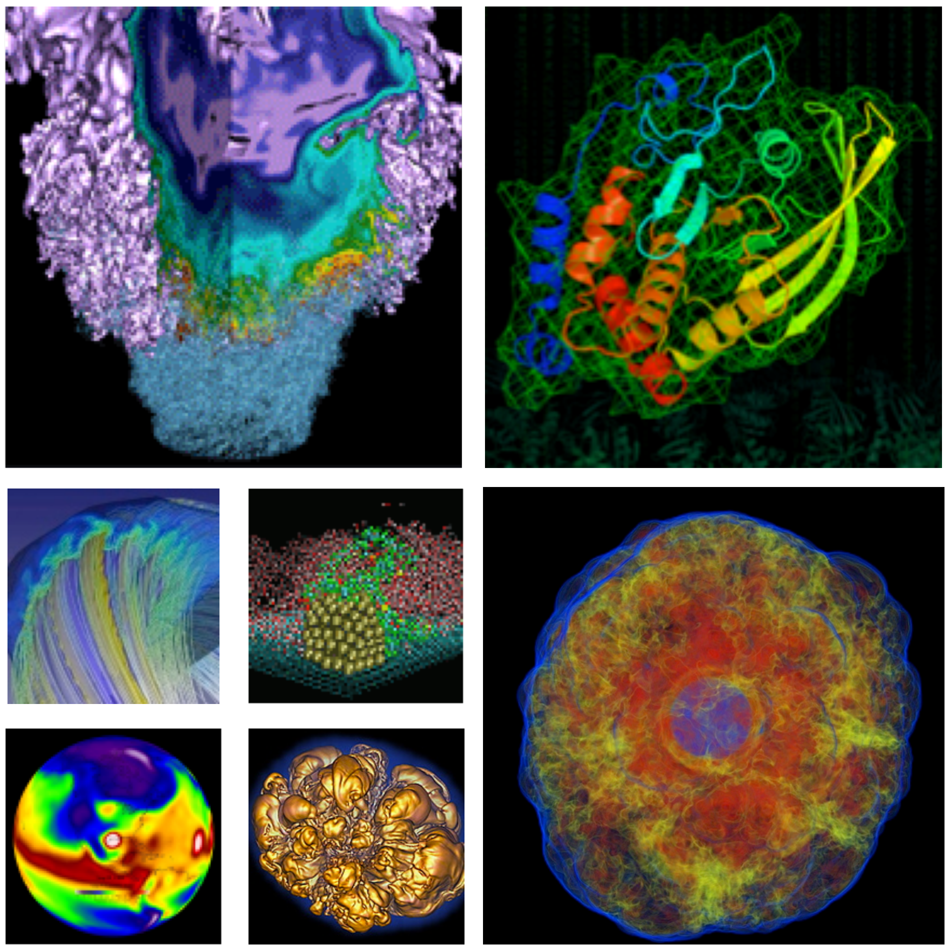
Instrumenting Slurm User Commands to Gain Workload Insight



Douglas Jacobsen[1], and Zhengji Zhao[2]

