plugin (2)

- Proof-of-concept Cgroups v2 features added for **HTCondor (v23)**
  - Similar as for SLURM
    - Sets the appropriate permissions/ownership when building the group
- As before
  - Check for given privileges
  - Attempt creating subgroups, move processes and apply limits
  - Attempt breaking limit

https://github.com/htcondor/htcondor/compare/main...zensanp:htcondor:main

Upstreaming!

- Proof-of-concept **working** on HTCondor as for SLURM
  - Also tested on Eevee as before
- Likewise, only useful if same changes can be added **upstream**
  - Availability to increase as sites upgrade their versions
- Request sent via the **condor-users** mailing list...
  - ...and almost immediate response!
  - Interest from Condor devs
    - Experimented with this earlier, but not quite working
    - Potential use-case with GlideInWMS
    - Included in test branch within 2 weeks
      - Big thanks to **Greg Thain**!
    - ~~Still some missing features~~*

Upstreaming!

- Proof-of-concept **working** on HTCondor as for SLURM
  - Also tested on Eevee as before
- Likewise, only useful if same changes can be added **upstream**
  - Availability to increase as sites upgrade their versions
- Request sent via the **condor-users** mailing list...
  - ...and almost immediate response!
  - Interest from Condor devs
    - Experimented with this earlier, but not quite working
    - Potential use-case with GlideInWMS
    - Included in test branch within 2 weeks
      - Big thanks to **Greg Thain**!
    - ~~Still some missing features~~*

Merged to main and included in **HTCondor 23.1!**

New cgroups v2 features within the job pilot

- Previous cgroups v2 implementation used containers/containerizer
  - Built-in features of Apptainer
- Not compatible with the cgroup given by new HTCondor/SLURM
  - Apptainer always insists on moving processes
- Cgroup setup and management must now be done **manually** by **the job pilot**

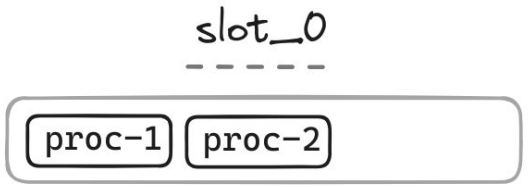
- Logic adjusted to accommodate the changes:
  - Top level slot cgroup creation and delegation of controllers
    - Via **JobRunner**
  - Job cgroup creation and limits
    - Via **JobAgent**

