University of Glasgow

Condor Cgroups

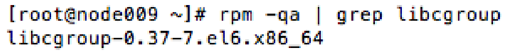
—— *Gang Qin, Gareth Roy*

Nov. 17th , 2014

Condor-Cgroups (1)

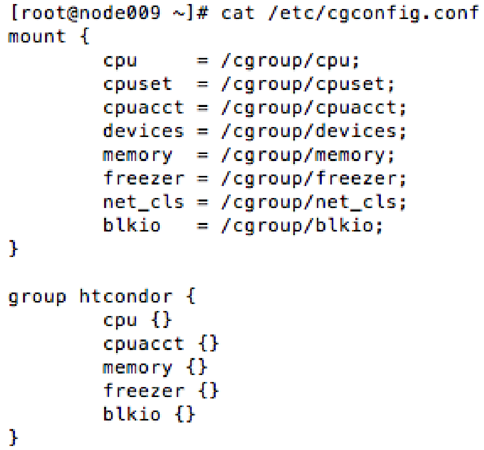
- Control Groups (Cgroups)
  - Linux kernel feature to limit/account/isolate resources usage among user-defined groups of tasks(processes) .
  - Available Resource Controllers (subsystems):
    - Block-I/O, cpu/cpuacct/cpuset/devices/freezer/memory/net_cls/net_prio/ns
- Installation/Configuration/Testing

①



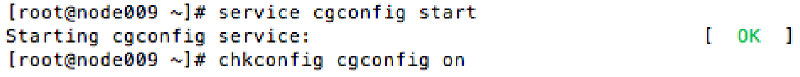
```
[root@node009 ~]# rpm -qa | grep libcgroup
libcgroup-0.37-7.el6.x86_64
```

②



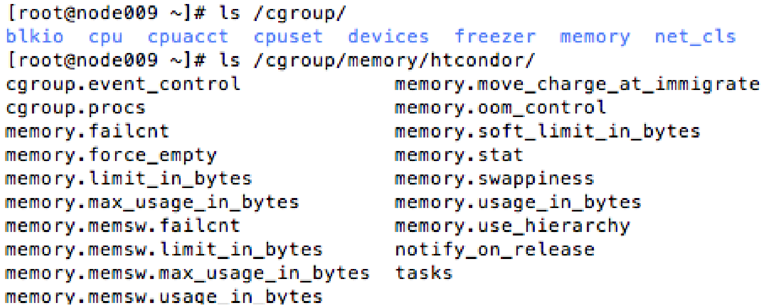
```
[root@node009 ~]# cat /etc/cgconfig.conf
mount {
        cpu     = /cgroup/cpu;
        cpuset  = /cgroup/cpuset;
        cpuacct = /cgroup/cpuacct;
        devices = /cgroup/devices;
        memory  = /cgroup/memory;
        freezer = /cgroup/freezer;
        net_cls = /cgroup/net_cls;
        blkio   = /cgroup/blkio;
}

group htcondor {
        cpu {}
        cpuacct {}
        memory {}
        freezer {}
        blkio {}
}
```

③



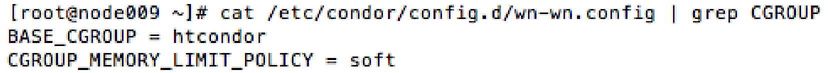
```
[root@node009 ~]# service cgconfig start
Starting cgconfig service:                                 [  OK  ]
[root@node009 ~]# chkconfig cgconfig on
```

④



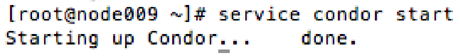
```
[root@node009 ~]# ls /cgroup/
blkio  cpu  cpuacct  cpuset  devices  freezer  memory  net_cls
[root@node009 ~]# ls /cgroup/memory/htcondor/
cgroup.event_control              memory.move_charge_at_immigrate
cgroup.procs                      memory.oom_control
memory.failcnt                    memory.soft_limit_in_bytes
memory.force_empty                memory.stat
memory.limit_in_bytes             memory.swappiness
memory.max_usage_in_bytes         memory.usage_in_bytes
memory.memsw.failcnt              memory.use_hierarchy
memory.memsw.limit_in_bytes       notify_on_release
memory.memsw.max_usage_in_bytes   tasks
memory.memsw.usage_in_bytes
```

⑤



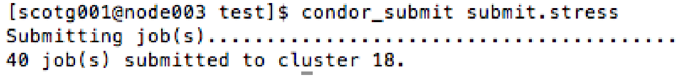
```
[root@node009 ~]# cat /etc/condor/config.d/wn-wn.config | grep CGROUP
BASE_CGROUP = htcondor
CGROUP_MEMORY_LIMIT_POLICY = soft
```

⑥



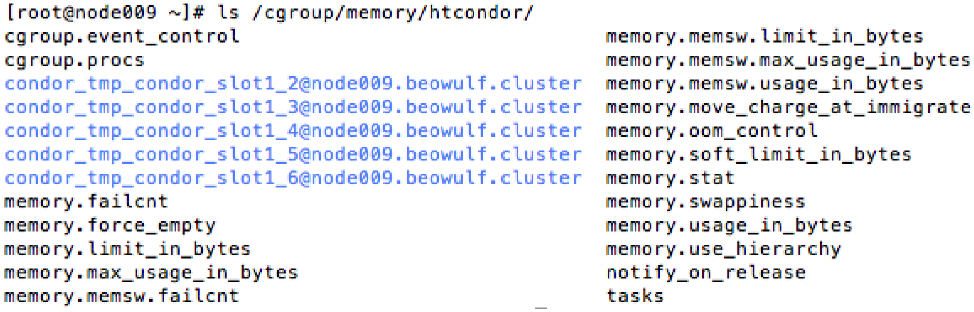
```
[root@node009 ~]# service condor start
Starting up Condor...     done.
```