[1] Computational Systems Group
[2] User Engagement Group
NERSC at LBNL

CUG, May 24 2018, Stockholm, Sweden

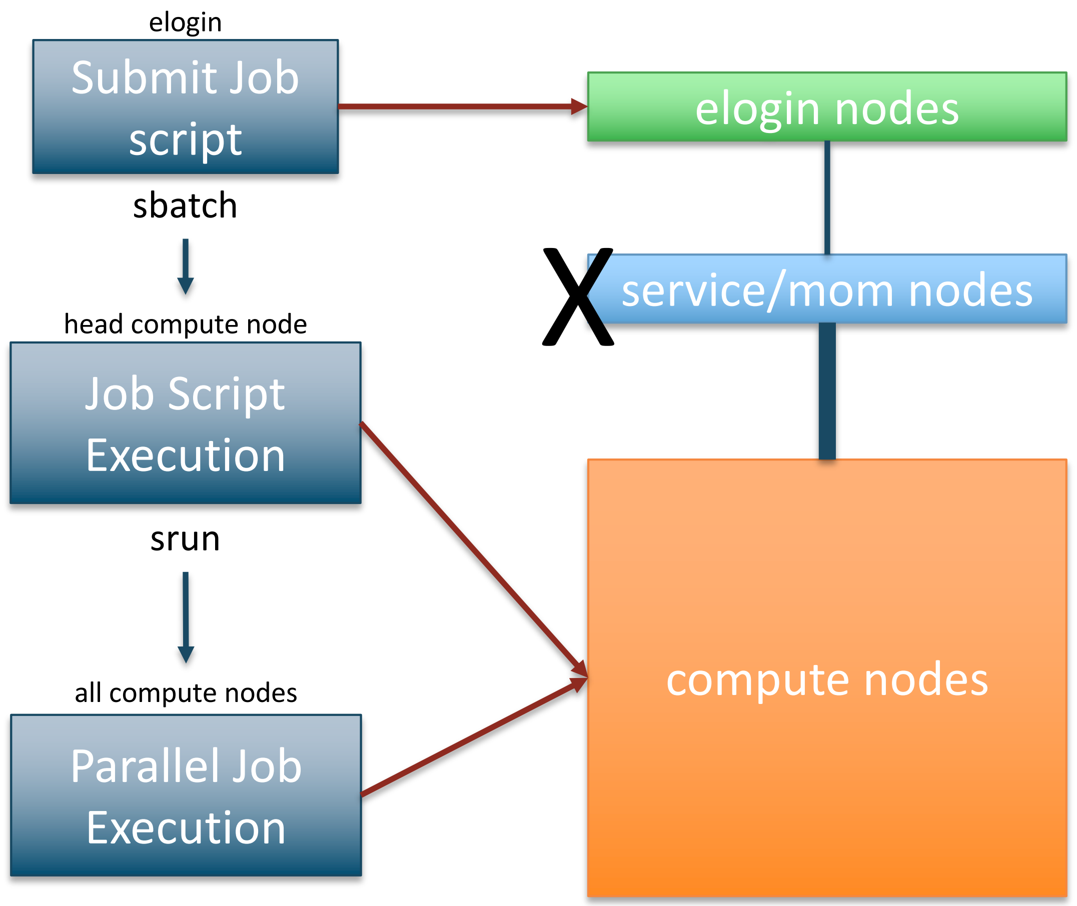
Motivation

- **Need to understand factors influencing job execution and success/failure**
- **Multiple ways users can specify job requirements**
- **Many different workloads of varying complexity**
- **Automated analysis to root cause job failures**

- **Challenges**
  - Instrumenting data sources of user activities
  - Automated collection and processing of user data from compute nodes