Constructing cgroup in JR/JA

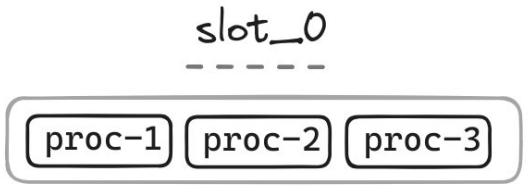
Constructing cgroup in JR/JA

Constructing cgroup in JR/JA

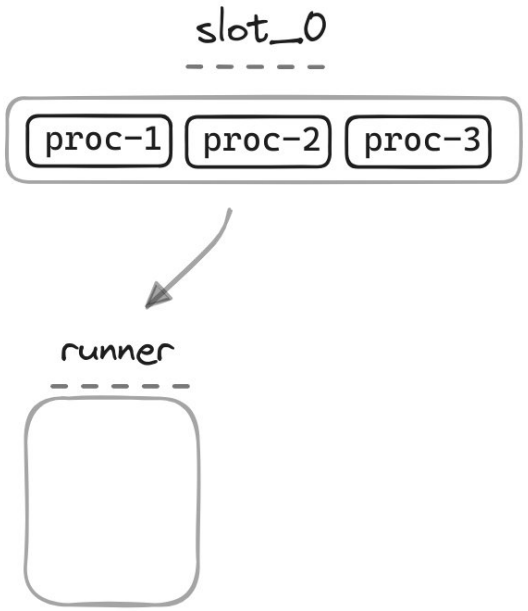
Constructing cgroup in JR/JA



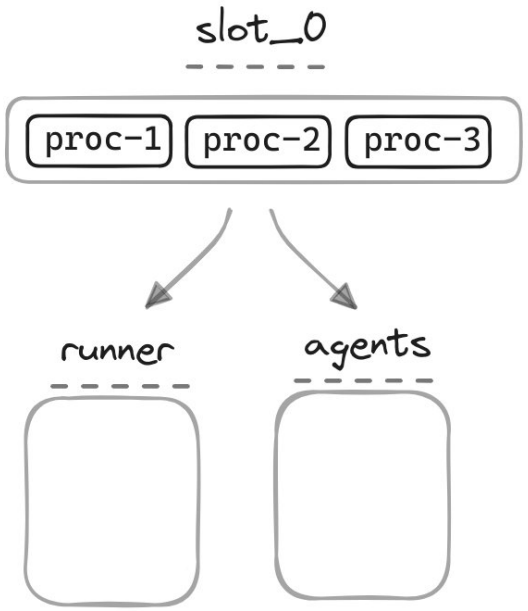
Constructing cgroup in JR/JA



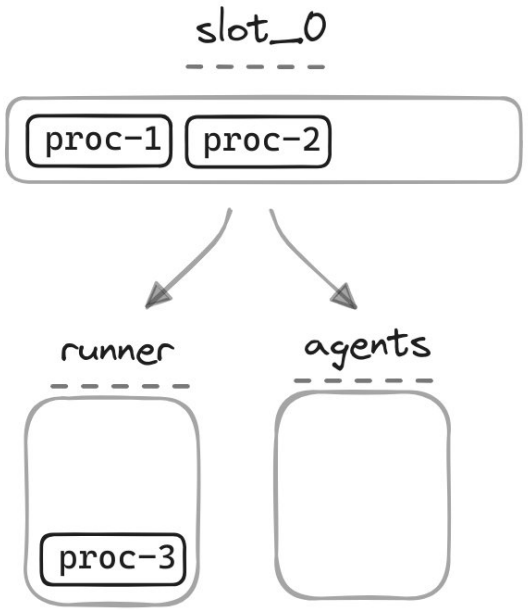
Constructing cgroup in JR/JA



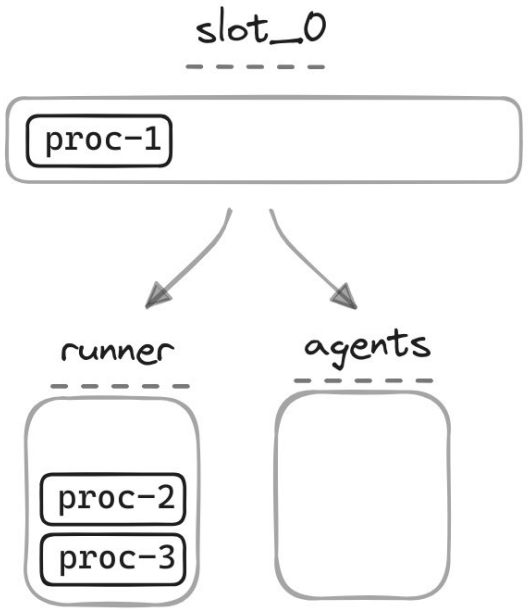
Constructing cgroup in JR/JA



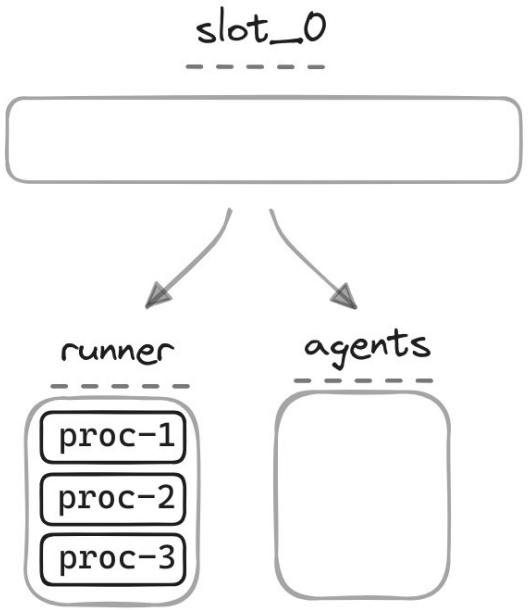
Constructing cgroup in JR/JA