⑦



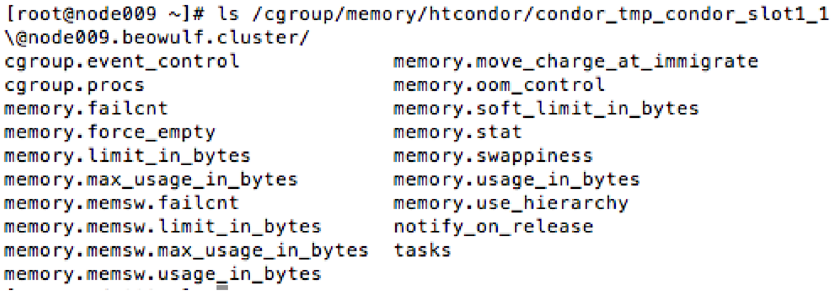
```
[scotg001@node003 test]$ condor_submit submit.stress
Submitting job(s)........................................
40 job(s) submitted to cluster 18.
```

⑧



```
[root@node009 ~]# ls /cgroup/memory/htcondor/
cgroup.event_control                          memory.memsw.limit_in_bytes
cgroup.procs                                  memory.memsw.max_usage_in_bytes
condor_tmp_condor_slot1_2@node009.beowulf.cluster  memory.memsw.usage_in_bytes
condor_tmp_condor_slot1_3@node009.beowulf.cluster  memory.move_charge_at_immigrate
condor_tmp_condor_slot1_4@node009.beowulf.cluster  memory.oom_control
condor_tmp_condor_slot1_5@node009.beowulf.cluster  memory.soft_limit_in_bytes
condor_tmp_condor_slot1_6@node009.beowulf.cluster  memory.stat
memory.failcnt                                memory.swappiness
memory.force_empty                            memory.usage_in_bytes
memory.limit_in_bytes                         memory.use_hierarchy
memory.max_usage_in_bytes                     notify_on_release
memory.memsw.failcnt                          tasks
```

⑨



```
[root@node009 ~]# ls /cgroup/memory/htcondor/condor_tmp_condor_slot1_1
\@node009.beowulf.cluster/
cgroup.event_control              memory.move_charge_at_immigrate
cgroup.procs                      memory.oom_control
memory.failcnt                    memory.soft_limit_in_bytes
memory.force_empty                memory.stat
memory.limit_in_bytes             memory.swappiness
memory.max_usage_in_bytes         memory.usage_in_bytes
memory.memsw.failcnt              memory.use_hierarchy
memory.memsw.limit_in_bytes       notify_on_release
memory.memsw.max_usage_in_bytes   tasks
memory.memsw.usage_in_bytes
```

Condor-Cgroups (2)

- How condor use cgroups?
  - Condor put each job into a dedicated cgroup for selected subsystems
  - Control cpu usage at job level:
    - Writing cpu.shares with fixed/dynamic value for static/partitionable slots
  - Control Memory usage at job level:
    - Writing **memory.limit_in_bytes** and **memory.soft_limit_in_bytes**:
    - Three policies for memory control
      - » **none**: No limit applied
      - » **soft**: job can access memory than allocated if there are still free physical memory available in the system
      - » **hard**: job can't access more physical memory than allocated
  - Test: For a job which requires 1000MB memory, we have:

Policy	Memory.limit_in_bytes	Memory_soft_limit_in_bytes
none	9223372036854775807	9223372036854775807
soft	9223372036854775807	1073741824
hard	1073741824	9223372036854775807

- Motivation for studying info collected by Cgroups
  - Get better knowledge of jobs to identify suspicious/broken jobs
  - Current studies focus on jobs' memory footprints

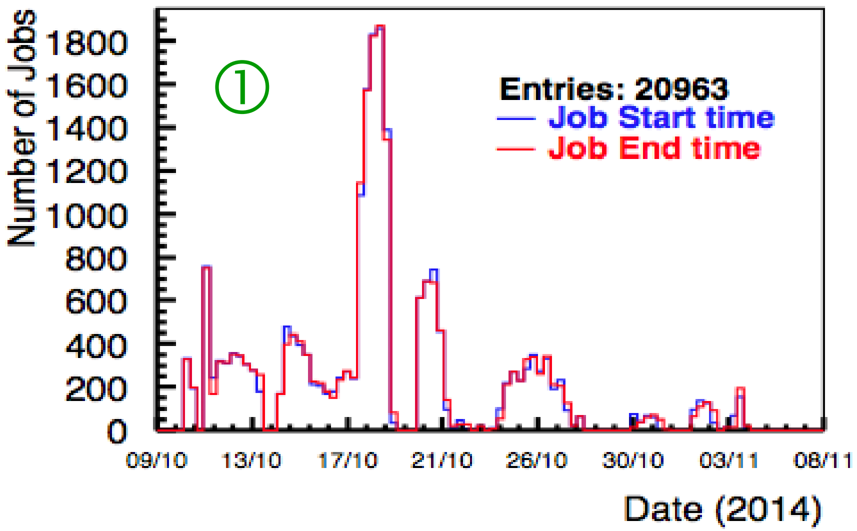
Memory Footprints of Jobs

(4)