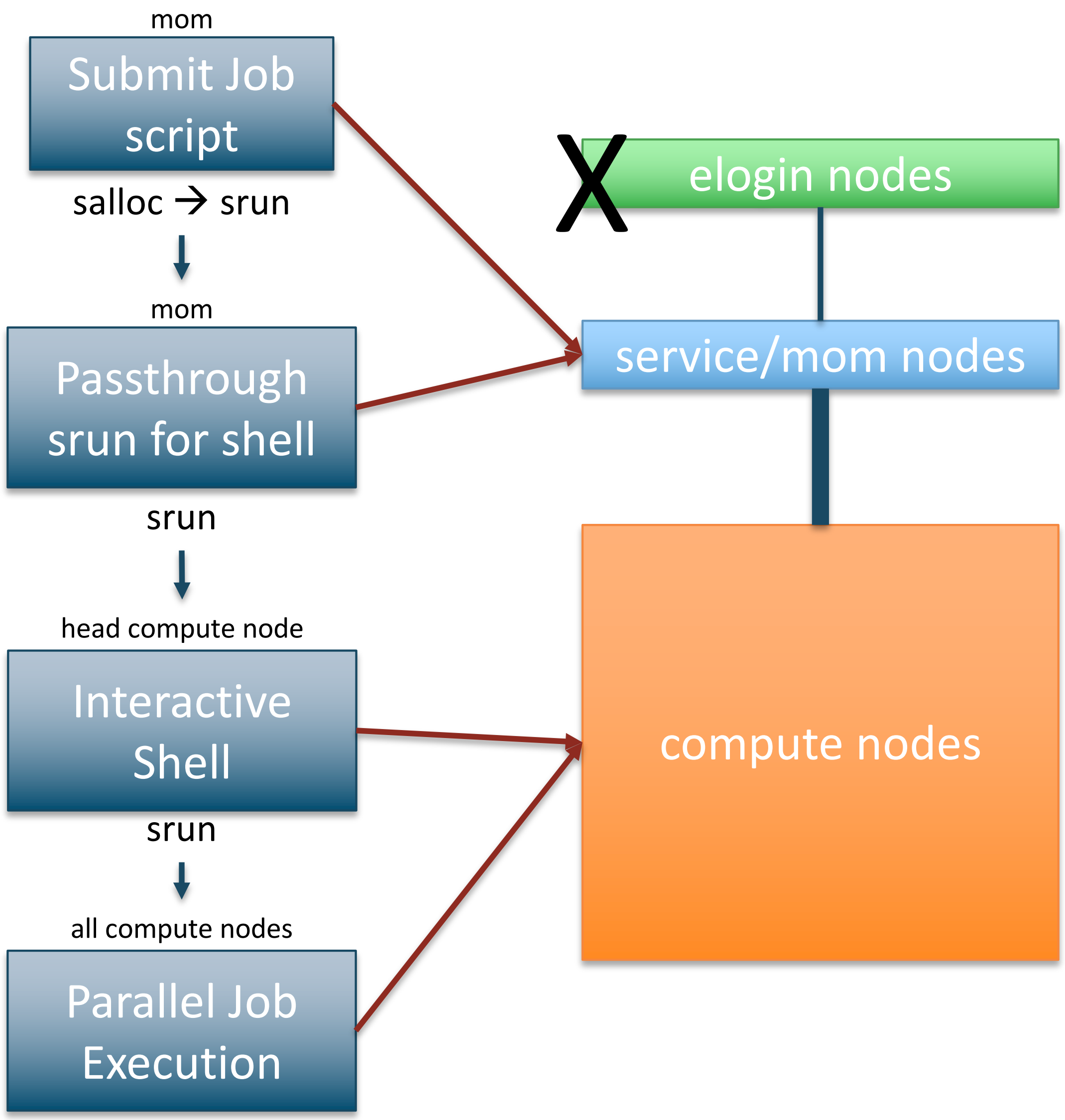
Slurm Batch Jobs

- **User submits job on external login nodes with sbatch**
- **Batch script is executed on a compute node in the job allocation**
- **srun launches the parallel application**
  - srun itself only executes on the job head node



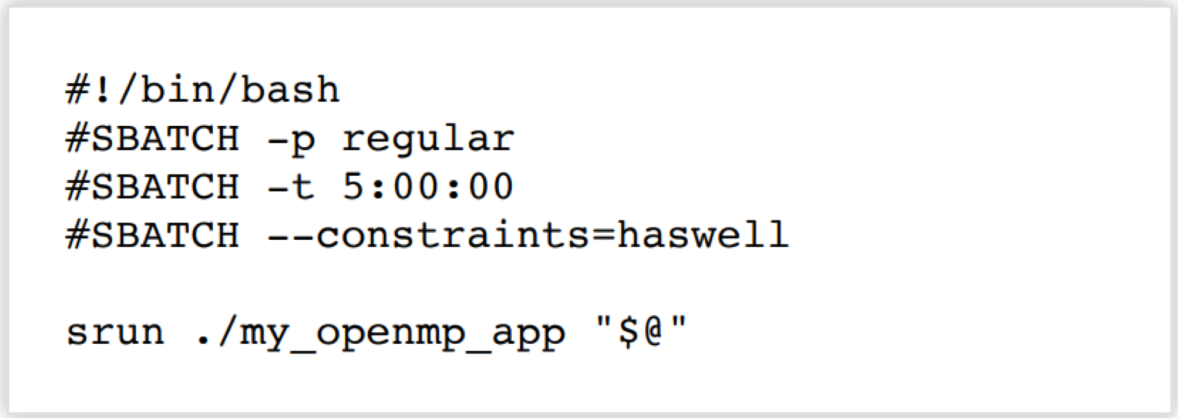
Slurm Interactive Jobs

- "salloc" on elogin proxies user to internal mom node
- salloc on mom node obtains compute allocation and launches one node srun to launch interactive shell
- srun launches the parallel application
  - srun itself only executes on the job head node



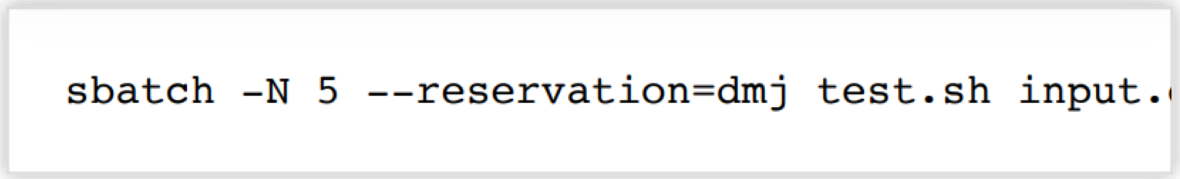
Example script

- **test.sh:**



```
#!/bin/bash
#SBATCH -p regular
#SBATCH -t 5:00:00
#SBATCH --constraints=haswell

srun ./my_openmp_app "$@"
```

- **Execution:**



```
sbatch -N 5 --reservation=dmj test.sh input.
```

Analysis of the example

- **Script makes it clear that the user requested**
  - the **regular** partition
  - a **5 hour** time limit
  - **haswell** nodes.
- **Issues**
  - The number of nodes, reservation, and script arguments are not recorded in the script.
  - It appears to be an openmp application, was $OMP_NUM_THREADS set? cpu_binding style?
  - Debugging this user's experience will rely somewhat on their memory of the job submission.

