TECHNISCHE UNIVERSITÄT DRESDEN

Real-time job monitoring using an extended slurmctld generic plugin

Introducing the plugin architecture SPACE

Mike Arnhold (mike.arnhold@tu-dresden.de)
Ulf Markwardt (ulf.markwardt@tu-dresden.de)
Danny Rotscher (danny.rotscher@tu-dresden.de)

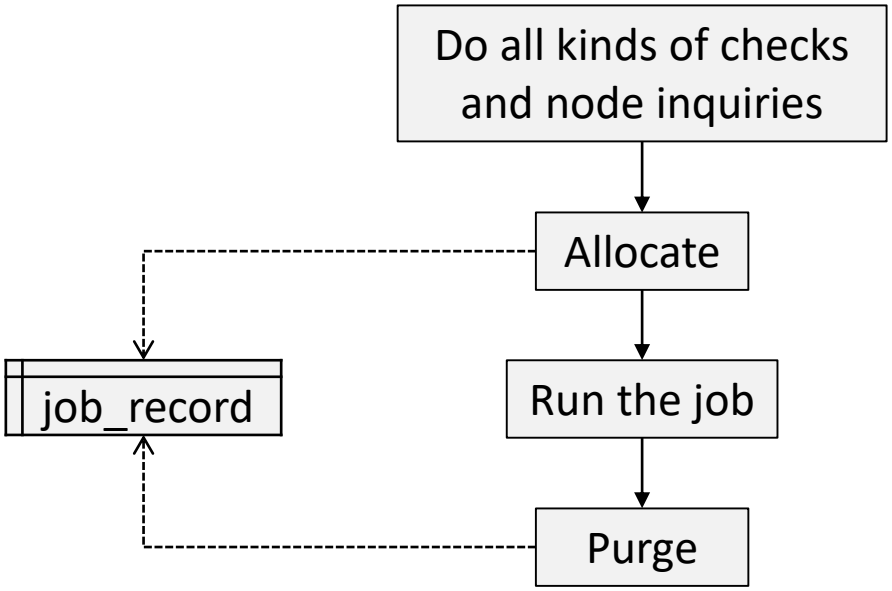
ZIH
Center for Information Services & High Performance Computing

Motivation

- tracking of life SLURM data
  - internal interface (performance monitoring)
  - public interface (web service)

- desired SLURM interface properties
  - synchronous and asynchronous access
  - as much job data as possible
  - avoid:
    - frequent alloc system calls
    - overhead from custom string operation and/or data conversions
    - high performance impact on slurmctld
    - accidental manipulation of internal SLURM job data