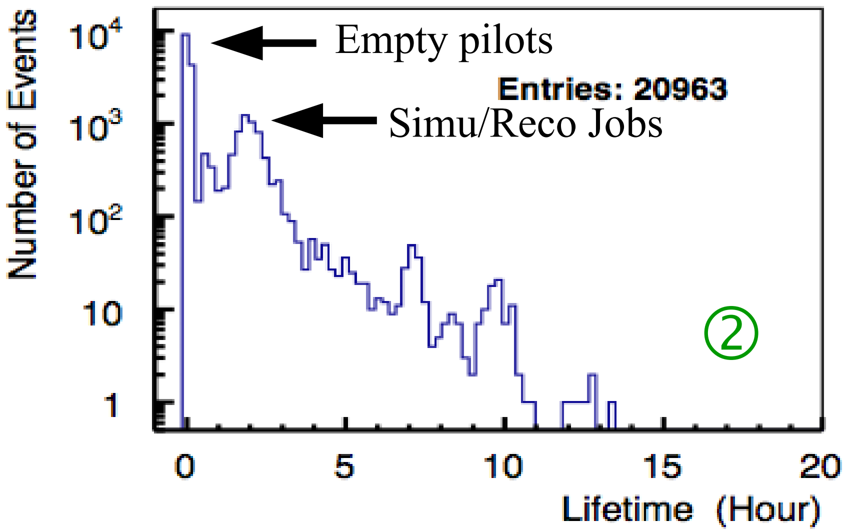
- Condor Cluster
  - Status: Fully in production instance since early Aug, receiving ~ 400k jobs
  - Scale: 1 ARC-CE (8core), 1 condor central server (8core), 16 worker-nodes (744cores)
- Condor Database
  - Mysql database setup to select/record historical info of condor jobs
    - ClusterId/GlobalJobId/JobStatus/ExitCode/LastJobStatus/RequestCpus/RequestMemory/JobMemoryLimit/JobTimeLimit/User and etc..
  - Updates at 5:00 every morning
- Data collection
  - Every minute on each WN, Cgmemd collects:
    - Timestamp, GlobalJobId(batchID), requested_cpus
    - RSS: anomymous and swap cache, not including tmpfs (shmem)
    - Cache: page cache, including tmpfs(shmem)
    - Mapped_file: size of memory-mapped mapped files, including tmpfs(shmem)
    - Swap: swap usage
- Analysis
  - Currently focus on ATLAS Multicore Simu/Reco jobs