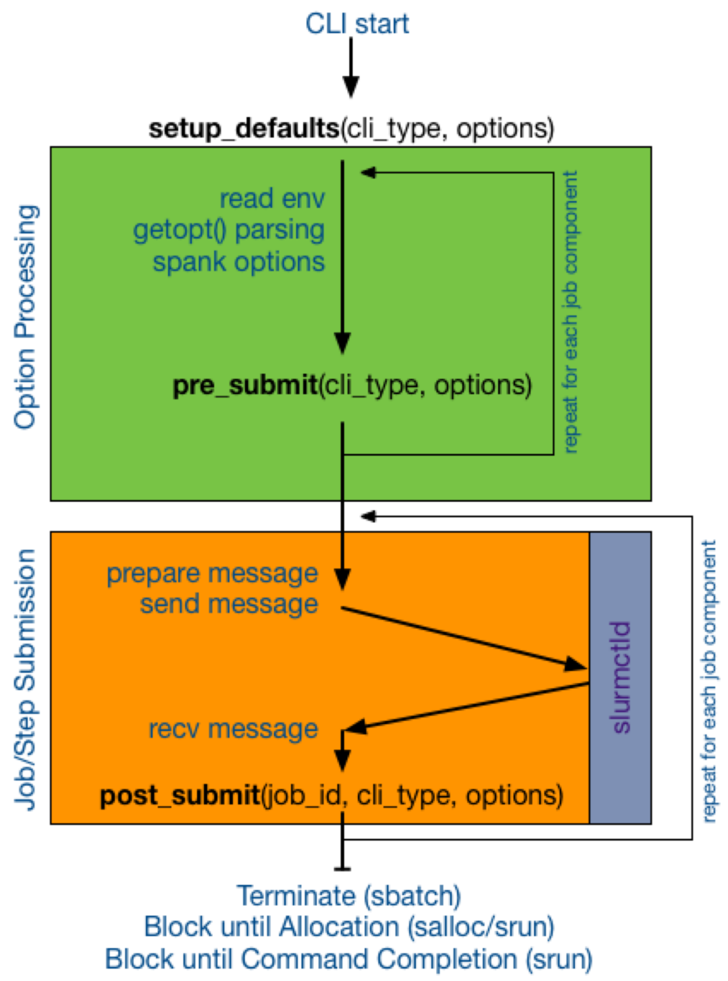
Monitoring Slurm Data

- **Needed Data beyond the Slurm Database**
  - slurmctld data structure representations of job/step data
    - jobcomp/nersc
  - capturing and logging all job and step submissions options, including aspects of the environment
    - **cli_filter (this topic)**
  - jobacctgather profiling data (site enforced)
    - need more scalable backends, hdf5 file per node per job doesn't scale well
    - 2018 NERSC priority

Instrumenting Slurm Commands

cli_filter

- new stackable plugin infrastructure
- adds hooks to allow site-definable, configurable behavior for
  - salloc
  - sbatch
  - srun
  - sbcast (limited support)


CLI start
setup_defaults(cli_type, options)
Option Processing
read env
getopt() parsing
spank options
pre_submit(cli_type, options)
repeat for each job component
Job/Step Submission
prepare message
send message
slurmctld
recv message
post_submit(job_id, cli_type, options)
repeat for each job component
Terminate (sbatch)
Block until Allocation (salloc/srun)
Block until Command Completion (srun)

non-zero exit from **setup_defaults**() or **pre_submit**() cause error exit of the CLI app.

exit status of **post_submit**() is not meaningful (the RPCs are already sent)

cli_filter setup_defaults()

- **setup_defaults**
  - Runs once per cli_filter plugin per CLI execution
  - Non-zero exit will terminate the CLI execution
  - Runs after opt data structure allocation and initialization, before environment or option processing
  - Run long-running checks exactly once
- **Implementations**
  - cli_filter/user_defaults reads ~/.slurm_defaults to set options
  - cli_filter/lua to set site default options

cli_filter/user_defaults

- **Set defaults command line options in $HOME/.slurm_defaults. Accepts (:?)(:?) = syntax.**
- **$HOME/.slurm_defaults example:**

```
partition = regular
cori:constraints = knl,quad,cache
edison:constraints = ivybridge
salloc:*:qos = premium
```

cli_filter pre_submit()