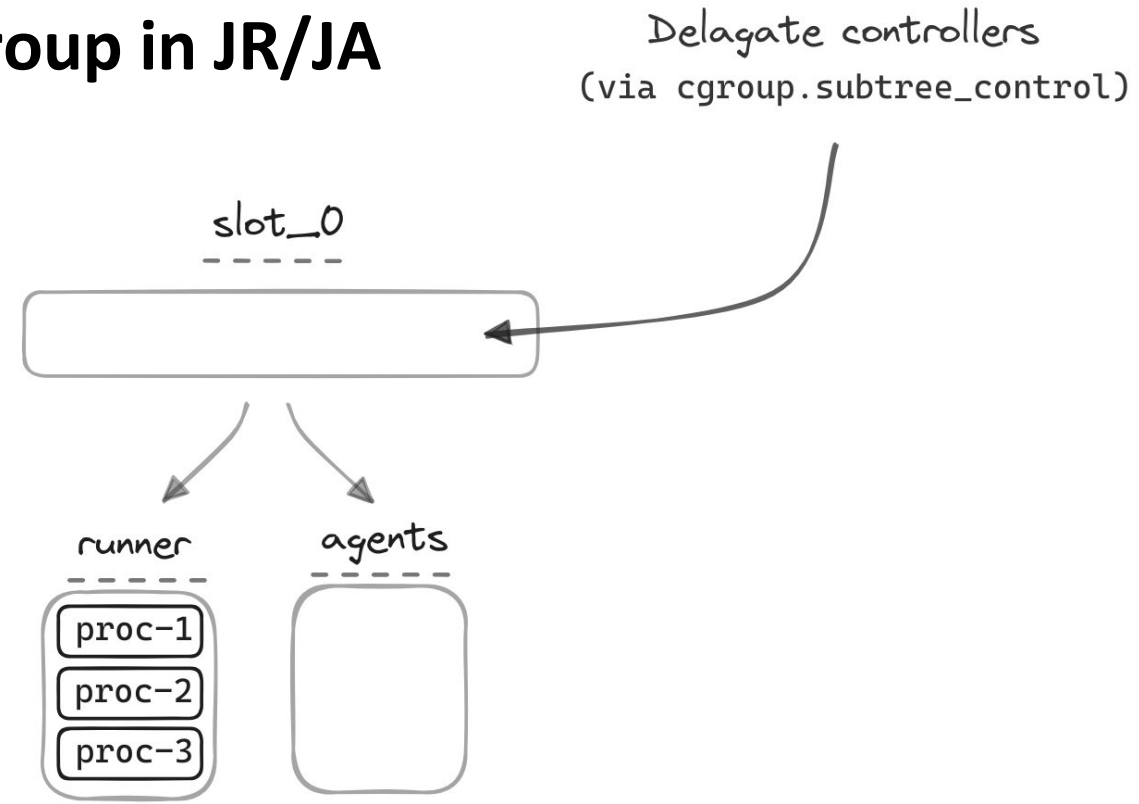
Constructing cgroup in JR/JA



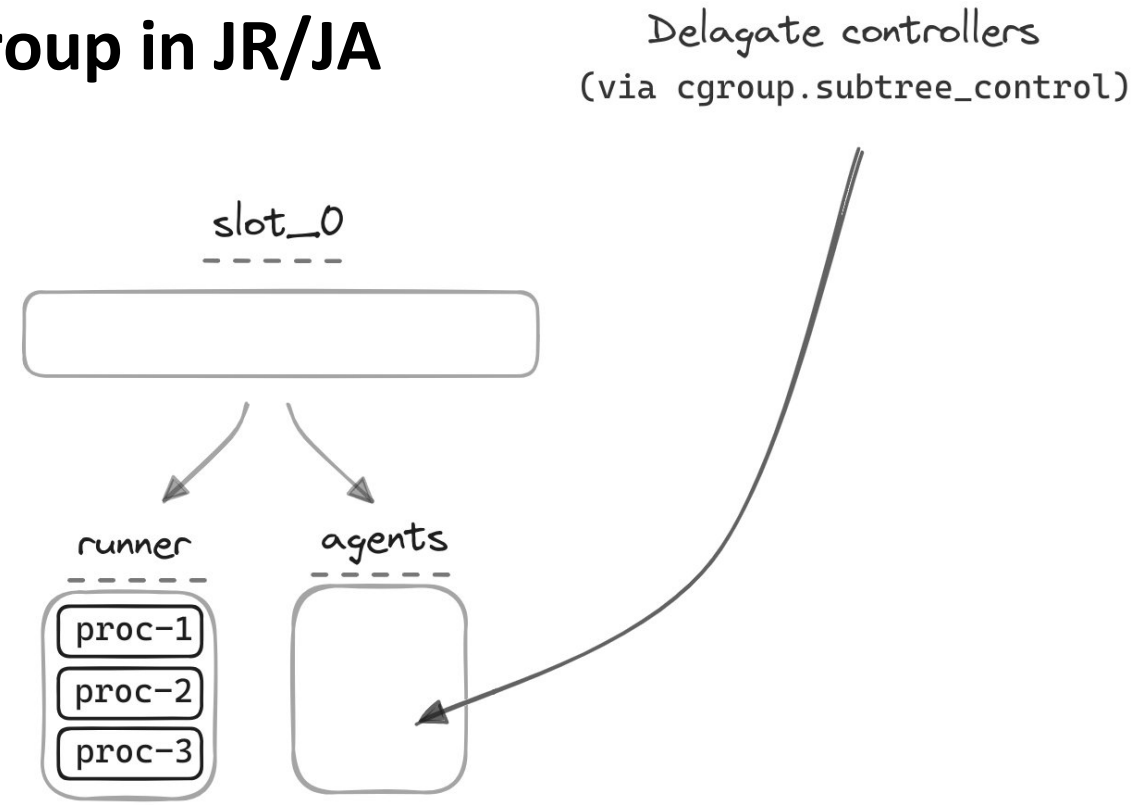
Constructing cgroup in JR/JA



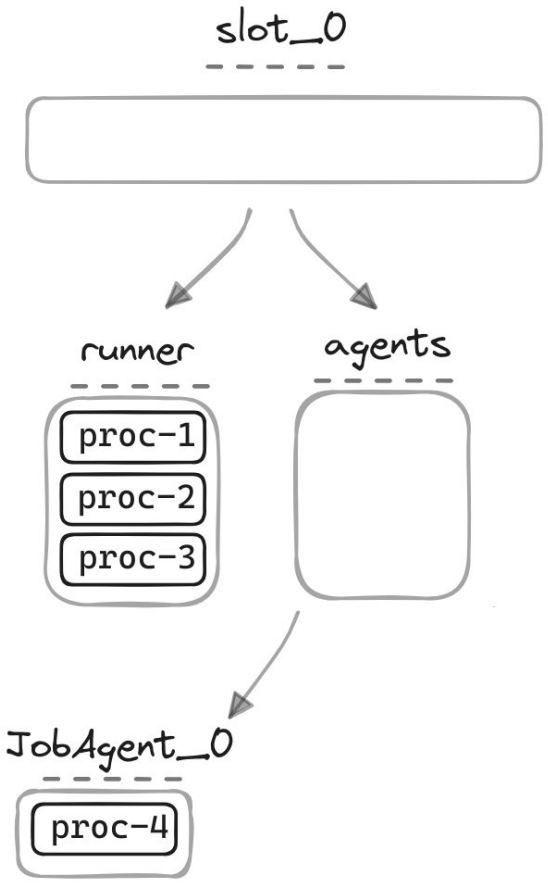
Constructing cgroup in JR/JA



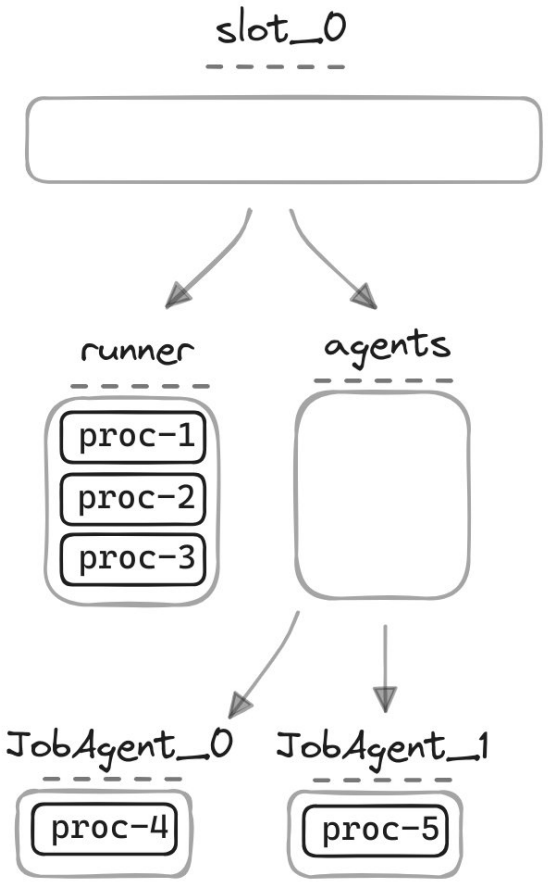
Constructing cgroup in JR/JA



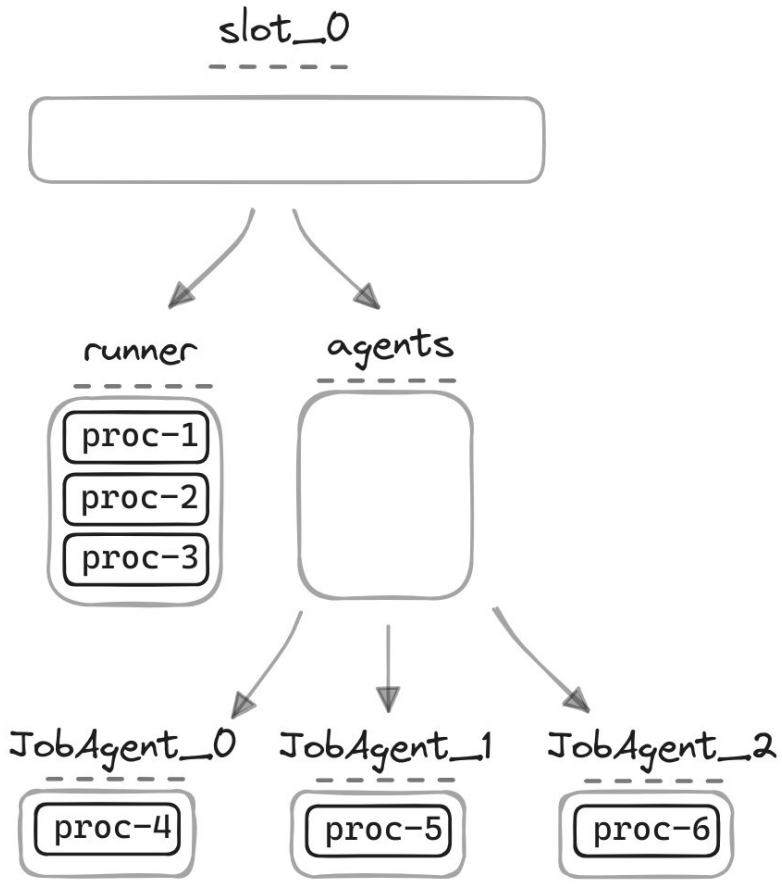
Constructing cgroup in JR/JA



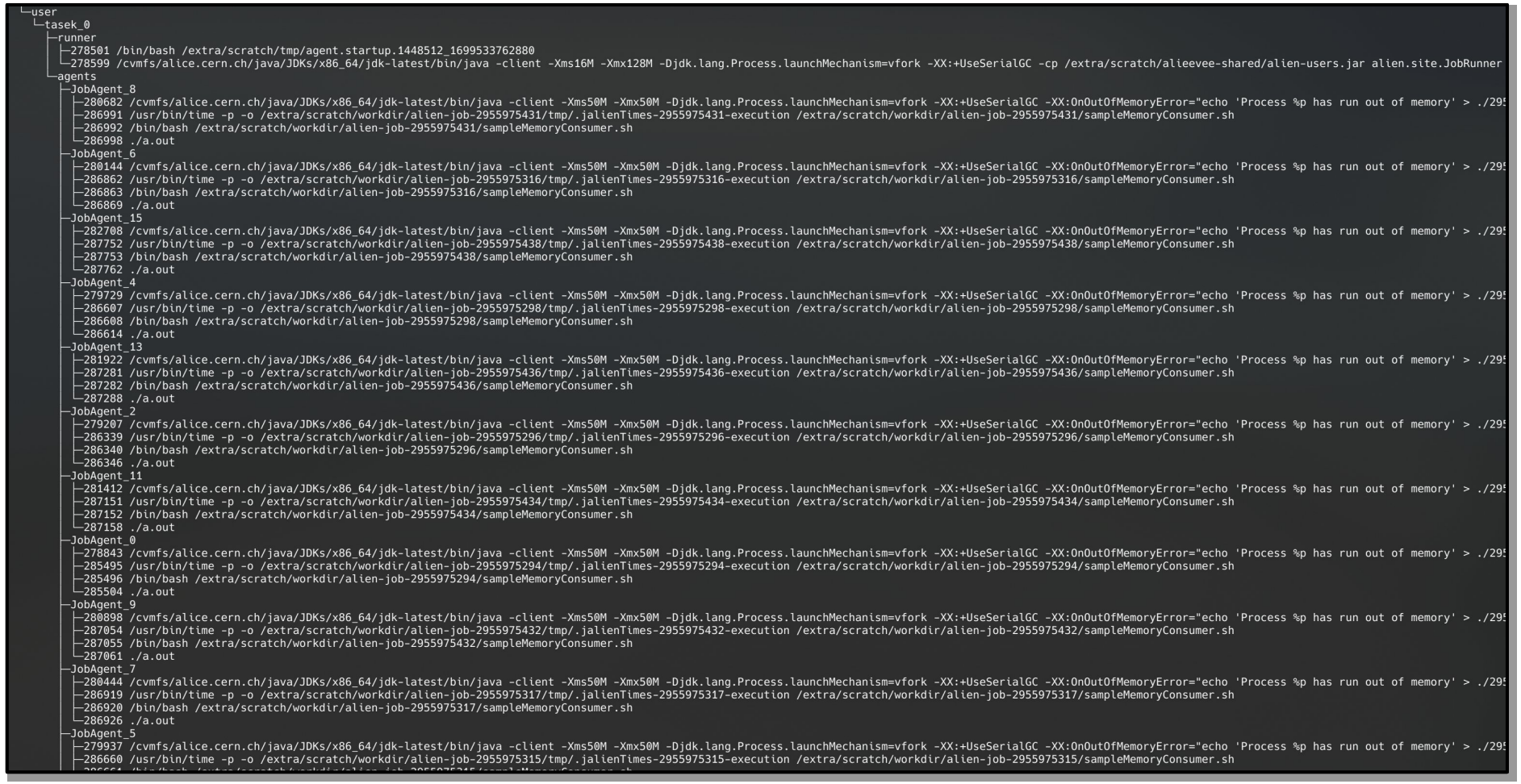
Constructing cgroup in JR/JA



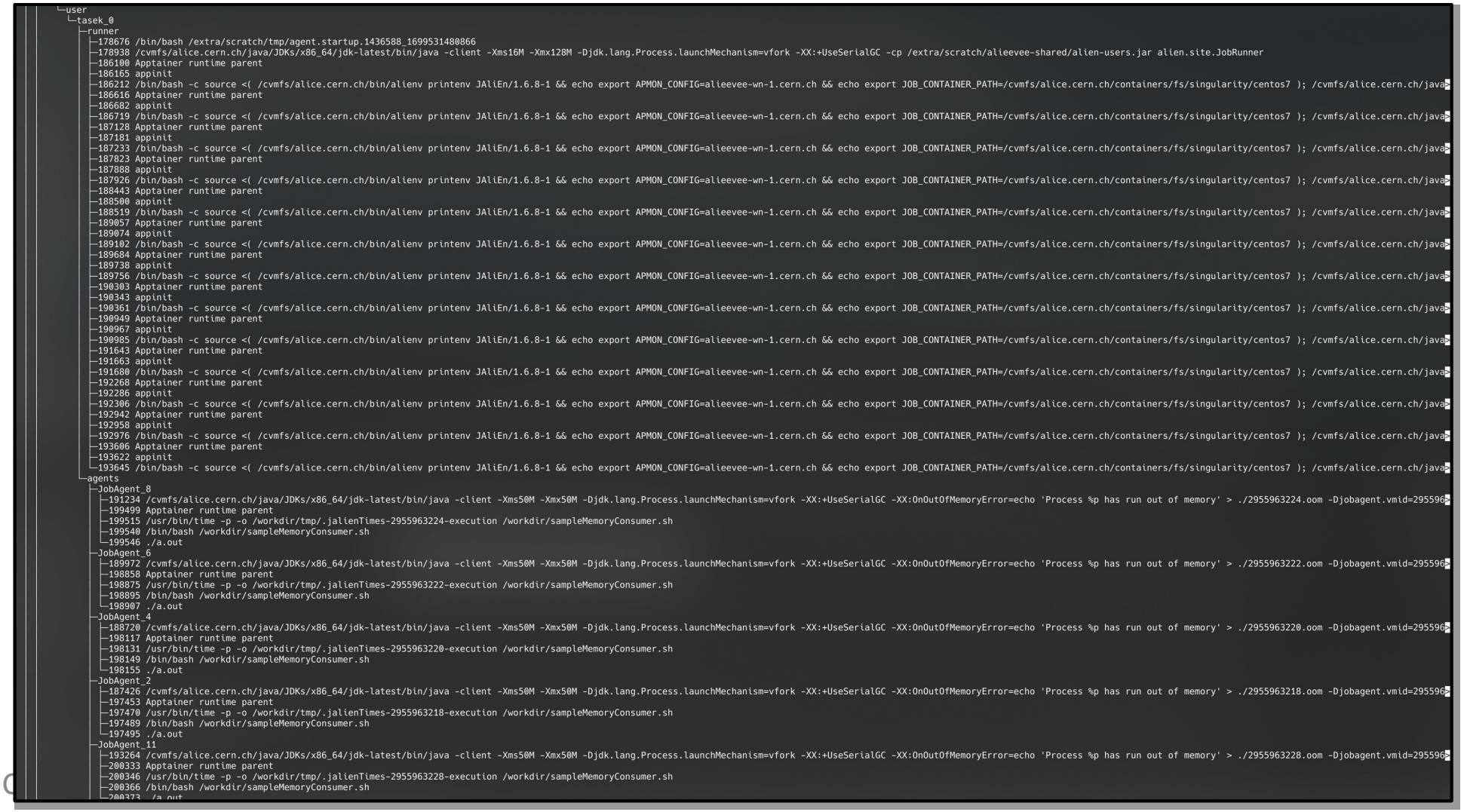
