·····

··Slurm·····8MiB·····

Slurm ····· CPU ····· task ··· CPU·····
MPI rank ·····

·····

--mem ····· MiB·--mem-per-cpu ····· CPU ····· --mem-per-gpu ···CPU ·
····· CPU ····· MaxMemPerCPU ··Slurm ····· CPU ·· cpus-per-task··
····· ReqTRES/AllocTRES ···

·· | ···· | ···

--ntasks | ··/···· | MPI·srun·····

--cpus-per-task | ·····CPU· | ·····

--mem | ····· | ····ReqTRES·AllocTRES

--mem-per-cpu | ·····CPU·· | CPU···MaxMemPerCPU

·····

····· 8 MiB ·····CPU ····· cgroup v2 ·····
·····/····· CHECKED_CGROUP····· cgroup
·

```
from pathlib import Path
import json, os, resource

report = {'pid':os.getpid(), 'allocation':{}}
for key in ('SLURM_JOB_ID','SLURM_NTASKS','SLURM_CPUS_PER_TASK',
            'SLURM_MEM_PER_NODE','SLURM_MEM_PER_CPU'):
    report['allocation'][key] = os.environ.get(key)
status = Path('/proc/self/status').read_text().splitlines()
report['allowed'] = [line for line in status if line.startswith(
    ('Cpus_allowed_list:','Mems_allowed_list:'))]
report['cgroup_membership'] = Path('/proc/self/cgroup').read_text()
cg = os.environ.get('CHECKED_CGROUP')
report['cgroup_values'] = {}
if cg:
    for name in ('memory.max','memory.current','memory.events'):
        try: report['cgroup_values'][name] = (Path(cg)/name).read_text()
        except OSError as err: report['cgroup_values'][name] = {'read_error':type(err).__name__}
else:
    report['cgroup_note'] = '·····'
buf = bytearray(8 * 1024 * 1024)
for offset in range(0, len(buf), 4096): buf[offset] = 1
report['touched_payload_bytes'] = len(buf)
report['linux_self_maxrss_kib'] = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss
print(json.dumps(report, indent=2))
```

··· quota_probe.py····· Slurm ····· JSON·
·····

```
#!/usr/bin/env bash
#SBATCH --job-name=quota-probe
#SBATCH --output=quota-%j.out
set -euo pipefail
: "${SLURM_JOB_ID:?···Slurm··}"
out="$PWD/quota-probe-$SLURM_JOB_ID"
mkdir "$out"
scontrol --version > "$out/version.txt"
scontrol show job "$SLURM_JOB_ID" > "$out/job.txt"
srun --ntasks=1 python quota_probe.py > "$out/probe.json" 2> "$out/probe.err"
python - "$out/probe.json" <<'PY'
import json, sys
with open(sys.argv[1]) as source:
    result = json.load(source)
if result.get('touched_payload_bytes') != 8 * 1024 * 1024:
    raise SystemExit('·····')
PY
```

·····

····· submit_probe.sh·····CPU····task ····· 512 MiB····· CPU 256
 MiB ··· 2 CPU····· MODE=node PARTITION=···· bash submit_probe.sh · MODE=cpu PARTITION=···· bash
submit_probe.sh····· CPU ····· JobID ·· scontrol
 show job ····· CPU ··

```
set -euo pipefail
: "${MODE:?··node·cpu}" "${PARTITION:?·····}"
case "$MODE" in
  node) memory=(--mem=512M) ;;
  cpu) memory=(--mem-per-cpu=256M) ;;
  *) exit 2 ;;
esac
sbatch --partition="$PARTITION" --nodes=1 --ntasks=1 \
  --cpus-per-task=2 --time=00:05:00 "${memory[@]}" quota.sbatch
# ·····submit_probe.sh·····MODE=node·MODE=cpu···
```

…… 8 MiB…… R/Python ……Slurm …… enforcement …… ····· cgroup ·····

……

…… 8 MiB ……
……MaxRSS · cgroup ……
……

Slurm …… memory.max ……
…… enforcement …… OOM ……

·· CPU …… JobID……MaxMemPerCPU…… cgroup…… OOM …… rank·GPU ·····
……

····

SchedMD··sbatch·mem·mem-per-cpu·CPU···· https://slurm.schedmd.com/sbatch.html

SchedMD··sacct·ReqTRES·AllocTRES····· https://slurm.schedmd.com/sacct.html

Linux····cgroup v2…… https://docs.kernel.org/admin-guide/cgroup-v2.html