- **pre_submit**
  - Runs once per job-pack per cli_filter plugin per CLI execution
  - Non-zero exit will terminate CLI execution
  - Runs after all option processing but before slurmctld message preparation (can change options here)
- **Implementations**
  - cli_filter/lua plugin can be used to read options, implement policy, change options or terminate job submission

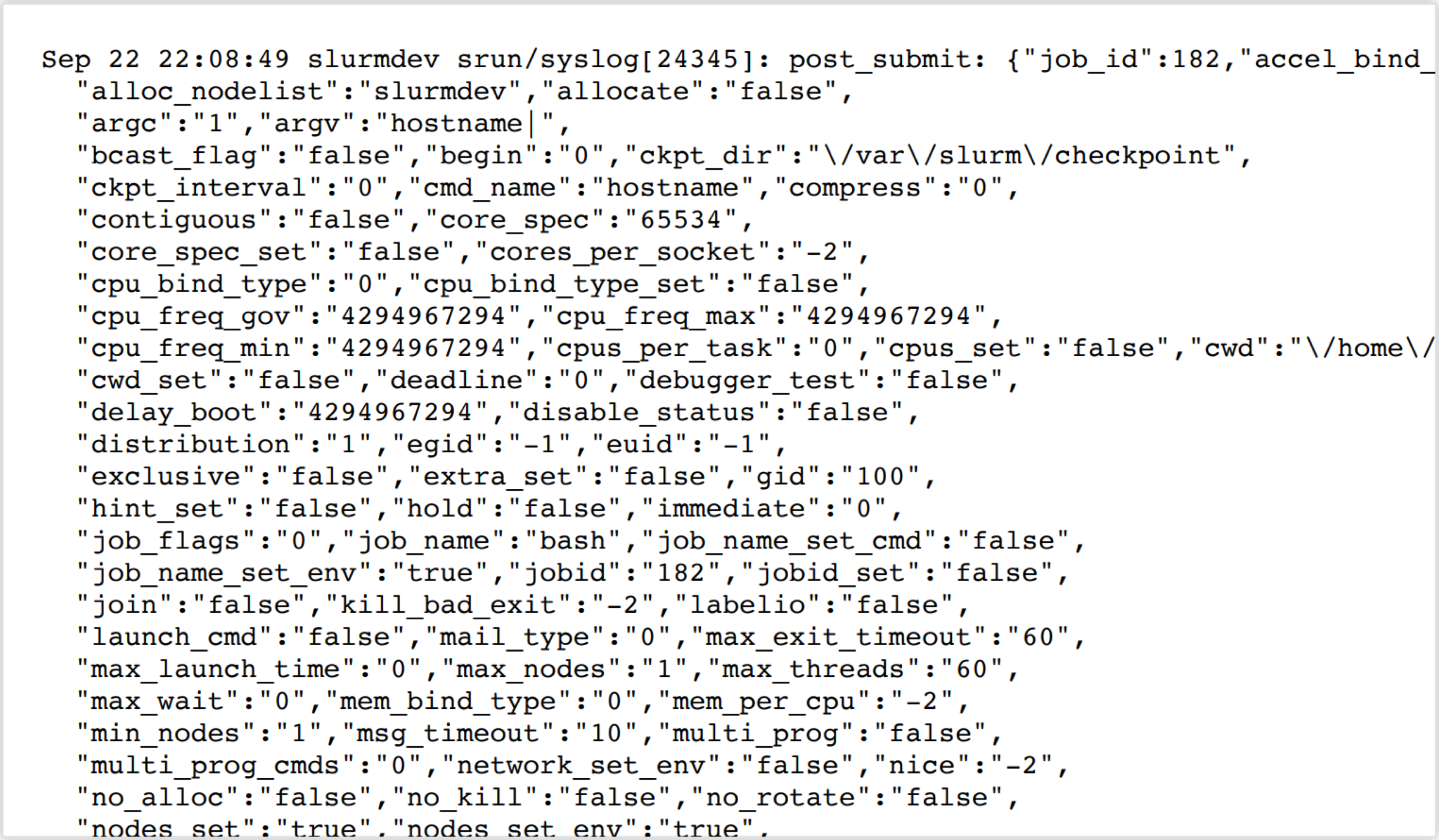
cli_filter/lua example

```
function slurm_cli_pre_submit(cli_type, options)
  -- dangerous to run on controller node, may get stuck if PFS misbehaving
  local fs_quota_auth = os.execute("/usr/bin/myquota -c")
  if fs_quota_auth ~= 0 then
    slurm.log_error("ERROR: in violation of quota limits. " ..
                    "Job submission disabled.")
    return slurm.ERROR
  end
  -- TODO: check options['workdir'] to check aux filesystem quotas

  if cli_type == CLI_ALLOC and options["qos"] ~= nil
          and options["qos"] == "interactive" then

    options["immediate"] = 30
  end

  local balance = io.popen("/something/to/get/external/accounting")
  local time_requested = calculate_time(options)
  if balance > time_requested and not options["parsable"] then
    slurm.log_info("WARNING: Low on allocation, your job moving to scavenger")
  end
  return slurm.SUCCESS
end
```

cli_filter post_submit()