Constructing cgroup in JR/JA



JobRunner/JobAgent cgroup creation



JobRunner/JobAgent cgroup creation



Site requirements for cgroups v2

Site requirements for cgroups v2

- Must be an OS with support for Cgroups v2, e.g.
  - **EL 9** (Recommended)
  - **EL 8*** (Needs extra workaround if on Slurm)
- Must be **enabled** (default on EL9)
- **HTCondor 23.2** (i.e. not LTS)
- Or, **Slurm 22.05+**
  - For now, still with custom plugin for setup
    - Alternatively, can also be done via Slurm *prolog/epilog scripts*
  - Additional **workaround needed** for **EL 8**
    - Kernel missing "*cgroups.kill*" feature, used for cleanup
    - If cleanup fails, WN will go into drain state

Site requirements for cgroups v2

- Must be an OS with support for Cgroups v2, e.g.
  - **EL 9** (Recommended)
  - **EL 8*** (Needs extra workaround if on Slurm)
- Must be **enabled** (default on EL9)
- **HTCondor 23.2** (i.e. not LTS)
- Or, **Slurm 22.05+**
  - For now, still with custom plugin for setup
    - Alternatively, can also be done via Slurm *prolog/epilog scripts*
  - Additional **workaround needed** for **EL 8**
    - Kernel missing "*cgroups.kill*" feature, used for cleanup
    - If cleanup fails, WN will go into drain state

In other words, if already on **HTCondor 23.2** and **EL 9**, no action needed!

Summary and outlook

- JAliEn **job pilot** is becoming more **flexible**
  - Agnostic setup
  - Automatic matching of
    - Platform/containers
    - Packages
    - Architectures
      - ... including **aarch64**
- New isolation features, such as layered containers and **cgroups v2**
  - ... but needs changes to how CE/LRMS construct/clean cgroups
    - **HTCondor** – all set from **v23.2**
    - **SLURM** – TBD
      - Custom plugin or prolog needed for now
- Nevertheless, functionality **in place** for when support becomes available!

Thank You

[Questions, comments]?

email: mstoretv@cern.ch