EBRAINS

Co-funded by the European Union

ESD: Service Deployments II

Eric Müller

2024-11-29

Motivation

- ESD Clients and Tightly-coupled Services:
  - Client-server exchange via a common communication protocol
  - direct contact points: API (i.e. data serialization formats and functional API) → obvious fails
  - indirect: payload → late/undetected errors
  - Definition of an communication protocol/API level?
  - Alternative (for shared code base cases):
    Coupled builds and ensure deployments in lock-step

Goals

- ESD defines software states → allows create service software environment (stripped down to only include the service and its runtime dependencies)
  - could be a common OCI-style service container
  - or any other service packaging that allows for containerized software
    e.g., plain `systemd-nspawn` managing a (containerized) executable somewhere
  - or even flat deployments

Steps

Representation in the ESD

- Allow for keeping client and server in the same packages?
- Always split out the recipe? ($\rightarrow$ `sv-` spack packages (next to `wf-` and `bm-`))
- (For subset builds: Keep constraints as in the full set)

Synchronization of deployments

- Delayed (site) deployments:
  Can we (via the Service WG) suggest to keep oldstable running until all sites have been updated?

ESD Release $\rightarrow$ trigger service deployments

- Taken up (via poll) by CI of individual service owner's
- Actively signaled by EBRAINS CI to the service deployment flows

BrainScaleS-specific Interest

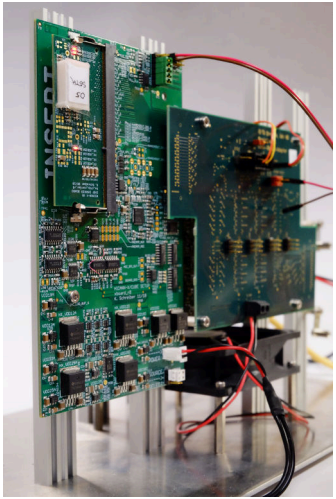
BrainScaleS Experiment Service

- BSS Experiment Client deployed via ESD (but the server executable is in the same package build)
- low-level RPC, non-portable binary serialization of C++ data structures and function calls (for “reasonable” efficiency there’s no REST, or any other web tech ;))
- (for now: only local bearer token vs. pubkey checking; but should interact with quota service to check for resource allocation)

Problem:

- . . . but it basically breaks every time the ESD deployment changes

BrainScaleS-specific Interest



BrainScaleS-specific Interest

```
mueller@helvetica:~ [3] $ systemctl status quiggeldy@ebrains_experimental.service
● quiggeldy@ebrains_experimental.service - Quiggeldy service
     Loaded: loaded (/etc/systemd/system/quiggeldy@.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2024-11-29 03:59:20 CET; 5h 13min ago
   Main PID: 3821555 (bash)
      Tasks: 17 (limit: 232026)
     Memory: 14.7M
        CPU: 15.316s
     CGroup: /system.slice/system-quiggeldy.slice/quiggeldy@ebrains_experimental.service
             ├─3821555 bash /wang/users/quiggeldy/cluster_home/service-quiggeldy/spawner.sh
             ├─3821905 socat tcp-l:11750,fork,reuseaddr tcp:EINCHost9:11750
             ├─3821906 /bin/bash /usr/local/bin/srun --jobid=6569594 -- sleep infinity
             ├─3821947 Singularity runtime parent
             ├─3821969 /bin/bash /opt/slurm-skretch/deployed/bin/clusterize srun --jobid=6569594 -- sleep infinity
             ├─3822006 srun --jobid=6569594 -- sleep infinity
             └─3822009 srun --jobid=6569594 -- sleep infinity

Nov 29 03:59:20 helvetica.kip.uni-heidelberg.de systemd[1]: Started Quiggeldy service.
```

BrainScaleS-specific Interest

Solution?

- $\rightarrow$ synchronized deployment and keeping oldstable running for some time
- (It's not only high-level software... we don't want to keep a large number of protocol versions alive)
- We have local CD in place to update service executables and to reconfigure the service... we just need to trigger and provide the correct target version.