- **post_submit**
  - Runs once per job-pack per cli_filter plugin per CLI execution
  - Non-zero exit will attempt to terminate job (invalid for sbatch)
  - Runs after all option processing but before slurmctld message preparation (can change options here)
- **Implementations**
  - cli_filter/lua plugin can get data and log it cli_filter/syslog dumps json record of submission to syslog

cli_filter/syslog Example output



```
Sep 22 22:08:49 slurmdev srun/syslog[24345]: post_submit: {"job_id":182,"accel_bind_
  "alloc_nodelist":"slurmdev","allocate":"false",
  "argc":"1","argv":"hostname",
  "bcast_flag":"false","begin":"0","ckpt_dir":"\/var\/slurm\/checkpoint",
  "ckpt_interval":"0","cmd_name":"hostname","compress":"0",
  "contiguous":"false","core_spec":"65534",
  "core_spec_set":"false","cores_per_socket":"-2",
  "cpu_bind_type":"0","cpu_bind_type_set":"false",
  "cpu_freq_gov":"4294967294","cpu_freq_max":"4294967294",
  "cpu_freq_min":"4294967294","cpus_per_task":"0","cpus_set":"false","cwd":"\/home\/
  "cwd_set":"false","deadline":"0","debugger_test":"false",
  "delay_boot":"4294967294","disable_status":"false",
  "distribution":"1","egid":"-1","euid":"-1",
  "exclusive":"false","extra_set":"false","gid":"100",
  "hint_set":"false","hold":"false","immediate":"0",
  "job_flags":"0","job_name":"bash","job_name_set_cmd":"false",
  "job_name_set_env":"true","jobid":"182","jobid_set":"false",
  "join":"false","kill_bad_exit":"-2","labelio":"false",
  "launch_cmd":"false","mail_type":"0","max_exit_timeout":"60",
  "max_launch_time":"0","max_nodes":"1","max_threads":"60",
  "max_wait":"0","mem_bind_type":"0","mem_per_cpu":"-2",
  "min_nodes":"1","msg_timeout":"10","multi_prog":"false",
  "multi_prog_cmds":"0","network_set_env":"false","nice":"-2",
  "no_alloc":"false","no_kill":"false","no_rotate":"false",
  "nodes_set":"true","nodes_set_env":"true",
```

Use Case 1: ALTD runtime library tracking

- **cli_filter/lua allows the runtime library tracking**
  - Automatic Library Tracking Database (ALTD) is used to track the library usage at NERSC by wrapping the ld (linker) and srun (job launcher) commands at compile and runtime. However, wrappers are often not desirable, especially at runtime.