Job Scheduling

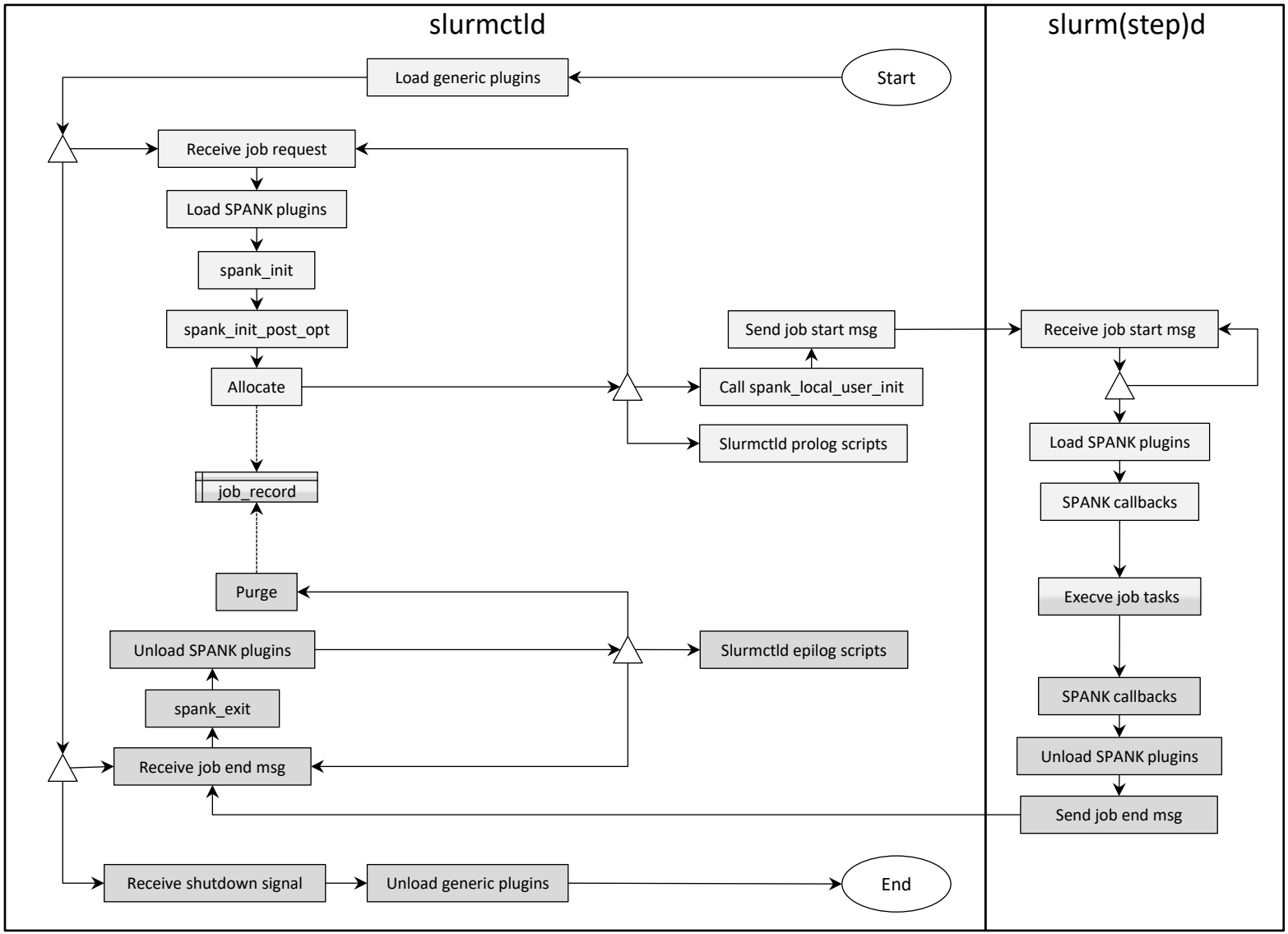


- job_record
  - survives job run time
  - contains all relevant job information

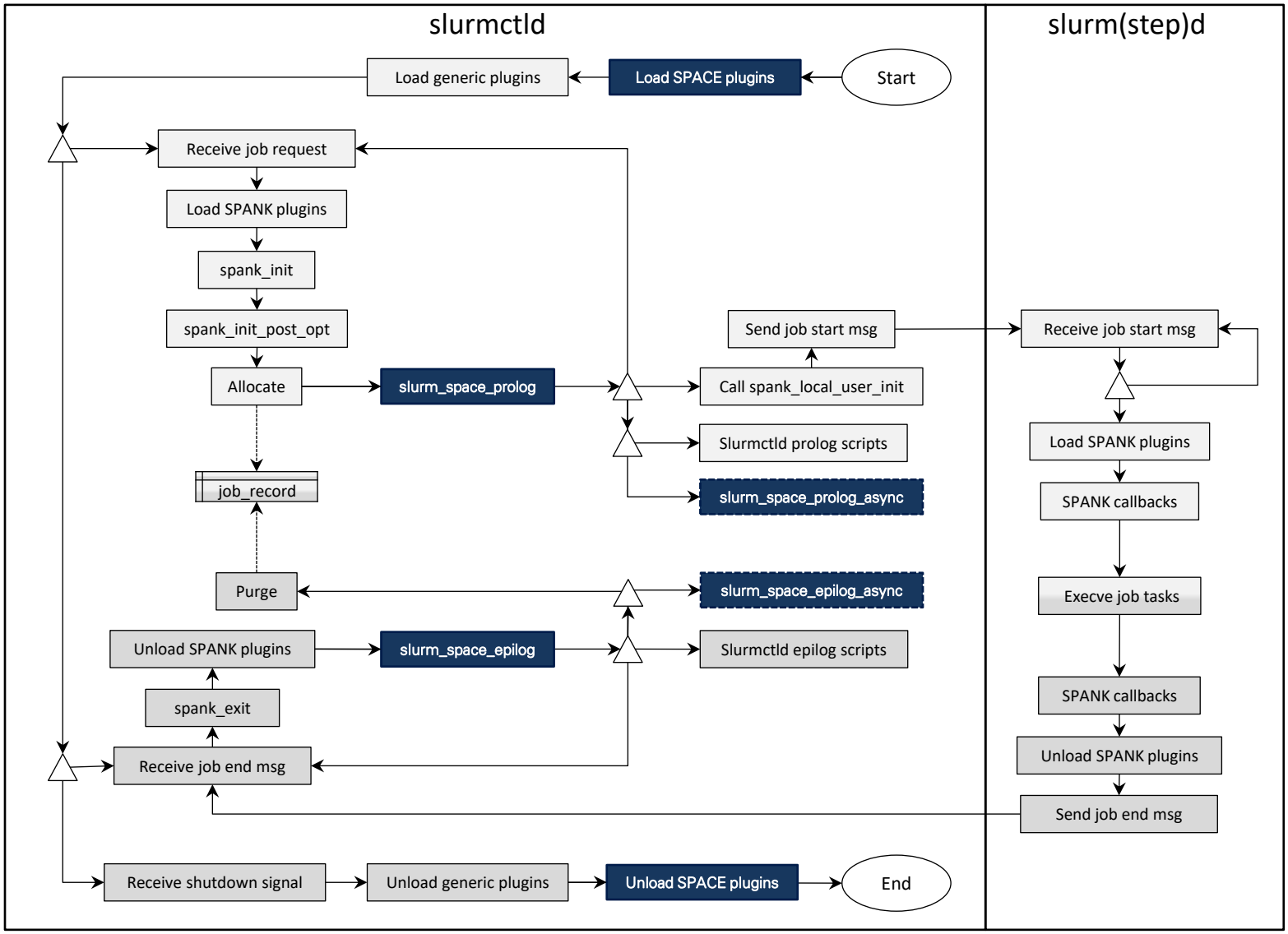
Stackable Plug-in Architecture for SLURM Controller Extension

- similarities to SPANK
  - similar interface (space.h)
  - compilation only against space.h
- differences to SPANK
  - each plugin only loaded once
  - usable as slurmctld generic plugin
  - read-only getter

SLURM Plugins



SLURM Plugins



SPACE Plugins - Configuration

/etc/slurm/plugstack.conf

```
# SPANK and SPACE config file

required /opt/slurm/space/lib/slurmctld_rabbitmq.so /etc/slurm/rabbitmq.conf
```

/etc/slurm/rabbitmq.conf

```
# RabbitMQ config file for SPACE plugin

port "5672"
hostname "localhost"
virtualHost "vhost.slurm"
userName "slurm"
userPass "slurm"
exchange "exchange.job_events"