Overview of good ATLAS Multicore Jobs

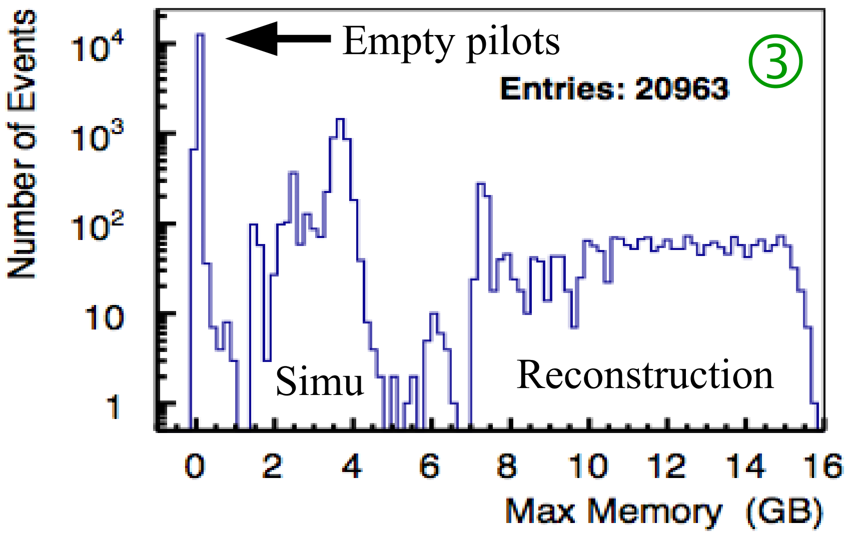
History of Multicore jobs



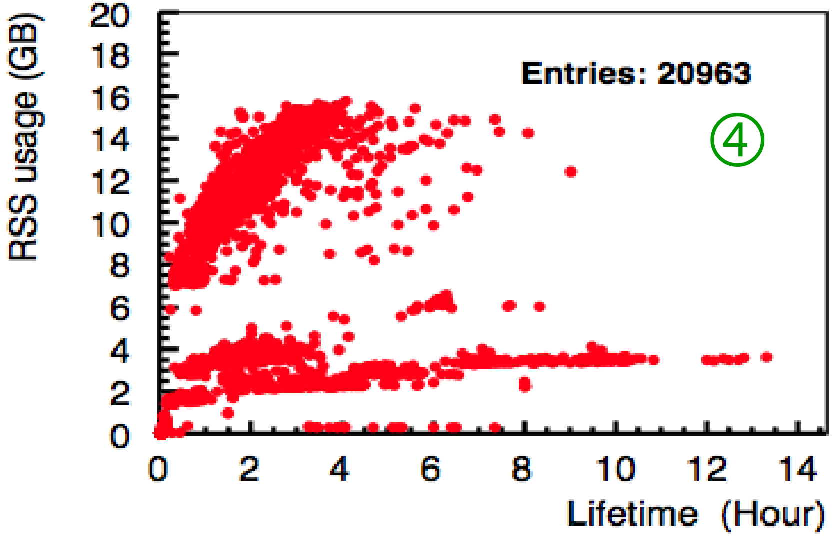
Distribution of Job Lifetime



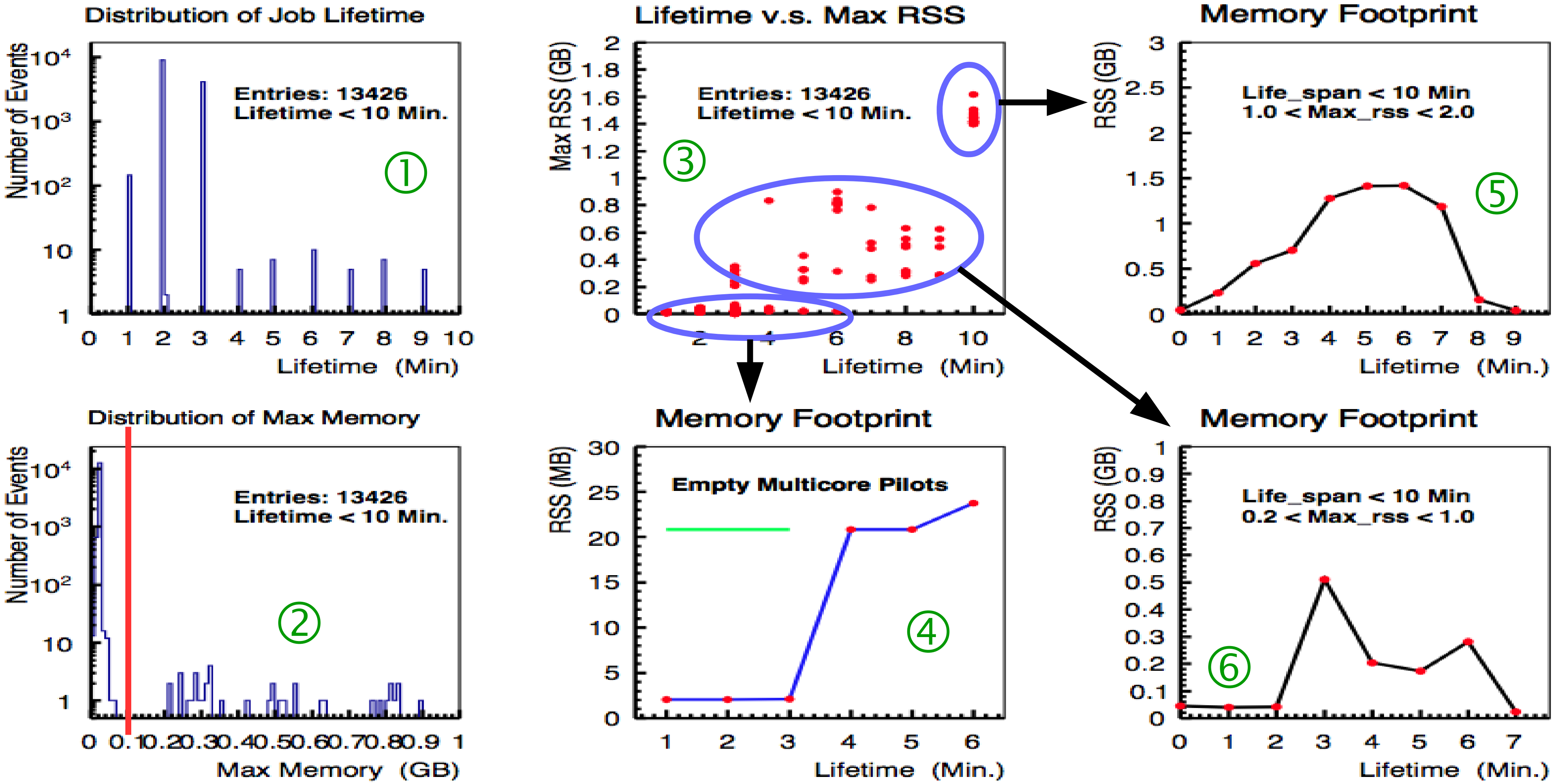
Distribution of Job Max Memory



Lifetime v.s. Max RSS