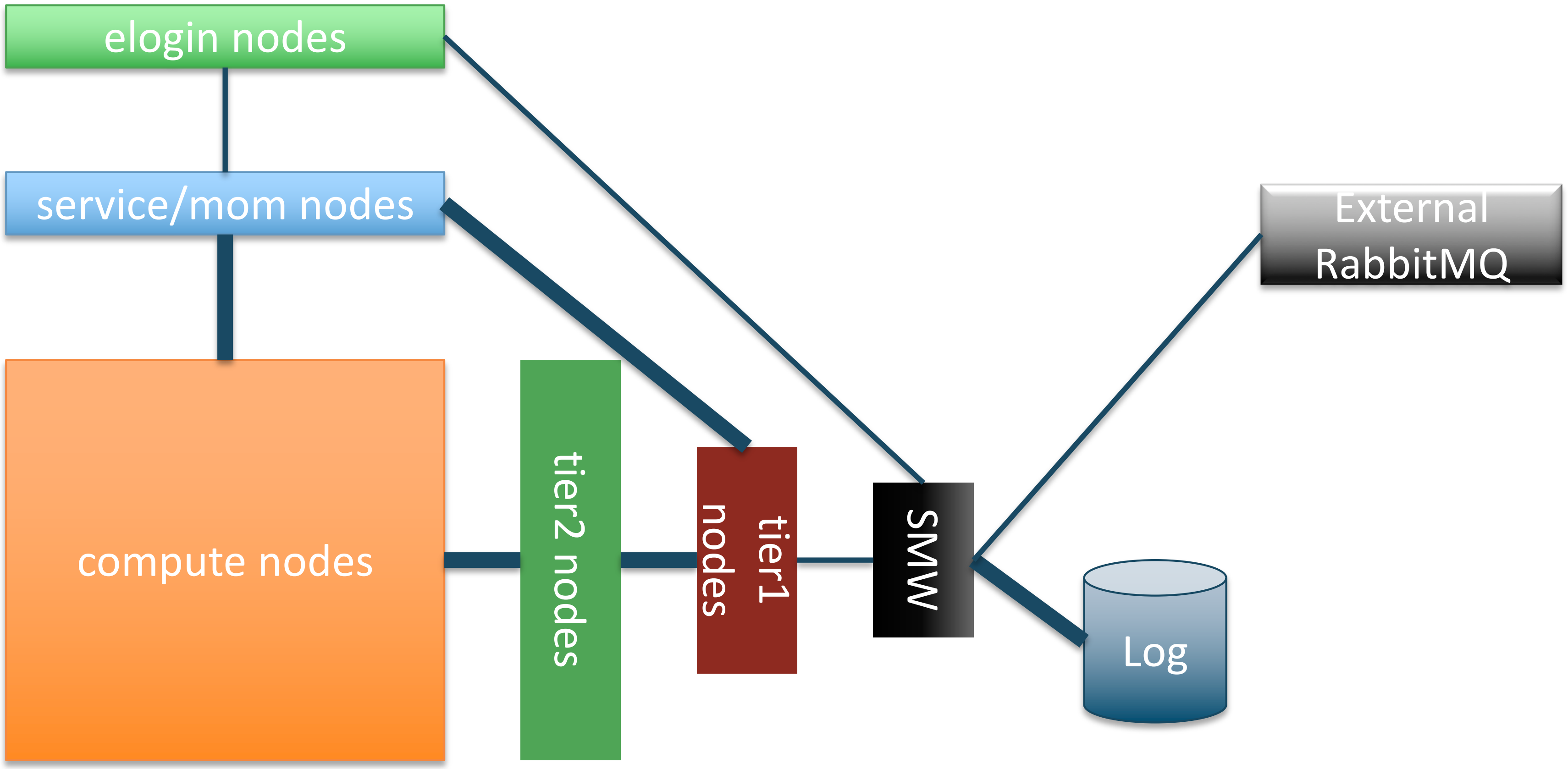
Just need to collect the Tag_id.

```
a.zz217@cori11:~/tests> objdump -s -j .altd a.out

a.out:     file format elf64-x86-64

Contents of section .altd:

 0000 414c5444 5f4c696e 6b5f496e 666f0000  ALTD_Link_Info..
 0010 00000000 00005665 7273696f 6e3a322e  ......Version:2.
 0020 303a004d 61636869 6e653a63 6f72693a  0:.Machine:cori:
 0030 00546167 5f69643a 39653131 35313965  .Tag_id:9e11519e
 0040 2d333031 652d3433 37392d61 3736652d  -301e-4379-a76e-
 0050 66346265 36363134 34613132 3a005965  f4be66144a12:.Ye
 0060 61723a32 3031383a 00000000 00000000  ar:2018:........
 0070 414c5444 5f4c696e 6b5f496e 666f5f45  ALTD_Link_Info_E
 0080 6e6400                               nd.
```

Use case 2: Application workload analysis

- **cli_filter/syslog used to get the detailed workload analysis for all srun/sbatch/salloc executions**
  - Slurm can track the application names that are run with the srun, however, it is often desirable to know more details about how the applications are used, e.g., what are the most commonly used code paths, which provides developers with a more targeted code optimizations
- **cli_filter/lua used to collect workload-specific data**
  - e.g, VASP, we can collect its input control files (not atomic positions), to find out the commonly used job types (code paths) to guide the application readiness effort for the developers.

Gathering Data via LLM



Setting up LLM Rules

Service Node add /var/spool/rsyslog/local-rules/ruleset.conf

```
### Direct SlurmCli Syslog
$RuleSet rule.slurmcli.syslog
$MainMsgQueueFileName slurmcli_syslog
$MainMsgQueueDiscardMark 880000
$MainMsgQueueHighWatermark 660000
$RulesetCreateMainQueue on
$IncludeConfig /var/spool/rsyslog/local-rules/always.conf
$IncludeConfig /var/spool/rsyslog/local-rules/rule.slurmcli.syslog.conf
$IncludeConfig /var/spool/rsyslog/rsyslog-forward.conf
$IncludeConfig /var/spool/rsyslog/rsyslog-options.conf
*.* -?file-userslurm;format-rfc5424
$inputPTCPServerBindRuleset rule.slurmcli.syslog
$InputPTCPServerRun 5187
$InputUDPServerBindRuleset rule.slurmcli.syslog
$UDPServerRun 5187
```