routingkeyProlog "prolog"
routingkeyEpilog "epilog"
```

SPACE Plugins - Callbacks

```
typedef struct space_handle * space_t;
typedef int (space_f) (space_t space, int ac, char *argv[]);

space_f slurm_space_init;
space_f slurm_space_fini;
space_f slurm_space_job_prolog;
space_f slurm_space_job_epilog;
space_f slurm_space_job_prolog_async;
space_f slurm_space_job_epilog_async;
```

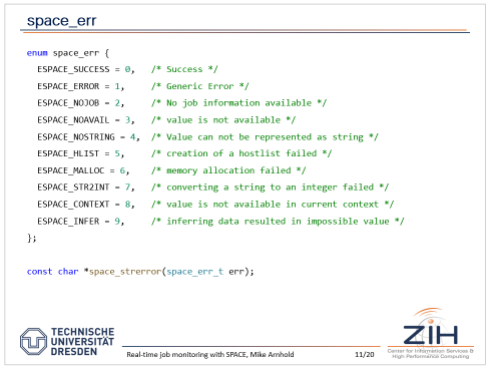
SPACE Plugin

```c
#include <slurm/space.h>                        /* must-have */

#define PLUGIN_NAME "slurmctld/slug_2018: "   /* log convenience */
SPACE_PLUGIN(slurmctld/slug_2018);             /* must-have */

int slurm_space_job_prolog(space_t space, int ac, char *argv[])
{
    slurm_debug(PLUGIN_NAME “enter %s", __func__);
     /* do stuff */
    slurm_debug(PLUGIN_NAME “leave %s", __func__);
    return 0;
}
```