ATLAS Multicore Empty Pilots

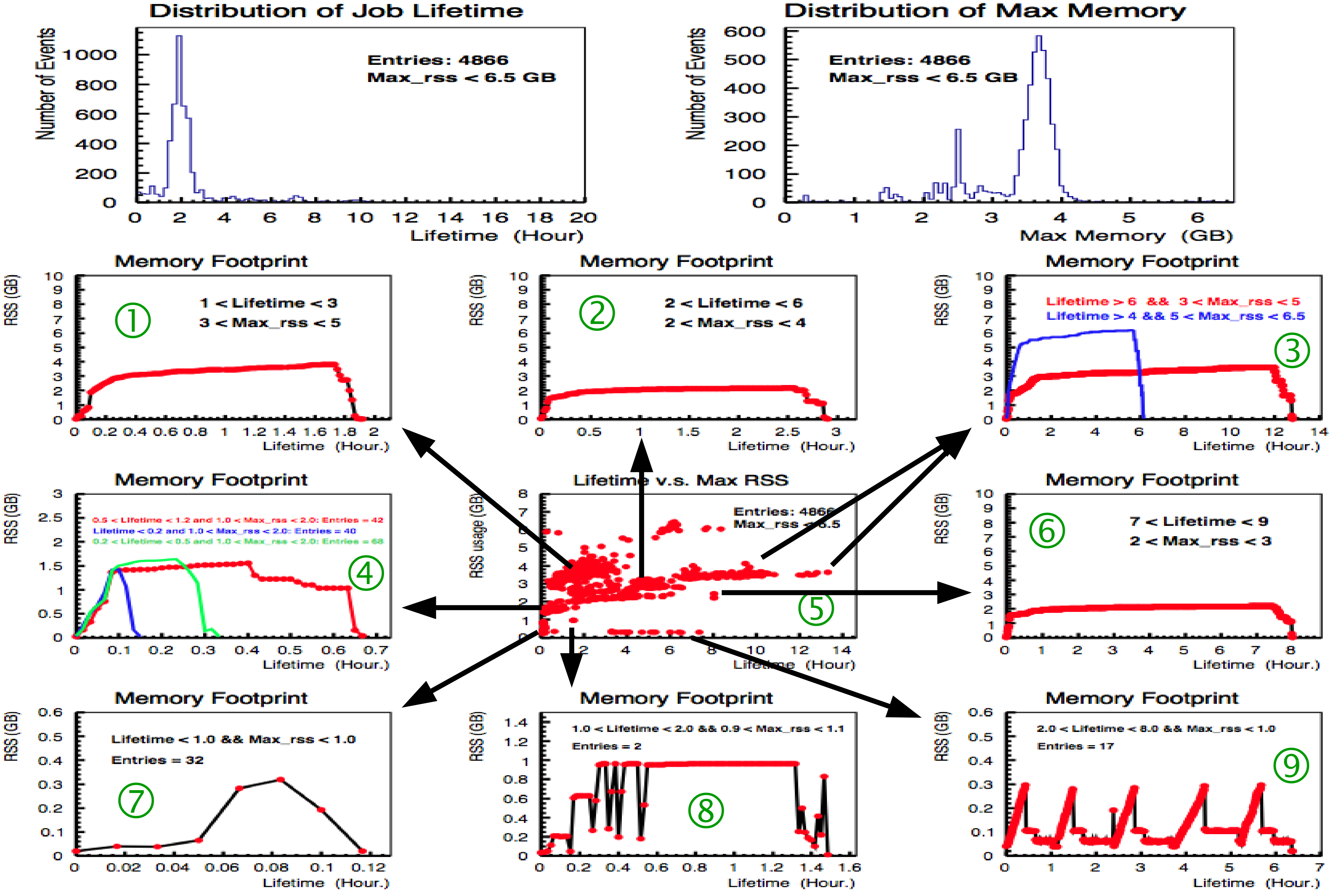


- ~ 2/3 jobs runs < 10 minutes
- In future analysis, we require jobs Lifetime > 3 minutes and Max_rss > 0.2GB

ATLAS Multicore Simulation Jobs

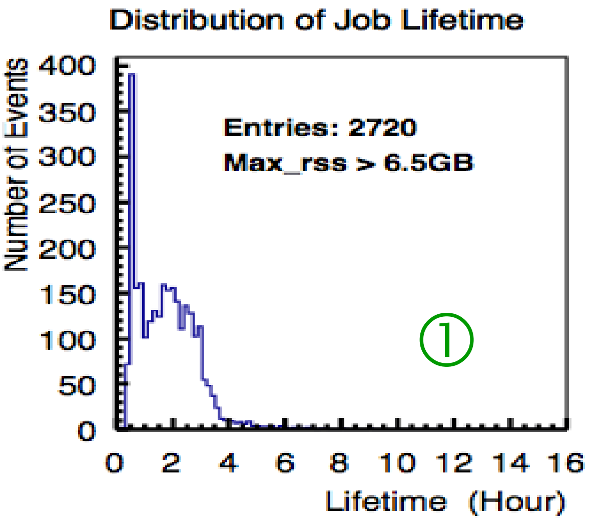
(7)