Setting up LLM Rules

SMW add /var/spool/rsyslog/local-rules/main.conf

```
$template file-slurmcli, "/var/opt/cray/log/%MSGID%/slurmcli/%APP-NAME%-
%$YEAR%%$MONTH%%$DAY%"

if $structured-data == '[slurmcli@34]' then
   -?file-slurmcli;format-message
& stop
```

Using LLM

Have cli_filter plugins use script to:

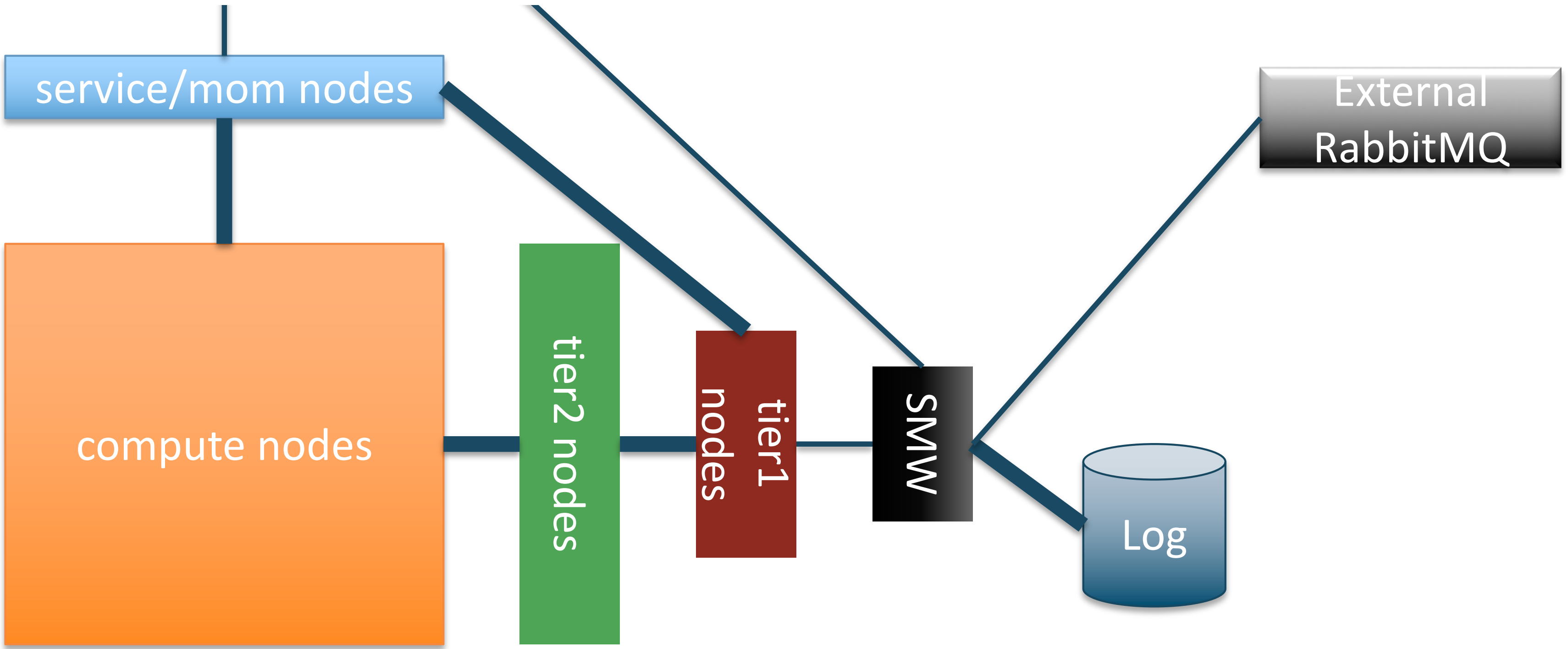
- **On elogin, use syslog**
- **On service node, use syslog**
- **On compute, send UDP message to randomly selected tier2 node on configured LLM port**

RFC 5424 formatted message like:

<34>1 2018-05-24T00:51:15.003Z c0-0c0s1n1 altd 42 bootsession [slurmcli@34] test userslurm message

Future Work

- **Upcoming**
  - Working with SchedMD to explore cli_filter merge options
  - Finish modeling work on test system
  - Export via rsyslog to site data collect (RabbitMQ)



NERSC