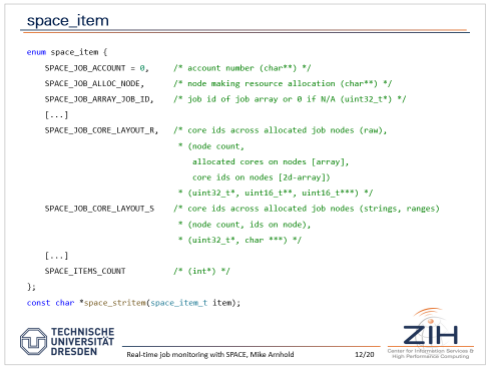
SPACE Plugin

```
space_err_t space_get_item(space_t space, space_item_t item, ...);
```

```
typedef enum space_err space_err_t;
```




```
typedef enum space_item space_item_t;
```



space_err

```
enum space_err {
  ESPACE_SUCCESS = 0,    /* Success */
  ESPACE_ERROR = 1,      /* Generic Error */
  ESPACE_NOJOB = 2,      /* No job information available */
  ESPACE_NOAVAIL = 3,    /* value is not available */
  ESPACE_NOSTRING = 4,   /* Value can not be represented as string */
  ESPACE_HLIST = 5,      /* creation of a hostlist failed */
  ESPACE_MALLOC = 6,     /* memory allocation failed */
  ESPACE_STR2INT = 7,    /* converting a string to an integer failed */
  ESPACE_CONTEXT = 8,    /* value is not available in current context */
  ESPACE_INFER = 9,      /* inferring data resulted in impossible value */
};

const char *space_strerror(space_err_t err);
```

space_item

```
enum space_item {
    SPACE_JOB_ACCOUNT = 0,      /* account number (char**) */
    SPACE_JOB_ALLOC_NODE,       /* node making resource allocation (char**) */
    SPACE_JOB_ARRAY_JOB_ID,     /* job id of job array or 0 if N/A (uint32_t*) */
    [...]
    SPACE_JOB_CORE_LAYOUT_R,    /* core ids across allocated job nodes (raw),
                                 * (node count,
                                    allocated cores on nodes [array],
                                    core ids on nodes [2d-array])
                                 * (uint32_t*, uint16_t**, uint16_t***) */
    SPACE_JOB_CORE_LAYOUT_S     /* core ids across allocated job nodes (strings, ranges)
                                 * (node count, ids on node),
                                 * (uint32_t*, char ***) */
    [...]
    SPACE_ITEMS_COUNT           /* (int*) */
};
const char *space_stritem(space_item_t item);
```

SPACE Plugin – Retrieve Single Value

```
int slurm_space_job_prolog(space_t space, int ac, char *argv[])
{
    uint32_t jid;
    space_err_t get_rc;
    int rc = 0;
    get_rc = space_get_item(space, SPACE_JOB_ID, &jid);
    if(get_rc == ESPACE_SUCCESS){
        slurm_debug3(PLUGIN_NAME "job id: %u", jid);
    } else {
        slurm_error(PLUGIN_NAME "id failed with %s", space_strerror(rc));
        rc = -1;
    }
    return rc;
}
```

SPACE Plugin – Retrieve Arrays

```
uint32_t node_count;
uint16_t *core_counts;
uint16_t **core_ids;
space_err_t get_rc;

get_rc = space_get_item(space, SPACE_JOB_CORE_LAYOUT_R,
                        &node_count, &core_counts, &core_ids);

if(get_rc == ESPACE_SUCCESS){
    /* do smth with it,
}
```

SPACE Plugin – Retrieve Value as String

```
space_err_t space_sget_item(space_t space,
                            space_item_t item,
                            char *buff,
                            int buff_len,
                            int *written);
```