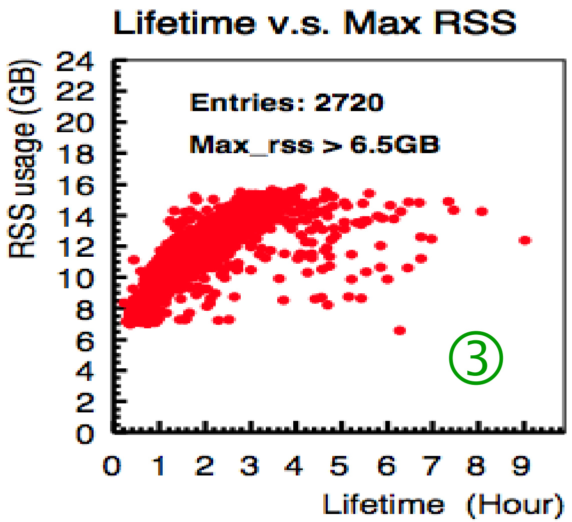
Max_rss < 6.5GB

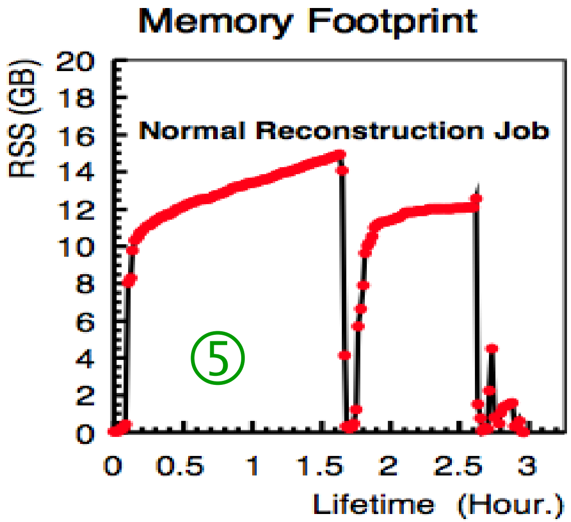


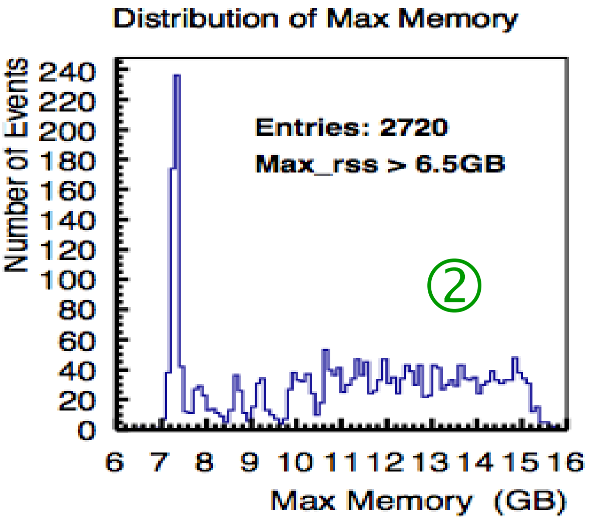
ATLAS Multicore Reconstruction Jobs

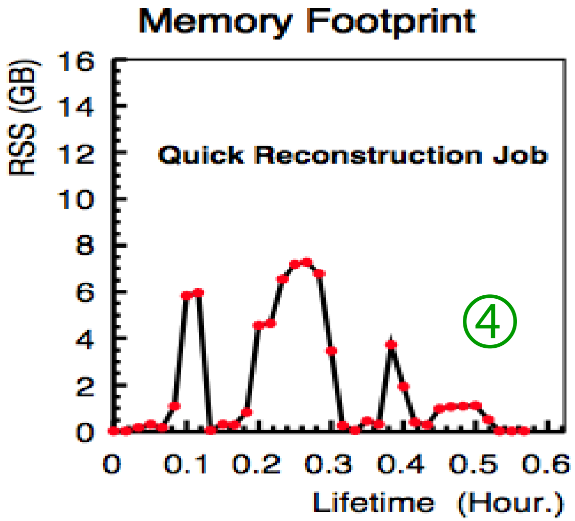
Max_rss > 6.5GB

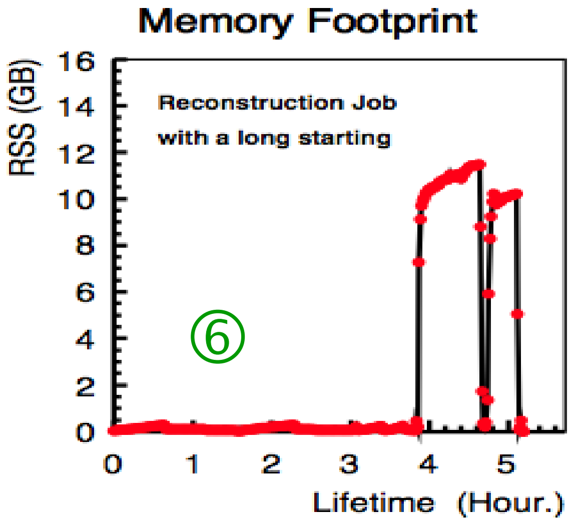

Distribution of Job Lifetime


Lifetime v.s. Max RSS


Memory Footprint


Distribution of Max Memory

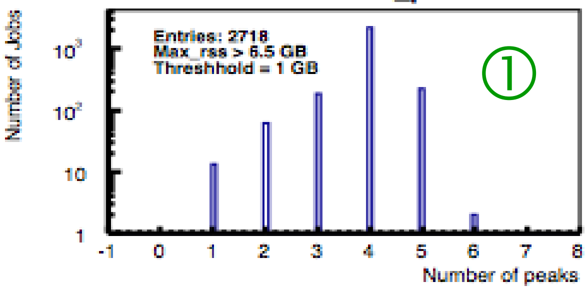

Memory Footprint


Memory Footprint

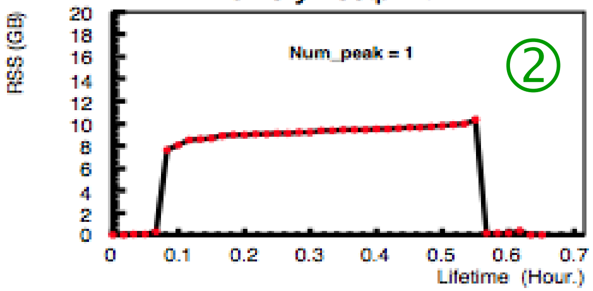
- Shape introduced for better identification
  - Studies on Number/Length of peaks with different thresh-hold