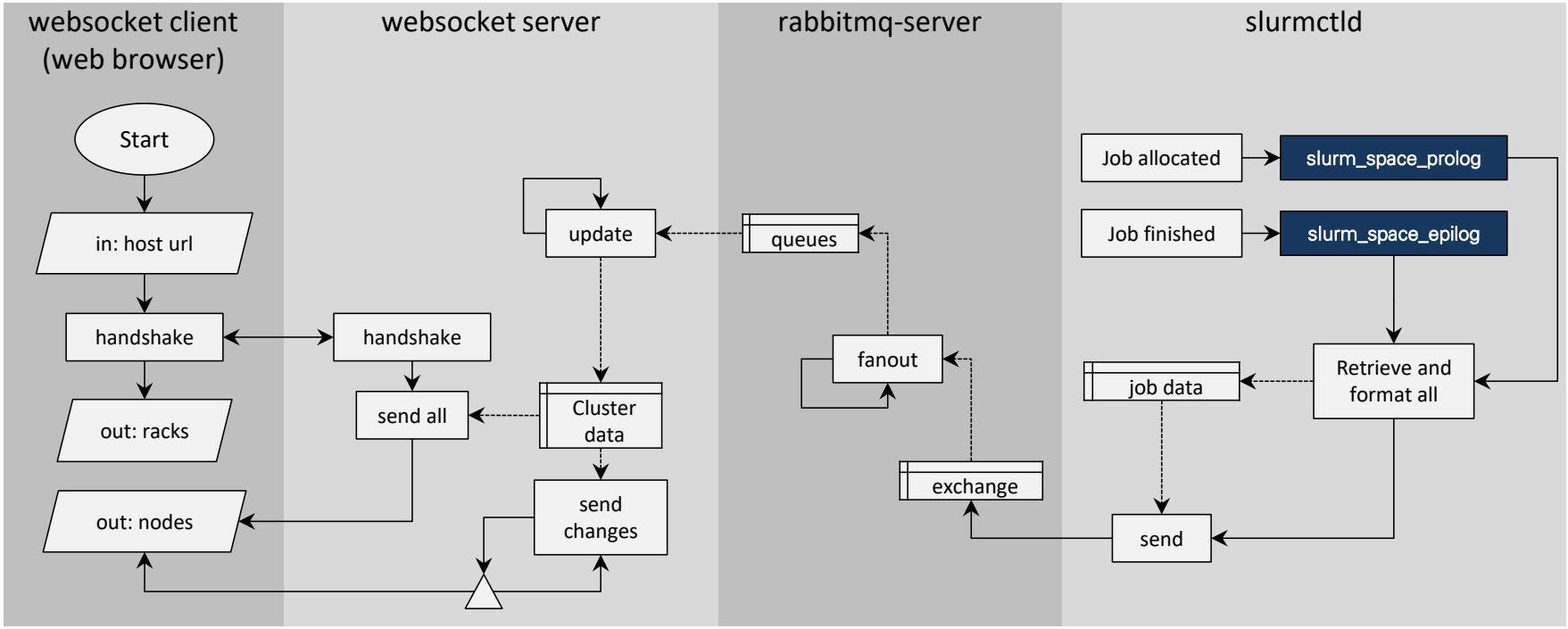
SPACE Plugin – Retrieve Value as String

```c
#define BUFF_LEN 512
int slurm_space_job_prolog(space_t space, int ac, char *argv[])
{
    int written = -1;
    char buff[BUFF_LEN] = "";

    if(!space_sget_item(space, SPACE_JOB_CORE_IDS,
                        buff, BUFF_LEN, &written)){
        /* do smth with it */
    }


    return written != -1;
}
```

SPACE Plugin – Retrieve Value as String

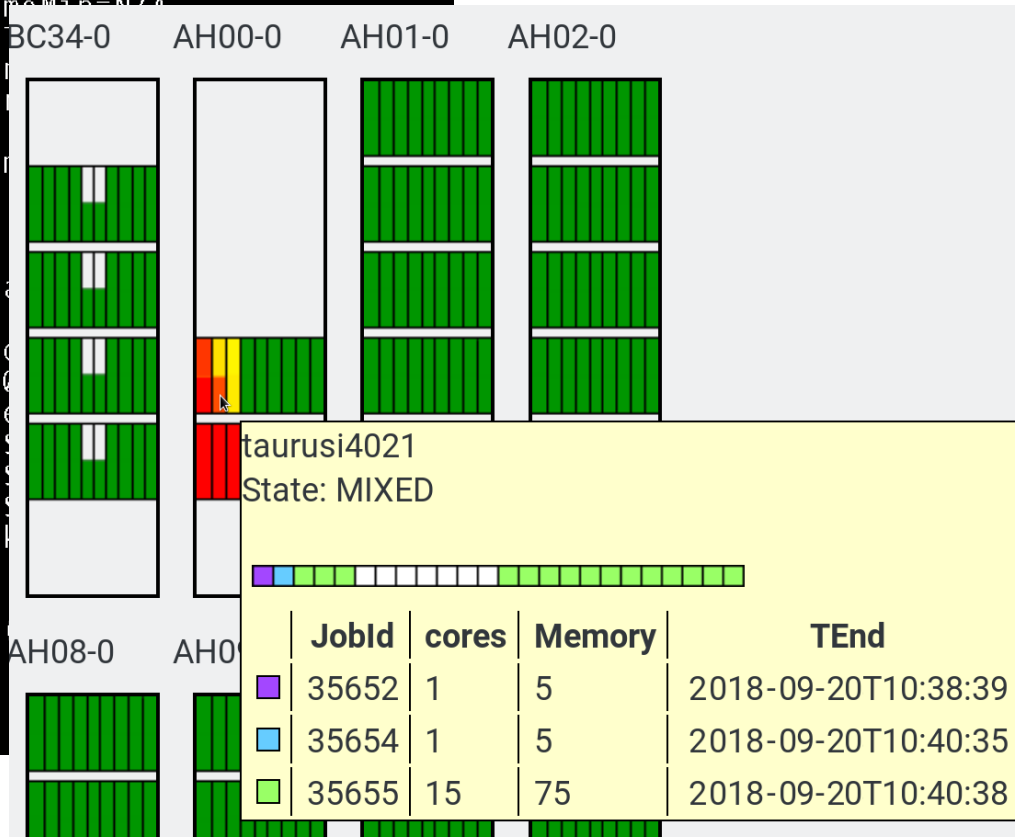
```c
#define BUFF_LEN 512
int slurm_space_job_prolog(space_t space, int ac, char *argv[])
{
    int written = -1;
    char buff[BUFF_LEN] = "";
    space_item_t item = 0;

    while(item < SPACE_ITEMS_COUNT &&
          !space_sget_item(space, item, buff, BUFF_LEN, &written)){
        /* do smth with it */
        ++item;
    }
    return item == SPACE_ITEMS_COUNT;
}
```

HPCView Framework



HPCView Framework

```
[root@bull-admin ~]# scontrol show -d job 35655
JobId=35655 JobName=sleep
   UserId=root(0) GroupId=root(0) MCS_label=N/A
   Priority=4294901743 Nice=0 Account=root QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=0 Reboot=0 ExitCode=0:0
   DerivedExitCode=0:0
   RunTime=00:02:04 TimeLimit=UNLIMITED Tim
   SubmitTime=2018-09-20T07:25:38 Eligible
   StartTime=2018-09-20T07:25:38 EndTime=U
   PreemptTime=None SuspendTime=None SecsP
   LastSchedEval=2018-09-20T07:25:38
   Partition=haswell AllocNode:Sid=bull-ad
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=taurusi[4020-4024]
   BatchHost=localhost
   NumNodes=5 NumCPUs=30 NumTasks=5 CPUs/T
   TRES=cpu=30,mem=150M,node=5,billing=30
   Socks/Node=1 NtasksPerN:B:S:C=0:0:*:1 C
     Nodes=taurusi4020 CPU_IDs=12-23 Mem=6
     Nodes=taurusi4021 CPU_IDs=2-4,12-23 M
     Nodes=taurusi4022 CPU_IDs=2 Mem=5 GRE
     Nodes=taurusi4023 CPU_IDs=1 Mem=5 GRE
     Nodes=taurusi4024 CPU_IDs=0 Mem=5 GRE
   MinCPUsNode=1 MinMemoryCPU=5M MinTmpDis
   Features=(null) DelayBoot=00:00:00
   Gres=(null) Reservation=(null)
   OverSubscribe=OK Contiguous=0 Licenses=
   Command=sleep
   WorkDir=/root
   Power=
```



JobId	cores	Memory	TEnd
35652	1	5	2018-09-20T10:38:39
35654	1	5	2018-09-20T10:40:35
35655	15	75	2018-09-20T10:40:38

Advantages of SPACE

- flexible live interface for job data
- extensive job information
- blocking and non-blocking callbacks
  - plugin finishes before job start

    vs.
  - plugin finishes independent of job run
- minimal performance impact