Thresh-hold = 1GB

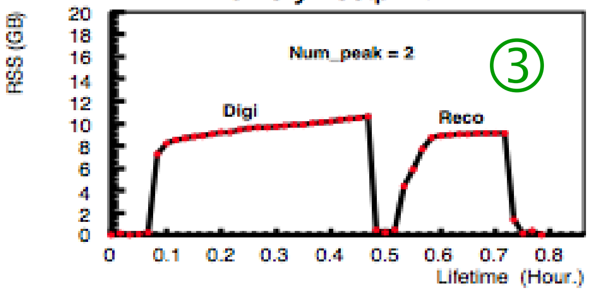
Distribution of Num_peaks



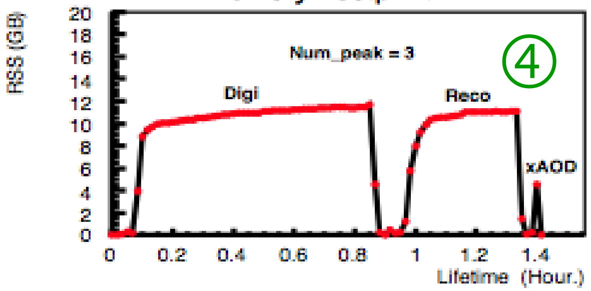
Memory Footprint



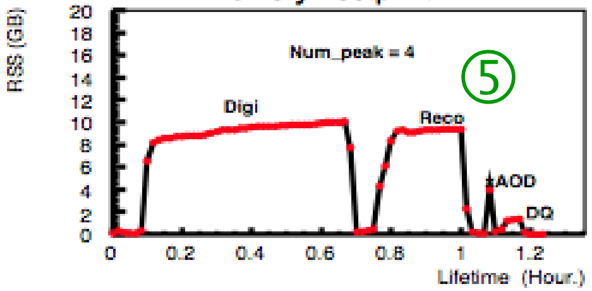
Memory Footprint



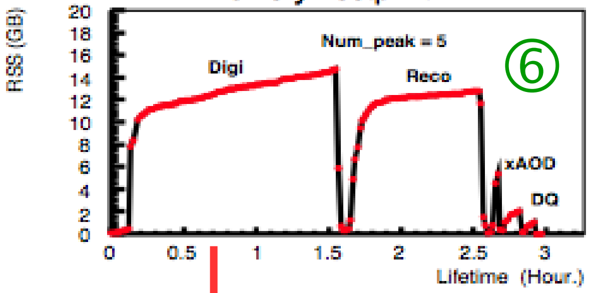
Memory Footprint



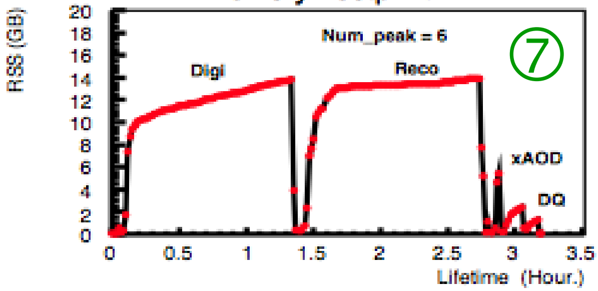
Memory Footprint



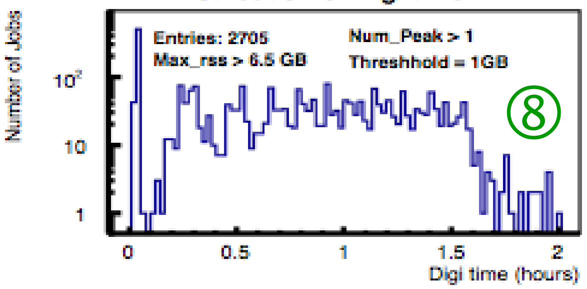
Memory Footprint



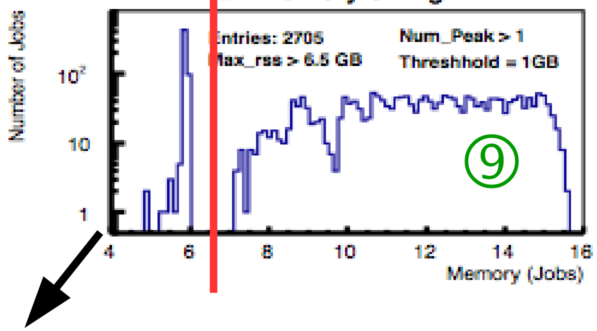
Memory Footprint



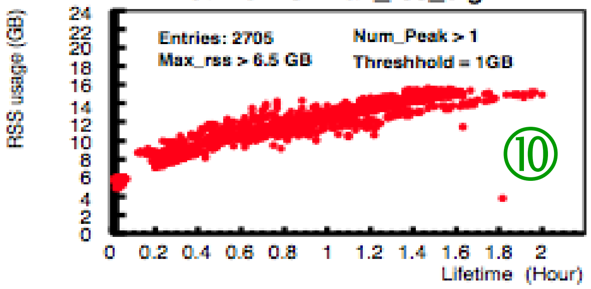
Distribution of Digi time



Max memory of Digi

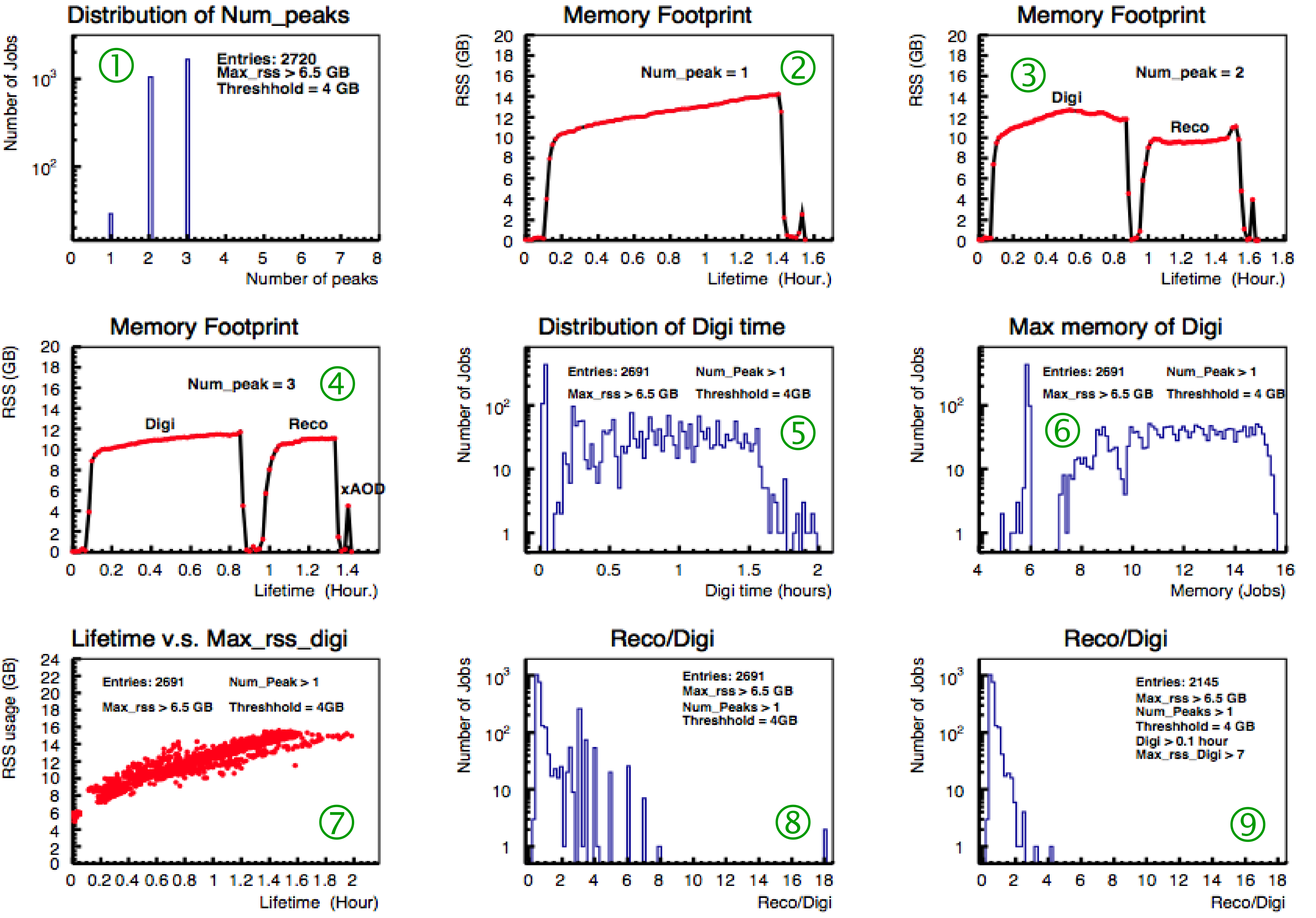


Lifetime v.s. Max_rss_digi

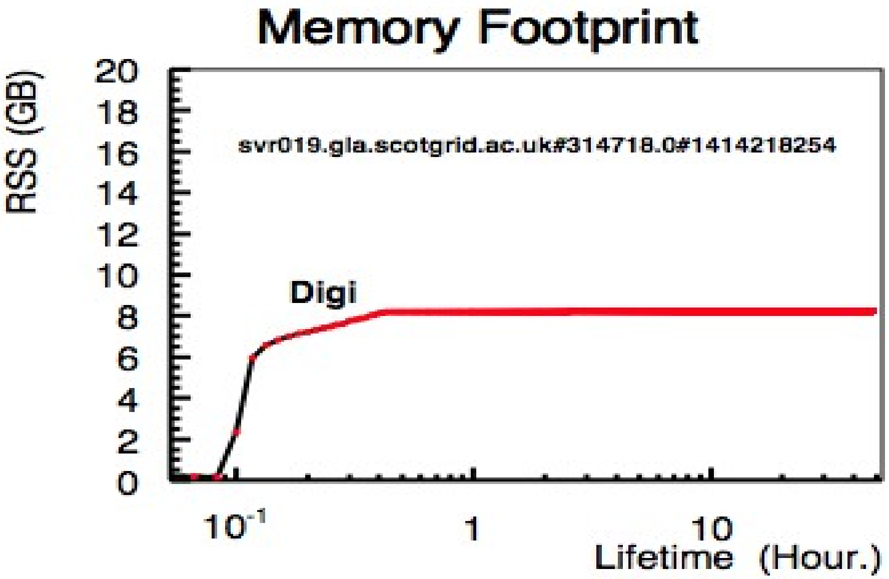


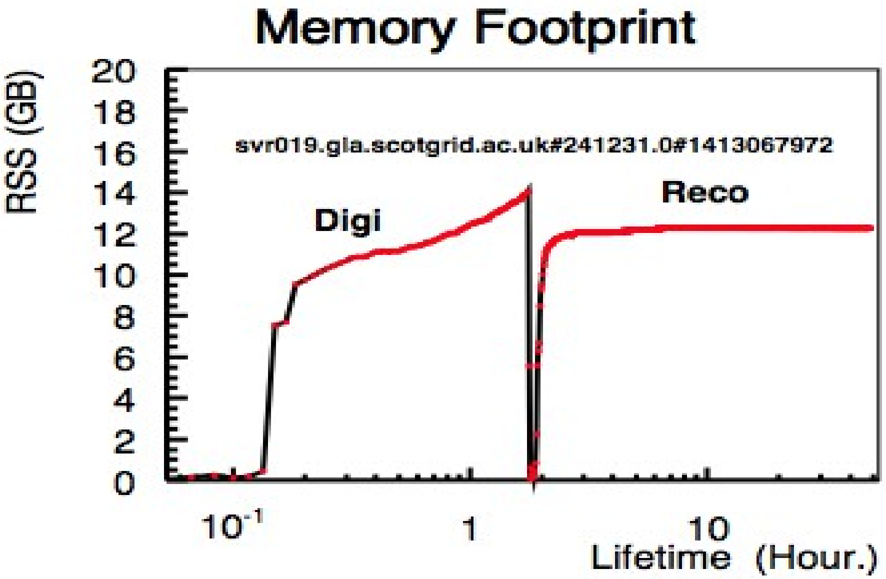
- Change threshold to 4GB

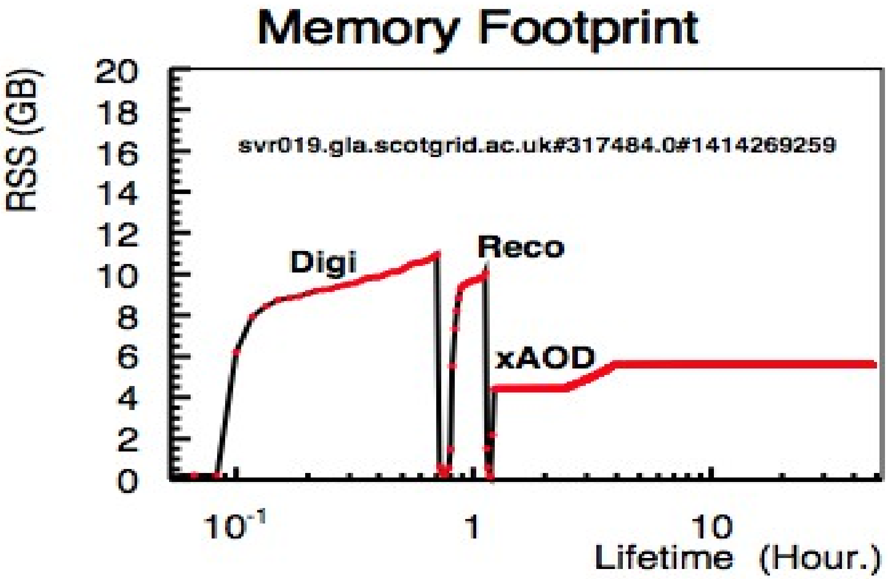
Thresh hold = 4 GB



Broken Multicore Jobs







- *Jobs could get broken at any step*
- *A broken job takes 48 hours while a normal multicore reconstruction job only takes ~ 2 hours*
- *A broken multicore jobs leads a loss of 384 cpu-hours*
- *Possible to be identified with its memory footprints*

Future Work

- Enrich Condor database
  - Some job info only exists on panda central monitoring page, frequent queries might crack down the database thus not allowed.
  - Use Cgmemd to retrieve more info from the logs of running jobs
- Further studies on more subsystems and more VO jobs
  - ATLAS
  - CMS
  - Small Vos: no good central monitoring, Machine learning techniques required
- Suspicious Job Detecting System
  - Jobs running too long become suspicious and it's recorded information in Cgroups could be used for further check
  - Periodical calibration required?
    - possibly Yes, depending on future studies
  - Integration with site monitoring/security tools

Questions?